IRIS

Jakubkovi, Kajke, Miškovi

Inteligencia ako výpočet

Ján Šefránek

IRIS

Vydal PhDr. Milan Štefanko — Vydavateľstvo IRIS

Vydané v rámci programu LIBRI ACADEMICI — Spoločný program Vzdelávacej nadácie Jána Husa a Nadácie otvorenej spoločnosti.

ISBN: 80-88778-96-4

Obsah

Ďakujem Slovenskému literárnemu fondu, ktorý mi udelil v roku 1994 tvorivé štipendium na písanie tejto práce.

Ďakujem Prof. MUDr. Milanovi Izákovi, CSc. Bez jeho pomoci by som túto knihu nemohol dopísať.

Predslov

Schopnosť poznávať a rozmýšľať fascinuje ľudí už tisícročia. S postupným zjemňovaním kultúry myslenia a s rastúcou hĺbkou, dôvtipnosťou reflexie myslenia sa objavovala predstava o príbuznosti až totožnosti rozmýšľania a počítania. Z dávnoveku týchto tendencií prenikavo žiaria dve osobnosti: Aristoteles a Leibniz. Obaja (a mnohí ďalší) podľa všetkého mali veľmi bezprostredný zážitok kombinovateľnosti (až automatizovateľnosti) základných myšlienkových operácií. Leibniz dokonca prišiel s nápadom vytvoriť univerzálnu vedu, v ktorej by sa odpovede na otázky hľadali výpočtami. Veril, že naše usudzovanie zlepšíme, ak ho urobíme tak priezračným, ako je matematické usudzovanie. Predstava o súvislosti výpočtov a usudzovania dozrieva spolu so zovšeobecňovaním pojmu výpočtu.

V dnešnom svete pojem výpočtu celkom prirodzene obsahuje akékoľvek spracovanie informácií. Neprekvapuje preto, že sa objavuje úsilie výpočtovo modelovať inteligenciu (inteligentné správanie, rozmýšľanie). Toto úsilie je jadrom disciplíny, zvanej umelá inteligencia. Predkladaný text je o výpočtovom modelovaní poznania a usudzovania. V centre pozornosti je schopnosť usudzovať hypoteticky: Formulovať domnienky vtedy, keď je naše poznanie neúplné. Vzdávať sa ich, revidovať ich vtedy, keď sa ukazujú byť neudržateľnými. Revízie sú pre optiku tejto knihy mimoriadne dôležité. Schopnosť modifikovať (názory, rozhodnutia, závery usudzovania, predstavy o perspektívnych alternatívach) vtedy, keď je to potrebné a adekvátne, je kľúčovou vlastnosťou intelektu.

Na druhej strane, systematický záujem o výpočty a prehlbovanie nášho poznania o výpočtoch ukazuje aj hranice (efektívneho) počítania. Umelá inteligencia sa zaujíma o množstvo problémov, ktorých výpočet je ťažký (prakticky nezvládnuteľný), prípadne nemožný. Špeciálne, štúdium hypotetického usudzovania je bohatým zdrojom problémov z výpočtového hľadiska veľmi ťažkých.

Na tomto mieste treba vysvetliť, v akom zmysle je text venovaný základom umelej inteligencie. Umelá inteligencia je oblasť, v ktorej sa stretávajú rozmanité typy ľudských aktivít (skutočne neobyčajne rozmanité). Má svoje rozmery vedecké i obchodnícke, básnické i inžinierske, pragmatické i mystické, teoretické i aplikačné. Pokiaľ ide o umelú inteligenciu ako vedu, jej charakter je nesporne interdisciplinárny a mnohotvárny. Rôzne oblasti umelej inteligencie (a prístupy k nej) sa výrazne líšia v závislosti od používanej konceptuálnej a metodologickej výbavy, od zamerania pozornosti, priorít, optiky a cieľov. Oblasti umelej inteligencie sú tak pestro diferencované, že si často nemajú vzájomne čo povedať. Presne ako ľudia. Preto nie je možné (domnievam sa) hovoriť o základoch umelej inteligencie ako celku.

Teda – v akom zmysle slova môžeme hovoriť o základoch umelej inteligencie v súvislosti s týmto textom? Po prvé, ide o základy *znalostnej paradigmy* umelej inteligencie. Mám na mysli prístup k umelej inteligencii, ktorého cieľom je pochopiť a výpočtovo modelovať spracovanie poznatkov (reprezentovaných vetami nejakého jazyka) a pochopiť usudzovanie (predovšetkým hypotetické). Ohraničujem tým pozornosť na tie systémy, prístupy a teórie, ktoré explicitne pracujú s nejakou bázou poznatkov a počas svojej činnosti odvodzujú dôsledky týchto poznatkov. Nechcem tým ani naznačiť, že iné, alternatívne prístupy k štúdiu a tvorbe výpočtových modelov inteligentného správania sú menej dôležité.

Po druhé, text je venovaný *teoretickým* základom tejto paradigmy. Jeho cieľom je predstaviť teóriu (jednu z možných) výpočtového modelovania poznania a usudzovania. Táto teória je ovplyvnená výskumami na pomedzí troch konvergujúcich oblastí: logického programovania, deduktívnych databáz a umelej inteligencie. Z hľadiska umelej inteligencie ide o štúdium tzv. nemonotónneho usudzovania (usudzovania, ktoré je sprevádzané príležitostnými revíziami prv odvodených záverov), všeobecnejšie možno hovoriť o hypotetickom, na kontexte závislom usudzovaní a o jeho dynamických aspektoch. Optika, ktorou sa k štúdiu problémov pristupuje, je (v rozhodujúcej miere) optikou logického programovania. To prináša do hry priezračný pojmový aparát a aj prijateľné prostredie pre špecifikáciu výpočtových modelov. Prostredie logického programovania umožňuje skĺbiť reprezentáciu a výpočty. Umožňuje v jedinom jazyku špecifikovať výpočty i reprezentovať poznatky. V porovnaní so staršími prístupmi ide o prechod od údajových štruktúr veľmi blízkych počítaču do úrovne väčšej abstrakcie, do úrovne konceptuálnej.

Pozornosť je zameraná na počítanie s vetami (a poznatkami). Abstrahuje sa od toho, aké okolnosti a procesy viedli k evolúcii intelektu a od toho, čo sa deje v našej nervovej sústave, keď rozmýšľame. Takýto uhol pohľadu je plne oprávnený. Denne sa stretávame s prejavmi rozmýšľania vlastného i iných ľudí. Tieto prejavy sú pozorovateľné vďaka nejakým vetám. Denne prijímame alebo odmietame nejaké argumenty; veríme alebo neveríme nejakým tvrdeniam; pozorujeme príklady vynikajúcej, ale aj biednej argumentácie; stretáme sa s tvrdeniami, ktoré možno veľmi ľahko vyvrátiť, ale aj s takými, ktoré odolávajú pokusom o falzifikáciu. Spomínané (a mnohé ďalšie podobné) pozorovania si zasluhujú analýzu i pokusy o výpočtové modelovanie tak ako každý iný výsek nášho prostredia. Ešte raz zdôrazním teoretický charakter knihy. Nemožno od nej čakať informácie o komerčných produktoch, ani návody, ako tvoriť aplikácie. Napriek tomu, nemožno pochybovať o praktickom význame dobrej teórie.

Domnievam sa, že teória, s ktorou sa čitateľ zoznámi, patrí k hodnotným výsledkom výskumu v umelej inteligencii. Oblasť, o ktorej hovoríme, je predmetom sústredenej pozornosti na popredných svetových pracoviskách.[1] Žiaľ, v našich končinách nie je dostatočne známa. Text sa usiluje o vyplnenie tejto medzery.

Kniha vznikala ako podklad prednášok o reprezentácii poznatkov a inferencii (usudzovaní) na Matematicko-fyzikálnej fakulte Univerzity Komenského v Bratislave. Jej rôzne verzie poznalo šesť ročníkov študentov informatiky (zaujímajúcich sa o umelú inteligenciu), vrátane jedného ročníka na Fakulte elektrotechniky a informatiky Slovenskej technickej univerzity. Text môže teda slúžiť ako učebnica alebo doplňujúce čítanie pre rôzne univerzitné kurzy umelej inteligencie.

Mojou ambíciou však je, aby text mohli čítať čitatelia rôzneho typu. Umelá inteligencia môže hľadať cenné podnety vo viacerých vedách, ktoré tradične – alebo menej tradične – študujú poznanie a intelekt (psychológia, filozofia, logika, lingvistika, kognitívna veda, neurofyziológia). Zdôrazniť treba úlohu analytickej filozofie. Jednou z jej charakteristických čŕt je tvorba formálnych modelov poznania a usudzovania. Umelá inteligencia sa od nej líši (z tohto hľadiska) iba dôrazom na výpočtové aspekty. Tvorcovia disciplíny, zvanej umelá inteligencia si to uvedomovali od samých počiatkov [MCH 69]. Interdisciplinárny význam umelej inteligencie je zosilnený okolnosťou, že aj informatika, matematika, biológia, fyzika, robotika a technické vedy dosahujú výsledky (a ponúkajú prístupy), relevantné pre pochopenie inteligencie. Navyše, vplyv týchto vied sa prejavuje na metódach a pojmovom vybavení tradičnejších disciplín, skúmajúcich poznanie a myslenie.

Podľa všetkého vplyv je vzájomný. Umelá inteligencia sa zdá byť zaujímavou a inšpirujúcou pre tradičné disciplíny. Dúfam, že aj táto kniha môže byť zaujímavá pre odborníkov v príbuzných disciplínach, i pre širšiu odbornú verejnosť, pre čitateľov s rôznym odborným pozadím.

Stručne o obsahu knihy. V kapitole 1 sú detailnejšie rozvedené základné východiská práce a jej optika. V kapitole 2 sa postupne buduje predstava o reprezentácii poznatkov. Buduje sa z jednoduchého základu – zo štandardnej databázy. Ukazuje sa, ako možno takúto jednoduchú reprezentáciu kompliko-

[1] Každý druhý rok sa uskutočňuje vrcholná celosvetová konferencia International Joint Conference on Artificial Intelligence (IJCAI). V deväťdesiatych rokoch je v zborníkoch tejto konferencie výrazným podielom zastúpená práve oblasť, ktorej sa tu budeme venovať.
V zborníku z IJCAI'99 [PAI 99] je to približne jedna štvrtina rozsiahlej dvojdielnej publikácie. Domovská stránka IJCAI je `http://ijcai.org`. Ďalšími dôležitými širšie zameranými konferenciami sú European Conference on Artificial Intelligence (ECAI, `http://www.ecai.org`), konferencie Americej asociácie umelej inteligencie (AAAI, `http://www.aaai.org`).

Pre prístup k základom umelej inteligencie, prijatý v tejto knihe, sú veľmi dôležité konferencie KR (Principles of Knowledge Representation and Reasoning, `http://www.kr.org`) a LPNMR - Logic Programming and Non-Monotonic Reasoning. Informácie o Special Interest Group on LPNMR možno nájsť na `http://www.cs.engr.uky.edu/~lpnmr`.

vať. Kapitolu 3 možno považovať za vlastný úvod do problematiky, ktorej je tento text venovaný. Jej jadrom je hypotetické usudzovanie, revidovateľné odvodzovanie z neúplného a nekonzistentného poznania. Možno povedať, že ide o centrálne aspekty výpočtového modelovania intelektu. Prvé dve kapitoly sú písané tak, aby čítanie kapitoly 3 bolo pripravené čo najrýchlejšie. Formálna dôslednosť a technické detaily sa nepovažujú za rozhodujúce, často sa zavádzané pojmy opierajú predovšetkým o príklady. Niekedy je to na úkor presnosti a dôkladnosti, verím však, že pozorný čitateľ si podrobnosti domyslí sám (ak to považuje za dôležité). Potrebné detaily môže nájsť v dodatkoch B až D.

Po všeobecnej charakterizácii hypotetického usudzovania sa niektorým jeho často študovaným príkladom venujem v kapitolách 4 a 5. Ide o defaultové teórie, odvodzovanie v hierarchických sieťach, indukciu a abdukciu. Po pomerne podrobnej analýze týchto prípadov sa prirodzene vynorí otázka o teoretickom rámci, ktorý by umožnil položiť sémantické základy pre štúdium hypotetického usudzovania. Takýmto rámcom môže byť logické programovanie. Z nášho hľadiska ide predovšetkým o spôsob, ako sa v logickom programovaní sémanticky špecifikuje narábanie s negáciou. Dôvod je jednoduchý – výpočet negácie v logickom programovaní veľmi pripomína falzifikáciu hypotéz. Téme sa venujeme v kapitole 6.

V kapitole 7 si podrobnejšie všimneme revízie, vzťahy rôznych formalizácií hypotetického usudzovania k revíziám, vzťahy medzi nimi navzájom a pokusy o jednotiace pohľady na hypotetické usudzovanie. Posledná je kapitola 8. Uzavrie jadro textu. Zhrnieme v nej predstavu o idealizovanom výpočtovom modeli inteligencie. V kontraste k teoretickému fundamentu, ktorému je text venovaný, načrtneme metódy a pojmy, smerujúce k praktickej realizovateľnosti výpočtových modelov inteligencie.

V texte je zozbierané značné množstvo všeobecne akceptovaných a známych výsledkov a teórií. Napriek tomu, poňatie témy, obsah a prístup je poznačený osobnými preferenciami a text obsahuje aj niektoré pôvodné prístupy a výsledky (ide najmä o časti 3.1, 3.2, 4.2.2, 4.2.3, 4.2.5 a 7.3).

Text je pomerne rozsiahly (dĺžka knihy je pravdepodobne na hranici zvládnuteľnosti, ak predpokladám, že ju niekto bude študovať). Záber je zámerne veľmi široký. Jedným z cieľov bolo dať prehľad o dosť širokej triede problémov a prístupov, ktoré sú na Slovensku takmer neznáme. Rozsah a šírka záberu však spôsobujú, že viaceré partie nebolo možné dotiahnuť do detailov. Niekde sú tvrdenia uvádzané bez dôkazov, niekde je iba naznačená základná myšlienka dôkazu.

Takmer každá kapitola obsahuje komentáre. Ich cieľom je orientovať čitateľa s hlbším záujmom a odkázať ho na literatúru. Vo všeobecnosti – často v texte myslím na čitateľa, ktorý bude hľadať zdroje ďalších, detailnejších a hlbších informácií. Kniha neukončí svoj vývoj dňom odovzdania do vydavateľstva. Doplnky, opravy a smerníky na ďalšie informácie sa budú objavovať na `http://www.ii.fmph.uniba.sk/kri/kri.html`. Moja e-mailová adresa je `sefranek@fmph.uniba.sk`. Som vďačný za akéhoľvek kritické pripomienky.

Moja ambícia je veľmi riskantná: Usilujem sa o rozsiahly text, predpokladajúci veľmi rozmanité druhy čitateľov. Preto musí obsahovať viaceré vrstvy, nie každá z nich je primeraná každému druhu čitateľov. O rôznych stratégiách čítania je pripravená samostatná informácia.

Mnohé dôležité oblasti, súvisiace s naším centrálnym problémom a optikou tu chýbajú. Patria medzi ne predovšetkým usudzovanie za prítomnosti neurčitosti, aproximatívne usudzovanie, reprezentácia času, priestoru, budovanie istej ontológie sveta, ktorý reprezentujeme, usudzovanie o akciách a ich účinkoch, usudzovanie v spoločenstvách agentov, komunikácia a kooperácia.

Nasleduje niekoľko drobností:

Ťažkosti mi spôsobovalo používanie prvej osoby v singulári a plurály. Usiloval som sa rozlišovať tak, že singulár používam tam, kde je jednoznačný osobný postoj, plurál v situáciách, keď si možno predstaviť spoločné úsilie (píšuceho a čítajúceho) pochopiť nejaký problém.

Text obsahuje veľa zátvoriek. Nedarilo sa mi ich likvidovať. Istým spôsobom kódujú hierarchiu a štruktúru (súvislosti myšlienok), niekde chcú slúžiť ako opora pre pamäť. Viem, že budú čitateľovi komplikovať život. Domnievam sa však, že ich odstránenie by viedlo k badateľnému nárastu textu.

Jazyková stránka textu: Zodpovedám za ňu sám. Niektoré z odchýliek od normy, štandardu sú zámerné. Nárokujem si právo na akúsi jazykovú licenciu, je to môj jazyk. Mám na mysli odbornú terminológiu, slovník i syntaktické konvencie. Na druhej strane, žiaľ, niektorým chybám, ktorým sa chcem vyhnúť, sa pravdepodobne nevyhnem.

Nakoniec by som sa veľmi rád poďakoval.

Mojim študentom. Mimoriadne som zaviazaný za množstvo cenných postrehov Broni Brejovej, Tomášovi Caletkovi a Petrovi Koleničovi. Veľmi mi pomohol v záverečnej fáze radami technickej povahy Ján Kľuka.

Okrem nich ďakujem za pripomienky a návrhy mojim bývalým študentom (niektoré dôležité mená mi mohli vypadnúť, ospravedlňujem sa): P.Hercek, B. Oliva, M. Takáč, H. Macko, J. Gašo, P. Maslen, J. Nagy, J. Ružarovský, M. Roháľová, G. Polčicová, K. Ostrovský, P. a P. Petrovičovci, P. Žuffa, O. Vršanský, T. Vinař, M. Nagy, M. Slašťan, P. Satury, P. Rajský, Š. Baloc, B. Květon, R. Cesnek, T. Kaláb, F. Matz.

Veľmi podnetnými a presnými postrehmi k poslednej verzii textu mi pomohli kolegovia Táňa Jajcayová a Dašo Gruska. Za pripomienky k niektorým častiam ďakujem Dane Pardubskej a Lacovi Kvaszovi. Za LaTeX-ovské konzultácie Jane Chlebíkovej. Za pomoc pri záverečných korektúrach ďakujem svojej žene Márii.

Ďakujem Ľube Beňuškovej, Jurajovi Hromkovičovi, Ivanovi Kalašovi, Vladimírovi Kvasničkovi, Lacovi Kvaszovi a Ľudovítovi Molnárovi za podporu vydania tejto knihy.

S vďakou si spomínam na ľudí, bez pomoci ktorých, by som sa asi nedostal k písaniu tejto knihy. Paľovi Cmorejovi ďakujem za dominantný vplyv na to, o čo som sa značnú časť môjho života usiloval. Milanovi Kolibiarovi za pomoc a povzbudenie v kľúčových okamihoch. Ďalej v chronologickom poradí: Eva Gedeonová, Zdenko Lapár, Fero Gliviak, Juro Steiner, Miki Popper, Jozef Kelemen, Puťo Mráz, Martin Bútora, Ľudovít Pezlár, Eduard Kostolanský, Katka a Andrej Fandákovci, Ján Rusnák, Ivan Haverlík.

Nakoniec – knihu venujem svojim vnúčatám: Jakubko sa narodil v čase, keď som s textom začínal (samozrejme, oveľa dôležitejšie bolo, že sa narodil on). V deň, keď som skončil písanie, Miško prišiel z pôrodnice domov. Medzitým sa narodila Kajka. Vďaka tomu, že som knihu písal v ich ohromnej spoločnosti, mi bolo vzácne jasné, že na svete existujú veci dôležité a menej dôležitejšie. Pochopiteľne, iba v prostredí, ktoré praje radosti, som mohol mať radosť z vnúčat. Ďakujem svojej žene, Márii, deťom Danielke a Braňovi, neveste Katke a zaťovi Braňovi. Viem, že to majú so mnou niekedy ťažké.

Návod na čítanie

Hneď na začiatku treba upozorniť, že nejde o ľahké čítanie. Ak sa čitateľ rozhodol venovať svoj čas a intelektuálne úsilie tejto knihe, pripájam niekoľko návrhov k možným spôsobom čítania.

Najprv krátky komentár o predpokladaných typoch čitateľov. Základný typ čitateľa je solídny študent informatiky vyšších ročníkov. Tento text bude primeranejší pre klasického univerzitného študenta. Navyše, často myslím na mimoriadne dobrého študenta, prípadne na adepta na výskum v tejto oblasti. Tým je latka náročnosti položená dosť vysoko. Text som sa však usiloval písať tak, aby bol osožný aj pre študenta informatiky technických a ekonomických univerzít. Domnievam sa, že aspoň jeho časti by mohli čítať aj odborníci (študenti), zaujímajúci sa o disciplíny, nejako súvisiace s umelou inteligenciou. Usiloval som sa text postaviť tak, aby sa v ňom mohol pohybovať aj čitateľ, ktorý sa nezujíma o detaily a o všetky formalizmy. Niekedy som aj načrtol možné obskočenia niektorých partií. Usiloval som sa stavať na pevných a zreteľných intuíciách, nielen z didaktických dôvodov.

Naznačil som, že predstava o rozmanitých čitateľoch viedla k viacerým vrstvám knihy. Jadro tvoria kapitoly 1 až 8. Venoval som značnú pozornosť tomu, aby jadro bolo čo najsebestačnejšie. Obsahuje veľa príkladov, pomerne dôkladné osvetľovanie intuícií. V samotnom jadre sú však rôzne vrstvy. Navyše, jadro je naviazané k mnohým dodatkom.

Potreba diferencovať rôzne vrstvy textu je zosilnená tým, že text chce byť v maximálnej miere sebestačný, usiluje sa minimalizovať predpokladané vedomosti. Ďalej, nechce zaťažovať zbytočnými detailami a formálnym výkladom pomocných partií. Preto obsahuje väčší počet dodatkov. Ich značná časť vlastne ani nie sú dodatky v pravom zmysle slova. Skôr by patrili na začiatok knihy ako nevyhnutný predpokladaný materiál. O umiestnení na konci, medzi dodatkami, rozhodlo to, že aj týmto spôsobom chcem upozorniť, čo považujem za dôležité a čo za technický predpoklad (alebo detail).

Dodatkom musím venovať podrobnejšiu pozornosť: Sledujem nimi viaceré ciele. Prvým cieľom je uľahčiť čitateľnosť knihy. Dodatky B, C, D, E, F, G, H sú pripravené pre čitateľa, ktorý pociťuje istý handicap v oblasti logiky, teórie relačných databáz, logického programovania, algebry, teórie množín, teórie vypočítateľnosti a výpočtovej zložitosti, prípadne pri čítaní algoritmov. Pre viacerých čitateľov bude dobré, keď budú čítať najprv dodatky (prípadne niektoré z nich), alebo aspoň keď ich budú čítať paralelne so zodpovedajúcimi kapitolami jadra textu.

Dodatky môžu slúžiť aj čitateľom, ktorí majú isté znalosti o spomínaných oblastiach – prinajmenšom ako príručka, ako prostriedok osvieženia pojmov, základných faktov, prípadne ich doplnenia. Niekedy po neformálnom výklade nejakej partie v jadre textu možno nájsť formálnejší výklad s presnými definíciami v dodatkoch. Napríklad kapitola 2 (o databázach) využíva viac metódu príkladov ako presné definície (kvôli rýchlemu postupu k dôležitým partiám). Čitateľ, ktorý cíti potrebu výklad doplniť formálnejším textom, môže nahliadať do dodatkov C, ale aj D a B. Teda, ďalším cieľom dodatkov je zhrnutie a technickejšie doplnenie menej technických partií z jadra textu. Týmto oddelením „technikalít" od jadra textu sa usilujem odlíšiť základy od detailov, myšlienky od prezentácie.

Usiloval som sa aj o to, aby čitateľ mohol čo najrýchlejšie postupovať dopredu cez témy, ktoré sú pre túto knihu podstatné. Kvôli rýchlejšiemu postupu vpred boli medzi dodatky zaradené aj A, I, J. Predstavujú rozšírenie, doplnenie jadra textu a do vlastného jadra neboli zaradené preto, aby nebolo príliš rozsiahle.

Niektoré pasáže z dodatkov a z jadra textu sa opakujú. Opakovaniu som sa neusiloval vyhýbať. Niekedy ide iba o pripomenutie synoným, o intuície alebo o jadro definície. Niekoho to môže rušiť. Mám však tie skúsenosti, že študentom vyšších ročníkov informatiky sa zišli (mnohé z nich vznikli na ich podnet). Predstavujem si istý druh „hypertextového" čítania (preskakovanie z miesta na miesto, riadené nejakými súvislosťami). Pri takomto čítaní opakovanie známych vecí nemusí byť na škodu. Pomôckou pri orientácii môže byť pomerne bohatý index a priame odkazy na dodatky v jadre textu. V dodatkoch bude výklad zväčša stručný, bude obsahovať iba málo motivačných partií. Tento nedostatok je – dúfam – vyvážený tým, že jadro textu obsahuje dosť motivácií a príkladov. Opakujem, predstava paralelného, alebo „hypertextového" čítania sa mi zdá optimálna. Zhustený formálny výklad z týchto dodatkov možno kombinovať s intuitívnejším, motivačným a na príkladoch založeným výkladom z jadra textu. Na druhej strane, čitateľ, ktorý narazí na neznámy pojem, môže siahnuť k dodatkom.

Tento text možno čítať rôznymi spôsobmi.

Najúspornejší spôsob čítania je asi ten, že sa čitateľ uspokojí s kurzívou na začiatku a konci každej kapitoly. Kapitolu 1 je pravdepodobne dôležité čítať kvôli pochopeniu zámerov a optiky autora. Na začiatku kapitoly 2 je krátky komentár k jej možnému preskočeniu. Kapitola 3 je asi dôležitá, ale mnohé partie, ktoré nasledujú, možno čítať aj bez zvládnutia jej detailov. Časť 4.2.5 však nadväzuje na časť 3.2.

Kapitoly 4 a 5 sú takmer nezávislé. Iba niektoré pasáže v časti o hierarchiách (4.2) podstatným spôsobom využívajú poznatky z časti o defaultových teóriách (4.1). Kapitola 6 motiváciou nadväzuje na časť 5.2, pre jej pochopenie však táto súvislosť nie je významná. Až na uvedené, kapitoly 4 až 6 možno čítať v ľubovoľnom poradí, prípadne aj s vynechávaním niektorých partií.

Posledné dve kapitoly majú za cieľ akési zhrnutie a vytvorenie (alebo aspoň hľadanie) jednotiaceho pohľadu. Asi bude ťažké čítať ich bez nadväzovania na jadro textu, i keď technicky sú relatívne nezávislé. V časti 7.1.3 sa využívajú poznatky z kapitoly 6.

Kapitola 1

Východiská

Cieľom úvodnej kapitoly je pripraviť východiskové pojmy a prijať základné rozhodnutia, nevyhnutné pre ďalší text.

Položíme si niekoľko otázok, najdôležitejšie sa týkajú vymedzenia pojmu výpočtového modelu inteligencie a pojmu reprezentácie znalostí.

Pretože umelá inteligencia je mnohovrstevná a mnohotvárna oblasť (nielen vedy[1] *a techniky, ale aj science-fiction i biznisu), dôležité je akceptovať isté ohraničenia a konvencie vzťahujúce sa na tému tejto práce a na optiku, ktorá sa bude používať. V zásade možno povedať, že sa budeme usilovať čo najviac zjednodušiť východiskové predstavy a opierať sa o nekomplikované, ale pomerne pevné intuície.*

Začneme s veľmi jednoduchou predstavou výpočtového modelu inteligencie. Potom sa budeme pýtať, kedy je výpočtový model zaujímavý z poznávacieho hľadiska a prijmeme zodpovedajúce konvencie.

Budeme sa venovať iba ohraničenej časti umelej inteligencie. Zúžime pozornosť na paradigmu znalostných systémov. Prijmeme rozhodnutia, charakterizujúce náš prístup a umožňujúce pracovať s produktívnymi abstrakciami. Zavedieme základné pojmy, späté s touto paradigmou (poznatky, odvodzovanie, reprezentácia znalostí).

Centrálnymi témami tejto knihy sú poznanie a odvodzovanie. Optika, ktorú pritom použijeme, bude determinovaná tým, čo by sme mohli nazvať výpočtovými modelmi inteligencie.

[1] Pokiaľ ide o umelú inteligenciu ako vedu, možno k nej pristupovať z hľadiska inžinierskeho, logického, informatického (a tu je mysliteľných tiež viac prístupov), z hľadiska biológie, neurofyziológie, kognitívnej vedy, psychológie, filozofie, aj z ďalších možných hľadísk.

S upresnením pojmu výpočtového modelu inteligencie si nebudeme v tejto chvíli robiť zvláštne problémy:

- Nebudeme sa venovať definovaniu inteligencie. Uspokojíme sa s tým, keď hodnoverný vonkajší zdroj označí nejakú činnosť (napríklad hranie šachu) ako inteligentnú, prípadne nejaké správanie (alebo triedu správaní) hodnotí ako inteligentné.

- Výpočtovým modelom bude pre nás program alebo jeho formálna špecifikácia.

Dohodnime sa pre začiatok:

Konvencia 1.1 Predpokladajme, že je daná charakterizácia inteligentného správania S. Predpokladajme ďalej, že program P vyhovuje tejto charakterizácii.[2] Potom P považujeme za *výpočtový model* správania S. ∎

Najpriamočiarejší prístup k realizácii výpočtových modelov inteligencie by spočíval v konštrukcii programu a nasledujúcom testovaní s cieľom ukázať, že jeho správanie vyhovuje danej charakterizácii. Komplikovanejší, ale adekvátnejší prístup venuje náležitú pozornosť špecifikácii programu. Nebude nás zaujímať ľubovoľný výpočtový model, teda ľubovoľný program, ktorý sa prejavuje inteligentným správaním. Z hľadiska umelej inteligencie ako vedy je dôležité, aby špecifikácia programu bola zaujímavá z poznávacieho hľadiska. Na výpočtový model nebudeme nazerať ako na čiernu skrinku. Naopak, preferujeme, keď mechanizmus, určujúci správanie výpočtového modelu, je dostupný pozorovaniu, skúmaniu.

V nasledujúcej konvencii sformulujeme podmienky, kladené na výpočtové modely. Tvorbu výpočtových modelov budeme považovať za súčasť empirickej, experimentálnej vedy. Treba tu upozorniť, že nemáme na mysli simuláciu prírodných alebo ekonomických procesov. Ide skutočne o experimentovanie – systematické pozorovanie vlastností nejakého programu v závislosti na zámerne pripravených okolnostiach (pozri napríklad experimenty z časti 4.3).

[2] Abstrahujeme tu od mnohých problémov. Čo je správanie programu? Ako ho identifikovať? Ako preveriť, či vyhovuje nejakej charakterizácii? Kedy toto preverenie možno považovať za pozitívne? Ako vlastne charakterizáciu vytvoriť? Uvedené problémy súvisia so špecifikáciou a verifikáciou programových systémov. Ide o fundamentálny problém informatiky, ktorý predstavuje oblasť intenzívneho bádania. Jeho dôležitosť je zdôvodnená vážnymi praktickými ťažkosťami, spojenými s identifikovaním významu programových celkov a verifikáciou korektnosti ich správania. Samozrejme, podstatné sú riziká používania nekorektných programov, najmä v oblastiach životne a ekonomicky kritických.

Obrazne, naprogramovať možno všetko – aj rozhodnuteľnosť formúl nerozhodnuteľného predikátového počtu, napríklad s využitím generátora náhodných čísel. Problémom však ostáva korektnosť takéhoto programu. Napriek tomu, že vedome obchádzame detailnejšiu diskusiu problémov okolo špecifikácie a verifikácie programových systémov, prijmeme konvencie a rozhodnutia, ktoré sa opierajú o súčasné tendencie v tejto oblasti.

Zavedenie spomínaných podmienok je motivované dobre osvedčenými kánonmi empirickej vedy: Experimentovanie je produktívne, ak je založené na nejakej koncepcii, hypotéze, ktorá určuje ciele experimentu a cesty k cieľu.[3]

Konvencia 1.2 Pozornosť budeme venovať iba takým výpočtovým modelom inteligencie, ktoré majú aspoň tieto vlastnosti :

1. sú podopreté explicitne akceptovanými princípmi (teóriou),
2. správanie výpočtového modelu sa dá predvídať na základe týchto princípov,
3. správanie výpočtového modelu možno nezávisle preveriť na jeho opakovanej konštrukcii,
4. odchýlky od očakávaného správania by sa mali dať vysvetliť a mali by viesť k návrhu modifikácií výpočtového modelu; teda od výpočtového modelu sa očakáva vývojaschopnosť.[4]

■

Čitateľa, ktorý by v tejto chvíli privítal konkrétnejšiu polohu, odkazujeme na dodatok A. Tam sa podrobnejšie venujeme programovému systému *AM* (Artificial Mathematician) [Len 76], ktorý na prelome sedemdesiatych a osemdesiatych rokov vzbudil veľký záujem a rešpekt. *AM* možno považovať za výpočtový model objavovania matematických pojmov a hypotéz. Detailnejší pohľad na jeho konštrukciu môže byť užitočný. Analýza jeho nedostatkov, ktorú čitateľ nájde v dodatku A, by mala podoprieť predstavu o želateľných vlastnostiach výpočtových modelov inteligencie.

Kritická analýza *AM* v [RHa 84] poukázala na to, že je dôležité, aby výsledky, ktoré sa dosahujú výpočtovým modelovaním, boli kontrolovateľné. Popisy výpočtových modelov majú byť tak detailné a transparentné, aby sa na ne dalo s dostatočnou istotou nadväzovať. Analýza *AM* prispela k formulácii vlastností nezávislej preveriteľnosti a vývojaschopnosti v konvencii 1.2.

V línii tohto textu má však analýza *AM* iba ilustračnú úlohu, i keď treba zdôrazniť, že je z nášho hľadiska nesporne zaujímavá.

1.1 Základné rozhodnutia

Ambíciu konštruovať programové systémy, ktorých správanie je inteligentné, možno uspokojovať veľmi rozmanitými spôsobmi. My sa tu budeme držať

[3] „Obratná ruka bez dirigujúcej hlavy je slepý nástroj“, napísal jeden zo zakladateľov modernej experimentálnej vedy, Claude Bernard.

[4] Túto konvenciu by sme mohli považovať aj za krok k vymedzeniu toho, čo je umelá inteligencia, rozhodne si však nerobím nároky na takéto niečo a ani sa mi ambície tohto druhu nezdajú príliš plodné.

paradigmy znalostných[5] systémov.

Zjednodušený model znalostného systému možno veľmi voľne načrtnúť takto (pozri napríklad [Guh 95]): Systém pozostáva z dvoch komponentov, bázy poznatkov a inferenčného stroja. Poznatky, ktoré obsahuje báza poznatkov, sú deklarácie. Deklarácie v tomto zmysle: konštatujú určitý stav vecí, formulujú nejaké hypotézy alebo zavádzajú definície pojmov, prípadne konvencie. Inferenčný stroj slúži na odvodzovanie dôsledkov poznatkov. Na ňom je vlastne založené inteligentné správanie – umožňuje z bázy poznatkov odvodiť to, čo v nej nie je priamo zaznamenané, ale je v nej implicitné.

Budeme sa teda venovať iba dvom častiam (zložkám) výpočtových modelov:

- reprezentovaným poznatkom,
- manipuláciám s poznatkami, odvodzovaniu.

Predpoklad, ktorý stojí za týmto zúžením poľa pozornosti: značnú časť ľudského inteligentného správania možno oprieť o nejaké dobre formulovateľné poznatky; odvodzovaním (usudzovaním) možno získavať ďalšie relevantné poznatky. Nasledujúca poznámka toto presvedčenie relativizuje.

Poznámka 1.3 (Inteligentné správanie) V tejto chvíli je namieste trochu sa pristaviť pri probléme inteligentného správania. U ľudí sa inteligencia správania prejavuje v rôznych kontextoch rôznym spôsobom a zdá sa, že mechanizmy, ktoré ju podmieňujú, sú veľmi rozmanité.

Inteligentný futbal môže hrať niekto, kto má problémy s trojčlenkou. Geniálny matematik môže hrať futbal veľmi tupo a bez invencie.

Podľa všetkého existujú rôzne typy racionality. Vezmime si príklad hľadania húb. Huby možno úspešne hľadať čisto reaktívne: smer prehľadávania budeme meniť podľa náhodných podnetov. Môžeme sa riadiť rozpomínaním na predchádzajúce prípady, keď sme našli huby (v inom prostredí). Môžeme však vychádzať aj z istých všeobecných pravidiel, ktoré hovoria o tom, na akom podloží obvykle možno nájsť huby a na základe týchto pravidiel voliť cestičky po lese. A napokon, v dôverne známom prostredí môžeme ísť priamo k dubu pod skalou, kde skoro zaručene nájdeme krásnu rodinku dubákov.

Niektoré z uvedených typov inteligentného správania možno charakterizovať nejakou bázou poznatkov a inferenčnými procedúrami, iné ťažko, ďalšie určite nie.

Potiaľ komentár k paradigme znalostných systémov a k typom racionality.

Bolo by príliš ambiciózne usilovať sa dostať do jediného výpočtového modelu viac typov racionality. Na druhej strane, nebolo by ľudské zamietať takúto perspektívu. ■

[5] Nechceme tým ani naznačiť, že by sme ju považovali za hodnotnejšiu, adekvátnejšiu ako iné prístupy. Jednoducho, iba jej budeme venovať pozornosť.

Teraz si osvetlíme hľadisko, z ktorého sa paradigme znalostných systémov budeme venovať. Prijmeme základné rozhodnutia, charakterizujúce optiku, ktorou budeme pristupovať k tomuto problému.[6]

Poznanie, znalosti, poznatky, inteligencia ostanú pojmami a predstavami v pozadí, s ktorými explicitne nechceme mať nič. Podobne, rozlišovanie medzi faktami, znalosťami, poznatkami, vedomosťami, dátami, informáciou (a podobné terminologické až filozofické problémy) nepovažujeme za dôležité pre naše ciele.

Naopak, vždy, keď použijem slová ako informácie, fakty, dáta, znalosti, poznatky, budem mať na mysli iba vety nejakého jazyka, pokiaľ z kontextu nevyplynie, že sa tieto termíny používajú v nejakom špeciálnejšom význame. *Poznatky (znalosti, poznanie) nás budú zaujímať potiaľ, pokiaľ sú vyjadrené (reprezentované) v nejakom formálnom jazyku.* Pre toto zúženie pozornosti máme dobrý dôvod. V konečných dôsledkoch predmetom našej pozornosti je, ako s reprezentáciou znalostí pracujú programové systémy. Pokiaľ programový systém nemá nástroje na verifikáciu viet, ktorým verí a na základe ktorých odvodzuje, nemá zvláštny zmysel pýtať sa, čomu v realite zodpovedajú vety, ktorými náš výpočtový model reprezentuje nejaké znalosti.[7]

V súvislosti s reprezentáciou znalostí si teda nebudeme všímať vzťah medzi znalosťami a svetom,[8] prípadne oddeľovať znalosti od jazyka, v ktorom sú vyjadrené.

To nemá znamenať, že pozornosť bude sústredená na syntaktické problémy. Ku každej vete možno priradiť nejaký význam. Medzi základné problémy pri štúdiu nejakého jazyka (a teda aj reprezentácie znalostí v tu uvedenom poňatí) patria sémantické problémy.[9] Značná časť pozornosti v tomto texte bude orientovaná práve na sémantiku. Teda, ak stotožňujeme znalosti s vetami, budeme venovať pozornosť aj sémantike znalostí (viet). Rozhodne však nemienime stotožniť sémantiku so svetom alebo s akýmisi znalosťami separovanými od jazyka. Sémantika nejakého jazyka nie je o realite, nanajvýš je o možných realitách, zodpovedajúcich danému jazyku.

---

[6] Nadväzujem na prvú formuláciu týchto rozhodnutí v [KSf 90].

[7] Od poznatkov sú neoddeliteľné spôsoby manipulácie s nimi – odvodzovanie. Opäť, nebudeme robiť zbytočné terminologické bariéry, i keď je veľa možností nachádzať zdôvodnené významové odtiene. Ako synonymá budeme používať slová ako odvodzovanie, inferencia, usudzovanie. V každom konkrétnom kontexte však špeciálny druh odvodzovania bude dostatočne presne definovaný.

[8] Táto abstrakcia teda vylučuje z poľa našej pozornosti reprezentáciu znalostí pre systémy, ktoré majú možnosť kontrolovať adekvátnosť vstupov, realizovať vlastné merania – napríklad roboty alebo vnorené systémy.

[9] Azda je potrebná terminologická poznámka o tom, ako sa chápe syntax a sémantika v tomto texte. Syntax nejakého jazyka je určená zoznamom jeho symbolov, pravidlami, ako vytvárať zložitejšie výrazy a pravidlami, ako odvodzovať jedny výrazy (vety) z iných výrazov (viet).

Sémantiku nejakého jazyka definujeme vtedy, keď jazykovým výrazom alebo ich postupnostiam priraďujeme nejaké objekty. Samozrejme, týmito objektami môžu byť nejaké formálne konštrukcie, dokonca aj jazykové výrazy. Dôležité je práve spomínané priradenie, nie povaha priradených objektov.

Predpokladajme teda, že je daný bližšie nešpecifikovaný formálny jazyk J. Naše poznanie je reprezentované množinou viet tohto jazyka. Spravidla iba časť tejto množiny vyjadríme explicitne a zvyšok dokážeme odvodiť (je vyjadrený implicitne). Je to pohodlnejšie, kladie to menšie nároky na prácu pri tvorbe bázy poznatkov a na priestor, potrebný na ich zapamätanie. Ďalším dôvodom pre uvedené rozlíšenie je, že prirodzene podporuje nároky na istý druh dôslednosti pri formulovaní (reprezentácii) poznatkov: ak čosi explicitne akceptujeme, pri úsilí o koherentnosť našich presvedčení by sme mali byť ochotní akceptovať aj to, čo je z nich odvoditeľné.[10]

Posledná prípravná poznámka sa týka odvodzovania. Odvodzovanie zatiaľ stotožňujeme s nejakými manipuláciami s poznatkami (výpočtami na poznatkoch). Aj tu bude dominantné sémantické hľadisko. Úlohou sémantiky bude definovať to, čo treba vypočítať. Inými slovami: špecifikovať vzťah odvoditeľnosti. Takýto prístup sa označuje prívlastkom *deklaratívny*. Výpočet nie je charakterizovaný postupnosťou príkazov (akcií). Výpočet je definovaný deklarovaním podmienok, ktoré spĺňa. (Na rozdiel od imperatívnej špecifikácie, ktorá určuje, *ako* výpočet prebieha, deklaratívna špecifikácia určuje, *čo* sa počíta.)

1.2 Reprezentácia znalostí

Reprezentáciou znalostí sa niekedy (napríklad podľa [DMy 86]) rozumie dvojica (E, Cn), kde E je množina viet jazyka J a Cn je operátor odvodenia,[11] ktorý množinám viet priraďuje množiny viet (jazyka J). Niekedy budeme hovoriť, že E je množina viet nad jazykom J. V každom jazyku, s ktorým budeme v tomto texte pracovať, je presne určená množina viet (spravidla ich budeme volať formulami).

Operátor odvodenia Cn začal v abstraktnej podobe študovať Tarski, pozri [Tar 56]. Ako vlastnosti charakteristické pre deduktívne odvodzovanie sa uvádzajú:

$$\begin{aligned} E &\subseteq Cn(E), \\ Cn(Cn(E)) &\subseteq Cn(E), \\ A \subseteq B &\Rightarrow Cn(A) \subseteq Cn(B). \end{aligned}$$

Prvá z nich (reflexívnosť) tvrdí, že pri odvodzovaní sa „nestratia“ informácie, z ktorých vychádzame. Druhá vyjadruje uzavretosť v tomto zmysle: Cn pracuje tak dôkladne, že po ukončení jeho „roboty“ už ďalšia aplikácia

[10] Samozrejme, môžu vznikať problémy s tým, ako silné odvodzovacie schopnosti špecifikujeme a realizujeme. Niekedy sa stáva, že odvoditeľné vety nemusia byť intuitívne akceptovateľné. V texte často narazíme na príklady nejakého druhu odvodzovania, ktorý vedie k neintuitívnym záverom. Vždy budeme aj diskutovať možnosti odstránenia tejto neintuitívnosti. Vyvažovanie špecifikácií a intuícií patrí medzi základné úlohy teórie reprezentácie znalostí.

[11] Symbol Cn je skratkou slova *consequence*.

Cn neprinesie nič nového. (Čitateľ iste postrehol, že z uvedených vlastností operátora Cn vyplýva, že $Cn(Cn(E)) = Cn(E)$.) Tretia vlastnosť sa nazýva monotónnosťou. Hovorí, že pri odvodzovaní z rozšírenej množiny predpokladov nestratíme žiaden z dôsledkov pôvodnej množiny predpokladov.

Časom uvidíme, že v niektorých situáciach nám nebudú vyhovovať vlastnosti, ktoré sformuloval Tarski. Zoznámime sa s viacerými operátormi odvodenia. Ich vlastnosti sa budú výrazne líšiť od vyššie uvedených. Napríklad, budú nás zaujímať také prípady odvodzovania, kedy pre dve množiny predpokladov A, B, kde $A \subseteq B$, neplatí $Cn(A) \subseteq Cn(B)$. Takéto odvodzovanie budeme nazývať nemonotónnym. Jednou z centrálnych otázok tohto textu je: aké sú dôvody študovať nemonotónne odvodzovanie? Budeme sa venovať aj revíziám. Tie nespĺňajú vlastnosť (1). Narazíme aj na odvodzovací operátor Cn, ktorý nespĺňa vlastnosť (2). Preto od tejto chvíle nepredpokladáme nijaké vlastnosti operátora Cn, pokiaľ na to zvlášť neupozorníme.

Symbol Cn sa obvykle používa iba v súvislosti s operátorom deduktívneho odvodenia, v tomto texte bude jeho použitie širšie, bude sa vzťahovať na akékoľvek odvodenie. Prípadné špeciálne prípady sa budú označovať indexami, napríklad Cn_{ded}.

Po zavedení operátora Cn ešte raz zdôrazníme rozlišovanie explicitných a implicitných znalostí. Vety z E môžeme nazvať explicitnými znalosťami. Vety z $Cn(E) \setminus E$ implicitnými znalosťami.

Charakterizovať reprezentáciu znalostí iba dvojicou (E, Cn) však nie je celkom adekvátne. Inteligencia nejakého systému závisí od toho, nakoľko využíva asociácie medzi znalosťami. Izolovaná znalosť, izolovaná veta sa stáva užitočnou až vtedy, keď ju vieme zaradiť do vhodného celku. V reprezentácii znalostí výpočtového systému môžeme takúto schopnosť dosiahnuť vtedy, keď dokážeme vytvárať *zhluky* znalostí. Nejaký druh organizácie poznatkov umožňuje vytvárať zhluky súvisiacich poznatkov a nachádzať medzi nimi asociácie.

Opíšme si teraz jednu z možností, ako vytvárať zhluky. Urobíme to tak, že k nejakému jazykovému výrazu z E alebo z $Cn(E)$ priradíme množinu viet. Toto priradenie nazveme *indexovaním*. Indexovanie je – okrem vytvárania zhlukov – užitočné aj z výpočtového hľadiska: uľahčuje, zefektívňuje prístup k tým vetám, ktoré sú v danej chvíli odvodzovania práve dôležité. Zdôvodnili sme si teda, že ďalším užitočným prostriedkom reprezentácie poznatkov je indexovací mechanizmus.[12]

Potrebujeme však presnejší opis indexovacieho mechanizmu. Budeme predpokladať, že k reprezentácii poznatkov patrí aj trieda funkcií[13] Ind. *Indexom* nazveme každú z funkcií $f \in Ind$. Definičný obor funkcie f si označme ako

[12]Ak čitateľ potrebuje ilustráciu, môže ju nájsť v dodatku A: v Lenatovom systéme AM sa pri reprezentácii znalostí účelne využíva indexovací mechanizmus. Sú ním rámce, ktoré vlastne zhlukujú súvisiace informácie.

[13]Využijem pojem funkcie a čitateľa upozorním na konvenciu, ktorú práve zavádzam. Medzi znakmi „⟨“ a „⟩“ budú odkazy na dodatky, kde možno nájsť doplňujúcu informáciu. Budú vyzerať takto: ⟨ *funkcia*: dodatok E ⟩.

$Dom(f)$ a množinu jej hodnôt ako $Range(f)$. Predpokladáme, že definičný obor funkcií z triedy Ind je ľubovoľná množina. Nebudeme nijako obmedzovať možné argumenty indexovacej funkcie. V typickom prípade to môže byť množina viet, ale aj množina jednoduchších elementov jazyka.

Predstavme si, že aplikujeme funkciu f na nejaký objekt $a \in Dom(f)$. Intuitívne, hodnotou $f(a)$ by mala byť množina viet asociovaných s jazykovým výrazom a. Indexovacie funkcie každému svojmu argumentu priradia nejakú množinu viet (aj prázdnu alebo jednoprvkovú). Značenie: nech $f \in Ind$, potom pre $f(a) = X \subseteq E$ zapisujeme $X \in 2^E$ a $Range(f) \subseteq 2^E$, kde 2^E je množina všetkých podmnožín E.

Definícia 1.4 *Reprezentácia poznatkov* je štvorica (J, E, Cn, Ind), kde J je nejaký jazyk, E je množina jeho viet, Cn je operátor odvodenia a Ind je množina indexov.

Konvencia 1.5 *Problémom reprezentácie poznatkov* budeme rozumieť túto štvoricu podproblémov:

- voľba existujúceho alebo špecifikácia a konštrukcia nového jazyka J na reprezentáciu poznatkov,
- formulácia množiny viet E (bázy znalostí) v tomto jazyku,
- voľba (alebo špecifikácia a konštrukcia) operátora odvodenia Cn,
- návrh a konštrukcia množiny indexov Ind.

1.3 Vyjadriteľnosť a efektívnosť

Zaviedli sme pojem reprezentácie poznatkov, s ktorým budeme ďalej pracovať a prijali sme konvenciu o probléme reprezentácie poznatkov. Táto konvencia má postihovať dôležité úlohy pri návrhu a realizácii výpočtového modelu inteligencie, založeného na paradigme znalostných systémov. Všimneme si dva základné aspekty reprezentácie poznatkov: vyjadrovaciu silu a výpočtovú efektívnosť. Budeme sa zaujímať o ich vzájomné vyvažovanie.

Potom si všimneme ďalšie dva dôležité aspekty problému reprezentácie znalostí, ktoré umožnia hodnotiť kvalitu reprezentácie. Nadviažeme tým na konvenciu 1.2 o výpočtových modeloch inteligencie. Tam sme odlíšili princípy, na ktorých model stojí, od správania modelu. Dva aspekty reprezentácie znalostí, ktoré im zodpovedajú, sú špecifikácia a implementácia. Budeme sa venovať otázke adekvátnosti špecifikácie a správnosti implementácie.

Na probléme reprezentácie znalostí môžeme rozlíšiť dva dôležité aspekty.

- Prvý z nich sa sústreďuje okolo obsahu poznania, ktoré reprezentujeme pre potreby výpočtového modelu inteligencie. Ide vlastne o určenie, čo patrí do E či $Cn(E)$. Pýtame sa, aké znalosti sú reprezentované, čo všetko je vyjadrené v E či $Cn(E)$. Môžeme sa aj pýtať, čo je vlastne reprezentovateľné, vyjadriteľné. Táto trieda otázok súvisí s tým, čo Hayes a McCarthy nazvali *epistemologickým aspektom* umelej inteligencie [MCH 69].

- Druhým je to, čo McCarthy a Hayes nazvali heuristickým aspektom umelej inteligencie. Rozhodujúce je tu hľadisko výpočtovej efektívnosti. Zaujímavé a kľúčové je, či existuje efektívny výpočtový mechanizmus, ktorý pre daný jazyk dokáže efektívne implementovať odovodzovaciu mašinériu Cn. Pri efektívnej implementácii Cn môže dôležitú úlohu hrať množina Ind.

Zhrňme: Jedným z kľúčových momentov pri riešení problému reprezentácie poznatkov je vyvažovanie vyjadrovacej sily a výpočtovej efektívnosti. Spravidla, väčšia vyjadrovacia sila vedie k znižovaniu výpočtovej efektívnosti (a naopak). Voľba medzi vyjadrovacou silou a výpočtovou efektívnosťou záleží na konkrétnej inštancii problému reprezentácie znalostí. Očividne, dôležitý trend je študovať a hľadať formalizmy, ktoré umožňujú vhodný kompromis. Epistemologický a heuristický aspekt predstavujú dve konfliktné a navzájom prepojené stránky problému reprezentácie znalostí.

Ďalší komentár je potrebné venovať dynamickej povahe reprezentácie znalostí. Epistemickú adekvátnosť reprezentácie poznatkov spravidla dosahujeme postupne. U výpočtových modelov inteligencie sa predpokladá postupný vývoj, postupná modifikácia reprezentovaných poznatkov.

Vráťme sa preto k úlohe formulovať množinu viet E v jazyku na reprezentáciu znalostí. Táto úloha nie je jednorázová, ale permanentná. Ľudské poznanie (a aj výpočtový model inteligencie) má dynamický charakter. Časom sa mení, v niečom dopĺňa, v niečom odmieta. Asimilácia novej informácie a adaptácia starej je jednou z dôležitých schopností výpočtového modelu inteligencie. Problému revízií sa budeme podrobne venovať, uvidíme aj jeho tesnú súvislosť s komplikovanejšie špecifikovanými operátormi Cn. Typ usudzovania, pri ktorom sú revízie neodmysliteľným komponentom, nazveme *hypotetickým usudzovaním.*

Prejdime teraz ku vzťahom špecifikácie a implementácie, k otázke adekvátnosti špecifikácie a správnosti implementácie. V konvencii 1.5 sme používali kľúčové slová voľba, špecifikácia, konštrukcia, návrh, formulácia. Tieto slová naznačujú, že s problémom reprezentácie poznatkov možno združiť dve rozhodujúce fázy: *špecifikačnú* (voľba, výber, návrh, formulácia, špecifikácia) a *implementačnú* (realizácia, konštrukcia). Vznikajú tu dva problémy verifikácie: či je špecifikácia adekvátna a či implementácia vyhovuje špecifikácii. Môžeme používať termíny *adekvátnosť špecifikácie* a *správnosť implementácie.*

Z hľadiska umelej inteligencie je verifikácia správnosti implementácie iba technickým problémom: Ak špecifikácia nie je implementovaná správne, experimentovanie s vytvoreným programom nie je relevantné pre preverovanie teórie, ktorá je v pozadí za zrealizovaným programom.

Podstatným problémom z hľadiska výpočtového modelovania inteligencie je adekvátnosť špecifikácie. Tým, že sme špecifikovali poznatky a prípustné manipulácie s nimi, vytvorili sme teóriu, ktorú má výpočtový model (programový systém) preveriť. Ak je implementácia správna, má zmysel verifikovať, či sa výpočtový model správa podľa očakávania. Výpočtové modely inteligencie sú často veľmi zložité a možno ich posudzovať z rôznych hľadísk. Preto niekedy až experimentovanie s bežiacim programom umožní dôkladné preverenie rozmanitých dôsledkov navrhnutej teórie vo viacerých kontextoch a návrhy na jej prípadné modifikácie. Špecifikované správanie nemusí byť vždy uspokojujúce. Najmä vtedy, keď experimentujeme na dostatočne komplexných situáciách a problémoch. Samozrejme, mnohé neadekvátnosti špecifikácie sa dajú odhaliť čistou myšlienkovou analýzou.

Niekoľko poznámok treba venovať technike špecifikácie. Je potrebná taká technika špecifikácie, ktorá by bola dostatočne abstraktná a dobre prezentovateľná, teda vzdialená od implementačnej úrovne.[14] Súčasne však musí byť dostatočne detailná, aby charakterizovala problém a jeho riešenie.

Náš prístup k technike špecifikácie ovplyvňujú aj základné rozhodnutia, ktoré sme prijali o reprezentácii znalostí. Budeme venovať pozornosť takým jazykom na reprezentáciu znalostí, ktorých syntax a sémantika je jasne definovaná. Osobitnú pozornosť treba venovať špecifikácii Cn. Budeme sa usilovať o sémantickú špecifikáciu Cn. Príklad AM (pozri dodatok A) ukazuje, že odvodzovanie znalostného systému môže byť veľmi komplikované a ťažko opísateľné. Dokonca môžu vznikať odborné spory o jeho skutočnom správaní. Sémantická charakterizácia Cn vytvára dobré predpoklady pre riešenie týchto komplikácií. V tejto chvíli sa však nebudeme podrobnejšie venovať tomu, čo máme na mysli pod sémantickou charakterizáciou Cn.

Okrem toho, že sémantická charakterizácia Cn umožňuje preverovať správnosť implementácie operátora Cn, umožňuje – skôr, než sa začne s implementáciou – analyzovať, aké odvodzovanie daná sémantika dovoľuje. V prípade, keď sa sémantická charakterizácia nezhoduje s intuíciami, treba ju modifikovať.[15]

Reprezentácii znalostí a inferencii vo výpočtových modeloch inteligencie sa môžeme venovať aspoň dvoma spôsobmi: buď máme ambíciu, aby model postihol mechanizmy ľudského správania, alebo sa uspokojíme s tým, že

[14] Tento názor nie je ani dnes všeobecne prijatý. Napríklad podľa Simona, pozri [Sim 95], teóriou, ktorá je v pozadí za experimentálnym výskumom v umelej inteligencii, je program. Rôzne implementácie tej istej idey pokladá za rôzne teórie. Samotná otázka, čo sú rôzne implementácie, je ťažká. Ak nechceme prijať pozíciu, že každé dva syntakticky odlišné programy sú rôzne, táto otázka je nezodpovedateľná bez dobre definovanej sémantiky. Logické programy majú dobre definovanú sémantiku, problém ich ekvivalencie je však nerozhodnuteľný. Navyše, treba rozlišovať, ktoré vlastnosti programov sú relevantné z hľadiska daného výpočtového modelu inteligencie a ktoré nie. Tým sa dostávame k potrebe jasne odlíšiť program a teóriu, ktorá je v jeho pozadí.

[15] Zámerom sústreďovať sa na sémantickú špecifikáciu Cn reflektujeme kľúčový význam problémov súvisiacich so špecifikáciou a verifikáciou programových systémov (a tým aj výpočtových modelov inteligencie).

preukazuje požadované správanie a nestaráme sa o to, či jeho konštrukcia zodpovedá biologickým alebo psychologickým danostiam. Teraz môžeme pridať aj tretí spôsob. Reprezentáciu znalostí (spolu s inferenciou) sme vymedzili ako štvoricu (J, E, Cn, Ind), preto k problému reprezentácie znalostí a inferencie môžeme pristupovať ako k problému počítania s vetami. Výpočty sú špecifikované pomocou Cn a Ind, ich argumentami sú vety jazyka J, v ktorom je vyjadrená báza znalostí E. Týmto prístupom aj začneme. Až neskôr bude stále častejšie vystupovať otázka adekvátnosti špecifikovaného počítania s vetami.

Pri našom výklade problematiky znalostí budeme vychádzať z databázového fundamentu. Ukážeme, že znalosti (v prirodzenom zmysle slova) sú uložené aj v databázach a že nie je priepastný rozdiel medzi takto chápanými znalosťami a tým, čo sa znalosťami nazýva v znalostných systémoch. Databáza obsahuje znalosti v presne tom istom zmysle ako iné, komplikovanejšie systémy, ktoré spracúvajú znalosti. Rozdiel, ktorý môžeme jasne odlíšiť, je nasledovný: jazyk, použitý na reprezentáciu znalostí v databázach, je jednoduchší a odvodzovacia mašinéria slabšia.

V nasledujúcej kapitole si preveríme, nakoľko produktívne sú naše základné rozhodnutia. V tomto preverovaní budeme pokračovať tak, že postupne budeme komplikovať predstavu o databáze, kým nedospejeme k typu databázy, ktorý môže byť dostatočne zaujímavý pre netriviálne problémy umelej inteligencie. Na ceste za týmto typom databázy uvidíme, aké znalosti sú reprezentované v relačnej databáze a v deduktívnej databáze. Potom vytvoríme konštrukt databázy s hypotézami. V nej budú kľúčovými schopnosti ako nemonotónne odvodzovanie, generovanie hypotéz, práca s neúplným a nekonzistentným poznaním. Tento konštrukt (s uvedenými a im podobnými vlastnosťami) sa stane typom databázy, na ktorom budeme ďalej stavať, rozvíjať predstavu o ňom.

Prijali sme konvencie o tom, čo považujeme za výpočtový model inteligencie, o vítaných vlastnostiach takýchto modelov a o tom, čo považujeme za problém reprezentácie poznatkov.

Rozhodli sme sa pre prísnu rozlišovaciu a pojmovú jednoduchosť. Znalosti stotožňujeme s vetami nejakého jazyka. Nebudeme rozlišovať rôzne druhy znalostí a nebudeme zavádzať terminologické komplikácie, pokiaľ pre to nebudú ostré syntaktické a vecné dôvody. Reprezentáciu znalostí sme stotožnili so štvoricou (J, E, Cn, Ind), kde E je množina viet nejakého jazyka J, Cn operátor odvodenia a Ind množina indexovacích funkcií. Odlíšili sme problém adekvátnosti špecifikácie reprezentácie poznatkov a verifikácie jej implementácie. Rozhodli sme sa klásť dôraz na sémantickú špecifikáciu Cn. Uvedené abstrakcie a rozhodnutia umožňujú nazerať na problém reprezentácie znalostí a inferencie ako na problém počítania s vetami.

Kapitola 2

Databázy

Budeme pátrať po znalostiach reprezentovaných v databáze. Charakterizujeme jednoduchý jazyk, v ktorom sú vyjadrené vety štandardnej databázy. Predovšetkým si však všimneme jazyk, ktorý formuluje požiadavky na odvodenie toho, čo je v databáze implicitné. Postavíme si otázku o hraniciach tohto prístupu: aké vety možno a aké nemožno odvodiť, ak používame zvolený jazyk a odvodzovacie prostriedky. Budeme hľadať možnosti, ako rozširovať predstavu o databáze (o vetách, ktoré sú v nej priamo reprezentované, o odvodzovacích schopnostiach). Venovať sa budeme databázam s neúplnou informáciou. Napokon prejdeme k dedukcii z viet databázy. Tým budeme postupne komplikovať našu predstavu o reprezentácii znalostí.

Táto kapitola môže nudiť čitateľa, ktorý pozná databázovú teóriu (vrátane teórie deduktívnych databáz). Môže ju preto obskočiť, ak si dobre uvedomuje, že:

- v relačnej databáze sú explicitne reprezentované nejaké znalosti,
- implicitne reprezentované sú v relačnej databáze ďalšie znalosti: môžeme ich dostať ako odpovede na otázky sformulované v nejakom databázovom jazyku,
- výpočtový mechanizmus, odpovedajúci na otázky, adresované relačnej databáze, je vlastne nejaký odvodzovací stroj a zodpovedá mu nejaký operátor odvodenia (ktorý môžeme špecifikovať sémanticky),
- to, čo môžeme vyjadriť v reči tabuliek, riadkov a stĺpcov (veľmi intuitívnej), môžeme vyjadriť v jazyku logiky,

- rozšírením relačného databázového jazyka a zodpovedajúceho operátora odvodenia môžeme reprezentovať znalosti, nevyjadriteľné v relačnej databáze,

- deduktívne databázy alebo databázy s neúplnou informáciou sú príkladmi takýchto rozšírení.

Na druhej strane, zaradenie tejto kapitoly do textu nie je samoúčelné. Text je venovaný oblasti, ktorej súčasnú tvár výrazne ovplyvnil konvergujúci výskum v oblasti deduktívnych databáz, logického programovania a umelej inteligencie.

Čitateľ, ktorý nie je oboznámený s takými sémantickými pojmami ako interpretácia a vyplývanie, by mal touto kapitolou prejsť, získa príklady a intuície. Dôležité bude intuitívne pochopenie sémantickej špecifikácie operátora odvodenia. Pravdepodobne aj z iných dôvodov bude užitočné zvládnuť túto kapitolu: Dúfam, že pripraví netrénovaného čitateľa na manipulácie s reláciami a pravidlami. Priblíži mu veľmi jednoduchý operátor odvodenia, pracujúci nad relačnou databázou. Ukáže, ako možno zosilňovať reprezentačný formalizmus (jazyk a operátor odvodenia) a vzájomne porovnávať silu rôznych formalizmov. Ukáže, že existujú hranice vyjadriteľnosti (v rámci nejakého reprezentačného formalizmu).

Všetky časti tejto kapitoly (okrem poslednej, 2.4) nepredpokladajú nijaké vedomosti o databázach. Ak by čitateľ privítal doplňujúce znalosti o logike, relačných a deduktívnych databázach, môže nahliadnuť do dodatkov B, C, D.

2.1 Štandardná databáza

Na začiatok si postavme jednoduchú otázku: aké znalosti sú reprezentované v (obyčajnej) databáze?

Databázu stotožníme s množinou nejakých viet a operácií na týchto vetách. Takýto pohľad na databázu priamo korešponduje s našou dohodou o chápaní reprezentácie znalostí. Navyše, korešponduje s bežným programátorským prístupom k databázam, aj so známymi databázovými modelmi.

Intuitívne, databázové vety štandardnej relačnej databázy môžeme chápať ako riadky v nejakých tabuľkách.

Príklad 2.1 Kvôli spríjemneniu ďalšieho výkladu si zavedieme jednoduchučkú databázu, pozostávajúcu z troch tabuliek. Prvá z nich registruje dvojice osôb, rodiča a dieťa. Ďalšie dve obsahujú množiny mužov a žien. Možno treba čitateľa upozorniť, že každé z použitých mien, napríklad *anna*, je menom jedinej osoby.

RODIČ	*DIEŤA*
anna	miro
anna	zuza
anna	štefan
karol	miro
karol	zuza
milan	štefan
zuza	eva
ondrej	eva

MUŽ
miro
štefan
karol
milan
ondrej

ŽENA
anna
zuza
eva

Každý riadok, napríklad druhý v prvej tabuľke, obsahuje nejakú – i keď až neslušne jednoduchú – znalosť. V uvedenom príklade je to fakt, že Anna je rodičom Zuzy. ■

Každá databáza obsahuje okrem explicitne zaznamenaných znalostí (riadkov, priamo obsiahnutých v nejakej tabuľke), aj implicitné znalosti. Tie si dokážeme z explicitne reprezentovaného obsahu odvodiť. Napríklad, z informácií o rodičovskom vzťahu si môžeme odvodiť informáciu o vzťahu byť starým rodičom. Takáto informácia nie je v tabuľke z príkladu 2.1 priamo (explicitne) obsiahnutá, je tam však implicitná, dá sa z uvedenej tabuľky vypočítať, odvodiť. Databázové systémy poskytujú vyjadrovacie prostriedky (databázové jazyky),[1] pomocou ktorých môžeme formulovať požiadavky na získavanie (odvodzovanie) implicitných znalostí.

Príklad 2.2 V tomto príklade si ukážeme dve otázky adresované databáze uvedenej v príklade 2.1. Jazyk, v ktorom ich formulujeme, je veľmi sugestívny a ľahko sa dá naučiť – Query by Example (QBE).

RODIČ	*DIEŤA*
$\_Z$	$P.\_Y$
$P.\_X$	$\_Z$

Otázka[2] z tejto tabuľky sa pýta na všetky dvojice *(starý rodič – vnúča)*. Odpoveďou na túto otázku v našom príklade je množina dvojíc

$$\{(anna, eva), (karol, eva)\}.$$

[1] V žargóne sa im hovorí dotazovacie jazyky, tomuto termínu sa však budeme vyhýbať. Podobne, nebudeme používať výraz „dotaz“, ale „otázka“.

[2] Väčšina čitateľov si pravdepodobne rozlúštila zásady, podľa ktorých sa uvedená otázka formulovala. Kvôli úplnosti ich však uvediem: Najprv sme špecifikovali pomocou premenných $\_Z, \_Y$ ľubovoľný riadok tabuľky naplnenej údajmi. Pri premennej $\_Y$ sme použili príkaz $P.$ (print), ktorý znamená, že na hodnoty tejto premennej sme zvedaví pri formulácii odpovede. Potom sme špecifikovali ďalší riadok a požiadavku na zobrazenie hodnoty premennej $\_X$. Dôležitý je vzťah oboch podmienok (riadkov) v našej formulácii otázky: raz je premenná $\_Z$ použitá ako predstaviteľka hodnoty stĺpca $RODIČ$, druhýkrát ako predstaviteľka hodnoty stĺpca $DIEŤA$. To znamená, že tá istá hodnota sa má v databáze vyskytovať raz v roli dieťaťa, raz v roli rodiča.

Môžeme si to preložiť do „reči reprezentácie znalostí“. Odpoveďou na našu otázku je poznatok o tom, kto je v našej databáze starým rodičom (a kto jeho vnúčaťom). V ľubovoľnom stave databázy odpoveď na túto otázku reprezentuje znalosti o vzťahu starého rodičovstva, zodpovedajúce momentálnym znalostiam o vzťahu rodičovstva. ■

Príklad 2.3 Uvedieme ešte jednu, trochu komplikovanejšiu, otázku (sformulovanú vo viacerých šablónach tabuliek).

RODIČ	*DIEŤA*
_Z	eva
_X	_Z

MUŽ
_Z

ŽENA
P._X

Táto otázka sa pýta na Evinu starú matku – po otcovej línii. Konštanty a premenné sa odlišujú iba tým, že pred premennou je podčiarovník „_“.[3] ■

To, čo sme si intuitívne vysvetlili pomocou tabuliek a jazyka QBE, preložíme do formálneho jazyka. Jazyk, v ktorom budeme otázky formulovať, sa nazýva *relačný kalkul*.

Príklad 2.4 Od veľmi intuitívnej a názornej tabuľkovej formulácie otázok prejdeme k formulácii tých istých otázok v jazyku logiky. Jazyk logiky sa často považuje za veľmi neprirodzený a ťažkopádny. Všimneme si však do očí bijúcu štrukturálnu podobnosť oboch formulácií. Naša otázka bude:

$$\{(X, Y) \mid \exists Z(rodi\check{c}(X, Z) \wedge rodi\check{c}(Z, Y))\}.$$

Pýtame sa na množinu dvojíc (X, Y), pre ktoré existuje Z, ktoré je dieťaťom X-a a súčasne rodičom Y-a.

Podobne môžeme vyjadriť otázku, odpoveďou na ktorú by malo byť meno Evinej starej mamy – po otcovej línii:

$$\{X \mid \exists Z(rodi\check{c}(Z, eva) \wedge mu\check{z}(Z) \wedge rodi\check{c}(X, Z) \wedge \check{z}ena(X))\}.$$

■

Porovnanie tabuľkovej formulácie (QBE) a logickej formulácie možno zhrnúť takto:

- vzájomne si zodpovedajú riadky v tabuľkovej formulácii a konjunkty logickej formulácie,

[3] Názov jazyka QBE pochádza z toho, že miesto „umelých“ premenných (aké sme my volili v tomto príklade) sa používateľovi zastrašenému matematikou odporúčalo používať nejaké príklady objektov z danej oblasti (domény) ako premenné (ako reprezentantov objektov tejto domény). Keby sme sa držali tohto zvyku, mohli by sme používať premenné ako $\_eva$, $\_otec$ a pod.

- odpoveď na otázku je určená hodnotami, ktoré má odpoveď obsahovať: v tabuľke sa tieto hodnoty označujú príkazom *P.*, v relačnom kalkule zoznamom voľných premenných.

Relačný kalkul, databázový jazyk nad relačnými databázami, si nebudeme detailne definovať. Uvedený príklad možno považovať za dostatočne zrejmý pre vybudovanie intuitívnej predstavy o tomto jazyku. Zhruba možno povedať, že relačný kalkul je jazyk, v ktorom špecifikujeme nejaké tabuľky. Špecifikujeme ich zoznamom voľných premenných (určuje počet stĺpcov tabuľky) a logickou formulou (určuje podmienku, ktorú musia spĺňať riadky tabuľky). V príklade 2.4 sme dvoma výrazmi relačného kalkulu špecifikovali dve tabuľky, jednu dvojstĺpcovú, druhú jednostĺpcovú.

Videli sme, že otázky v jazyku logiky sú štrukturálne zhodné s otázkami v názornej tabuľkovej forme (a teda máme argument v prospech ich intuitívnosti). Používanie jazyka logiky ako databázového jazyka má jednu ďalšiu (a veľkú) prednosť. Je ňou presne definovaná sémantika.

Príklad 2.5 Všimneme si príklad databázového jazyka, ktorému chýba jasná sémantická špecifikácia. Ide o jazyk systému Paradox. Minimálne v jeho 4. verzii funguje v dvoch rôznych kontextoch odlišne konštrukcia s tou istou sémantikou (pritom tvorcovia jazyka vychádzajú z QBE). Všimnime si nasledujúcu otázku:

TOVARY	Číslo	Cena
		<100000, $\_X$, changeto $\_X * 1.07$ $>= 100000$, $\_Y$, changeto $\_Y * 1.12$

Uvedená formulácia špecifikuje zmenu („changeto“) cien tovarov: Tovary s cenou nižšou ako 100 tisíc budú zdražené o 7%, ostatné tovary o 12%.

Celkom prirodzenou otázkou je, ako budú vyzerať nové ceny. Formulácia takejto otázky je zhodná s uvedenou formuláciou modifikácie, až na to, že „changeto“ sa zamení za „calc“ (vypočítaj!). Môžeme požadovať aj tlač čísla tovaru. Odpoveďou by mali byť očakávané zvýšené ceny tovarov. Keď sa však pokúsite túto otázku sformulovať, prichádza správa o chybe: „Query appears to ask two unrelated questions“.

To znamená, že implementovaný jazyk je závislý na kontexte. V jednom kontexte sa otázka vyhodnotí, v inom nie. ■

Jazyk s dobre definovanou sémantikou a korektná implementácia tejto sémantiky majú tú výhodu, že ten, kto rozumie definovanej sémantike, by nemal mať problémy s formuláciou otázok a ďalších konštruktov jazyka. Vie spoľahlivo špecifikovať ich efekt. Naopak, v jazyku bez týchto čŕt je často potrebné vyskúšať jazykové konštrukcie a zo skúsenosti v rôznych kontextoch sa postupne učiť, aký je ich efekt.

Doteraz sme sa usilovali o ilustráciu toho, ako je dôležité, že logika – resp. jej fragment – môže slúžiť ako databázový jazyk a môže mu poskytnúť svoju sémantiku.

V jazyku logiky však môžeme popri otázkach vyjadriť aj databázové vety. Vetami databázy sú základné atómy jazyka predikátovej logiky bez funkčných symbolov.[4] Základný atóm (bez funkčných symbolov) v jazyku predikátovej logiky je formula bez logických spojok, pozostávajúca z predikátového symbolu a zoznamu konštánt, pričom počet konštánt sa rovná arite predikátového symbolu. Takéto formuly možno nájsť v príklade 2.6.

Databázu z príkladu 2.1 teraz možno vidieť takto (zdá sa, že prostý pohľad na oba príklady a vhodná skladačka by umožnili malým deťom vzájomne transformovať obe formulácie):

Príklad 2.6 Databáza:

$rodič(anna, miro)$	$muž(miro)$	$žena(anna)$
$rodič(anna, zuza)$	$muž(štefan)$	$žena(zuza)$
$rodič(anna, štefan)$	$muž(karol)$	$žena(eva)$
$rodič(karol, miro)$	$muž(milan)$	
$rodič(karol, zuza)$	$muž(ondrej)$	
$rodič(milan, štefan)$		
$rodič(zuza, eva)$		
$rodič(ondrej, eva)$		

■

Predbežne zhrnieme, že v štandardnej databáze je reprezentáciou znalostí štvorica (J, E, Cn, Ind), kde E je nejaká množina základných atómov (prvorádového logického jazyka J), Cn si môžeme predstaviť ako stroj, ktorý akceptuje otázky napísané v databázovom jazyku (napríklad v relačnom kalkule) a produkuje odpovede.

Ind nech je množina obvyklých databázových indexov. Predpokladajme tabuľku s n stĺpcami. Potom index na stĺpci i je (čiastočná) funkcia f, ktorá (každej v tomto stĺpci použitej) hodnote h priradí zoznam viet, v ktorých sa h vyskytuje ako hodnota v stĺpci i. Podobne možno definovať index pre k-tice stĺpcov.

Príklad 2.7 Pripomeňme si tabuľku $RODIČ$ z príkladu 2.1. Index definovaný na jej prvom stĺpci priradí hodnote $anna$ zoznam $1, 2, 3$: práve v prvých troch riadkoch sa vyskytuje v prvom stĺpci hodnota $anna$. Hodnota indexu definovaného na oboch stĺpcoch priraďuje dvojici $(zuza, eva)$ hodnotu 6.

Čitateľ si iste ľahko vyjadrí uvedené indexy tak, aby fungovali na formulách (na logických reprezentáciách), napríklad na základných atómoch z príkladu 2.6. Stačí hovoriť o argumentoch základných atómov namiesto stĺpcov. ■

[4] ⟨ predikátová logika: dodatok B ⟩

Z hľadiska našej témy je kľúčová otázka, čo sa dá vyjadriť v takto poňatej databáze. Istotne všetky základné atómy priamo v nej zaznamenané (explicitné znalosti). Implicitné znalosti získame ako odpovede na otázky adresované databáze. Tieto odpovede môžeme – z logického hľadiska – chápať ako množiny základných atómov.[5]

Samozrejme, kľúčovým problémom je určiť, aké znalosti môžeme získať ako odpovede na otázky formulované v relačnom kalkule, prípadne určiť, aké znalosti nemožno dostať ako odpovede na otázku v relačnom kalkule. Tu sa na tieto problémy pozrieme ako na sémantické problémy.

Doposiaľ sme venovali pozornosť hlavne syntaxi - tvaru jazyka, v ktorom sú reprezentované vety v databáze a tvaru jazyka, v ktorom formulujeme otázky. Odpovede na otázky sme „videli, cítili" intuitívne. Príklad 2.5 nám ukázal, že čisto intuitívne postupy nie sú celkom spoľahlivé. Prirodzene vzniká otázka, ako špecifikovať Cn pre relačný kalkul. Odpoveď znie jednoducho: definovaním jeho sémantiky. Za sémantiku relačného kalkulu[6] budeme považovať relačnú algebru.

Vráťme sa od jazyka logiky k tabuľkám. Na tabuľkách môžeme definovať operácie relačnej algebry.

Aby sme pokročili pri charakterizovaní sémantiky, prijmeme dohodu, ktoré druhy viet môžu byť v E (akého tvaru bude databáza, aká bude jej schéma).

Definícia 2.8 *Databázová schéma* je zoznam predikátových symbolov, spolu s určením ich arity. Pre každý argument zavedieme jeho názov a typ. ■

Kvôli jednoduchosti však budeme od typov zväčša abstrahovať.

Príklad 2.9 Abstraktným príkladom databázovej schémy môže byť

$$\{p_1(A_{1,1} : T_{1,1}, \ldots, A_{1,k_1} : T_{1,k_1}), \ldots, p_m(A_{m,1} : T_{m,1}, \ldots, A_{m,k_m} : T_{m,k_m})\},$$

kde p_i sú predikátové symboly, k_i je arita predikátového symbolu p_i, $A_{j,l}$ sú atribúty (názvy argumentov) a $T_{j,l}$ sú typy (hodnôt atribútov). Z logického hľadiska, atribúty možno chápať ako premenné, pretože substitúciou konštánt zodpovedajúceho typu za ne dostaneme jednotlivé databázové vety.

[5] Totižto, ku každej otázke v relačnom kalkule môžeme zaviesť nový predikátový symbol, ktorého arita zodpovedá počtu voľných premenných danej otázky. Všetky k-tice, ktoré sú odpoveďami na danú otázku, možno považovať za k-tice argumentov zavedeného predikátového symbolu. Je teda zrejmé, že odpoveď na otázky možno stotožniť s nejakou množinou základných atómov. Korešpondencia základných atómov a odpovedí na otázky bude zrejmejšia onedlho, keď zavedieme pravidlá v časti 2.3.

[6] Takto postavený problém nezodpovedá celkom histórii, ani vzájomnej ekvivalencii relačného kalkulu a relačnej algebry – ide o dva rovnocenné databázové jazyky. Na druhej strane, uvedená konštrukcia bude sémantická v obvyklom zmysle slova. Ďalej, na tomto jednoduchom príklade si pripravíme pôdu pre ďalší výklad, v ktorom budeme dôraz klásť na sémantickú špecifikáciu rôznych Cn. Prístup použitý tu (sémantika je definovaná pevne zvoleným vzorom, mierkou – relačnou algebrou) je jeden z možných pri sémantickej špecifikácii Cn.

Konkrétnejší príklad:

$$\{zamestnanec(Id : N, Meno : S, Plat : N),\\ oddelenie(Odd\# : N, Názov : S, Vedúci : N)\},$$

kde arita oboch predikátov je 3, $Id, Meno, Plat, Odd\#, Názov, Vedúci$ sú atribúty a N, S sú typy (N je množina prirodzených čísel, S je množina reťazcov nad nejakou abecedou).[7] ■

Predstavme si tabuľku s k stĺpcami, v ktorých sa vyskytujú hodnoty typov $T_1, \ldots, T_k$ (vždy v i-tom stĺpci hodnoty typu T_i). Uvažujme kartézsky súčin $T_1 \times \cdots \times T_k$ a konečnú reláciu ρ, jeho (konečnú) podmnožinu.[8] Budeme hovoriť, že k je *arita* tejto relácie. Vidíme, že každej tabuľke s danou databázovou schémou zodpovedá jediná relácia a naopak, každej relácii jediná tabuľka. Teraz je zrejmý pôvod termínu relačné databázy, ktoré tu označujeme ako „štandardné databázy“ .

Štandardnú databázu môžeme stotožniť s interpretáciou databázovej schémy.[9] Tu sa uspokojíme s intuitívnym výkladom a urobíme iba prvý krok, ktorým si pripravíme pôdu pre pochopenie interpretácie predikátových symbolov. Predikátové symboly z databázovej schémy budeme interpretovať tabuľkami (reláciami). Počet stĺpcov v tabuľke musí zodpovedať arite predikátového symbolu. Typ hodnôt v príslušnom stĺpci zodpovedá typu[10] argumentu z definície databázovej schémy.

Príklad 2.10 Majme binárny predikátový symbol *rodič*. Zmysluplné atomárne formuly majú tvar $rodič(X, Y)$. Tento predikátový symbol si môžeme interpretovať tabuľkou (s dvoma stĺpcami) $RODIČ$ z príkladu 2.1.

Samozrejme, spomínaný predikát možno interpretovať akoukoľvek inou tabuľkou, ktorá má dva stĺpce a hodnoty v oboch stĺpcoch sú typu $OSOBA$. ■

Predikátové symboly sme intepretovali tabuľkami. Získali sme tým postačujúcu intuíciu pre narábanie s pojmom interpretácie. Objasníme si, tiež na príklade, pojem modelu. Na to však potrebujeme zaviesť integritné obmedzenia. Nasledujúci príklad jednak poslúži ako ďalšia ilustrácia pojmu interpretácie, jednak poskytne ukážku integritného obmedzenia a na jeho základe dá predstavu o pojme modelu.

Príklad 2.11 Všimnime si teraz tabuľky $MUŽ$ a $ŽENA$ z príkladu 2.1. Predikáty *muž, žena* budeme interpretovať práve týmito tabuľkami.

[7] ⟨ reťazec: dodatok E ⟩

[8] ⟨ kartézsky súčin: dodatky C a E ⟩

[9] ⟨ interpretácia: dodatok B alebo kapitola 3 ⟩

[10] Z hľadiska jazyka logiky ide o viacsortový predikátový počet. V dodatku B je načrtnuté, ako viacsortovú logiku možno redukovať na jednosortovú. Vzhľadom na databázovú aplikáciu je prirodzené používať viacsortový jazyk, nie je to však nevyhnutné.

Ak máme predikáty *muž, žena* interpretované tabuľkami tak, že sa nájde objekt, ktorý je muž aj žena, nie je s našou databázou všetko v poriadku. Môžeme si povedať, že naša databáza je v rozumnom stave vtedy, keď neexistuje objekt, ktorý je súčasne mužom a ženou. Samozrejme, môžeme vyžadovať splnenie ďalších požiadaviek. Napríklad: ak je niekto rodičom (dieťaťom), je buď mužom alebo ženou, ďalej nik nemôže byť súčasne rodičom a dieťaťom toho istého objektu, nik nemôže byť rodičom alebo dieťaťom seba samého. ■

Predchádzajúci príklad ilustruje, čo rozumieme pod „databázou v rozumnom stave". Táto predstava zodpovedá pojmu modelu. Stále iba intuitívne: Model (nejakej množiny viet, teórie) je taká interpretácia, ktorá robí túto množinu viet (teóriu) pravdivou. V našom prípade: interpretácie predikátov *muž, žena* tabuľkami $MU\check{Z}$ a $\check{Z}ENA$ nebudú modelom databázovej schémy, ak obsahujú zhodný riadok.

Vety, formulujúce podmienky pre rozumný stav databázy nazývame *integritnými obmedzeniami.*

Máme teda inutitívnu predstavu o pojmoch modelu a integritného obmedzenia. Teraz rozšírime našu intuíciu o sémantickej špecifikácii Cn. Zatiaľ sme sa dohodli o tom, ako interpretovať predikáty. Dôležité (ale jednoduché) je interpretovať logické symboly. Naznačíme, ako sa to dá robiť pomocou operácií relačnej algebry.

Najprv si uvedieme základné operácie relačnej algebry:

zjednotenie ak ρ_1 a ρ_2 sú dve relácie, potom do ich zjednotenia $\rho_1 \cup \rho_2$ patria všetky prvky, ktoré patria buď do ρ_1 alebo do ρ_2,[11]

rozdiel ak ρ_1 a ρ_2 sú dve relácie, potom do ich rozdielu $\rho_1 \setminus \rho_2$ patria všetky prvky, ktoré patria do ρ_1, ale nepatria do ρ_2,

kartézsky súčin ak ρ_1 a ρ_2 sú dve relácie, potom do ich kartézskeho súčinu $\rho_1 \times \rho_2$ patria všetky možné $k+m$-ice (x, y), kde k-tica $x \in \rho_1$ a m-tica $y \in \rho_2$,

selekcia ak ρ je nejaká relácia a F formula nejakého jazyka, potom $\sigma_F(\rho)$ je relácia, ktorá obsahuje presne tie prvky z ρ, ktoré spĺňajú podmienku, vyjadrenú formulou F,

projekcia ak ρ je nejaká relácia arity k a $\xi = (i, \ldots, j)$ je nejaký zoznam čísel, kde $1 \leq i \leq j \leq k$, potom $\pi_\xi(\rho)$ je relácia, obsahujúca presne tie prvky, ktoré získame z k-tic patriacich do ρ tak, že zabudneme na tie komponenty, ktoré nepatria do ξ; arita $\pi_\xi(\rho)$ je určená počtom prvkov v zozname ξ.

[11] Kvôli databázovej motivácii nás budú zaujímať iba zjednotenia relácií rovnakej arity: nedáva dobrý databázový zmysel zjednocovať tabuľky s rôznym počtom stĺpcov. To isté obmedzenie platí o operácii rozdielu.

Nasledujúci príklad ukazuje, ako možno zadefinovať reláciu *starý_rodič* (príklad 2.2) pomocou operácií relačnej algebry:

Príklad 2.12 Urobíme kartézsky súčin $RODI\check{C} \times RODI\check{C}$ (jeho arita je 4). Selekciou vyberieme iba tie prvky, ktorých druhý komponent (nejaké dieťa) sa rovná tretiemu komponentu (nejakému rodičovi v tej istej relácii). Nakoniec urobíme projekciu na prvý komponent a štvrtý komponent (dostaneme tak množinu dvojíc, ktorej prvým členom je nejaký rodič, ktorý má dieťa, ktoré je rodičom druhého komponentu výslednej dvojice):

$$\pi_{\{1,4\}}(\sigma_{\$2=\$3}(RODI\check{C} \times RODI\check{C})).$$

■

Teraz prejdeme k interpretácii formúl relačného kalkulu.

Budeme používať túto terminologickú skratku: ak p, q sú predikátové symboly arity k resp. m, potom formuly $p(X_1, \ldots, X_k)$ a $q(X_1, \ldots, X_m)$ nazývame predikátmi. Nech $\diamond$ je ľubovoľný unárny logický symbol a $\circ$ je ľubovoľný binárny logický symbol. Potom $\diamond\, p(X_1, \ldots, X_k)$ a

$$p(X_1, \ldots, X_k) \circ q(X_1, \ldots, X_m)$$

nazývame tiež predikátmi.

Najprv zadefinujeme operáciu prirodzeného spojenia. Predstavuje užitočný prostriedok spracúvania relácií.

Definícia 2.13 Nech sú dané predikáty

$$r(A_1 : T_1, \ldots, A_k : T_k, B_1 : U_1, \ldots, B_m : U_m),$$
$$s(A_1 : T_1, \ldots, A_k : T_k, C_1 : V_1, \ldots, C_n : V_n),$$

kde T_i, U_i, V_i sú typy, A_i, B_i, C_i sú premenné (atribúty). Nech R a S sú relácie, ktorými interpretujeme uvedené predikáty.

Pritodzeným spojením R a S, značenie $R \bowtie S$, je relácia, ktorú dostaneme vyhodnotením nasledujúceho výrazu relačnej algebry:

$$\pi_{A_1,\ldots,A_k,B_1,\ldots,B_m,C_1,\ldots,C_n}(\sigma_{R.A_1=S.A_1 \wedge \cdots \wedge R.A_k=S.A_k}(R \times S))),$$

kde $R.A_i$ $(S.A_i)$ je stĺpec v relácii R (S), ktorý zodpovedá atribútu A_i. ■

Teda – prirodzené spojenie počítame tak, že skonštruujeme kartézsky súčin, vyselektujeme z neho iba tie riadky, v ktorých majú obe relácie rovnaké hodnoty spoločných atribútov a nakoniec projekciou eliminujeme opakovaný výskyt rovnakých atribútov. Doposiaľ sme projekciu robili podľa poradových čísel stĺpcov. Čitateľ iste nemá problémy s jej špecifikovaním podľa atribútov.

Definícia 2.14 Predpokladajme, že predikát p arity k je interpretovaný tabuľkou U_1 a predikát q arity m tabuľkou U_2.

- predikát $p(X_{1,1}, \ldots, X_{1,k}) \vee q(X_{2,1}, \ldots, X_{2,m})$ bude interpretovaný tabuľkou $U_1 \cup U_2$,[12]
- predikát $p(X_{1,1}, \ldots, X_{1,k}) \wedge q(X_{2,1}, \ldots, X_{2,m})$ bude interpretovaný tabuľkou $U_1 \bowtie U_2$, pričom sa predpokladá nejaká množina spoločných premenných v oboch konjunktoch (ak je táto množina prázdna, prirodzené spojenie degeneruje v kartézsky súčin, ak sa premenné úplne zhodujú, degeneruje v prienik),[13]
- predikát $\neg p(X_1, \ldots, X_k)$ bude interpretovaný tabuľkou $(T_1 \times \cdots \times T_k) \setminus U_1$, kde T_i je typ hodnôt premennej X_i,[14]
- predikát $\exists X_i p(X_1, \ldots, X_k)$ tabuľkou $\pi_{1,\ldots,i-1,i+1,\ldots,k} U_1$.

■

Teraz sa vráťme k Cn. V kapitole 1 sme sa dohodli, že budeme preferovať sémantickú špecifikáciu Cn. Tu sme si zvolili prístup definovať sémantiku pomocou vybratého vzoru, relačnej algebry. Takto špecifikovaný operátor si môžeme označiť ako Cn_{RA}.

Diskutujme teda rozhodnutie, podľa ktorého relačná algebra poskytuje sémantickú špecifikáciu Cn_{RA}.

Majme nejakú otázku Φ, napísanú v relačnom kalkule. Predikátové symboly sú interpretované nejakými tabuľkami. Nech niektorá atomárna podformula ϕ z Φ obsahuje jedinú konštantu a (ľahko sa zovšeobecní na prípad viacerých konštánt) ako i-ty argument a predikát z ϕ je interpretovaný reláciou ρ arity k. Dohodnime sa na značení: $\sigma_{\$i=a}(\rho)$ je relácia, ktorá z ρ vyberá tie k-tice, v ktorých i-ty komponent je a.

Potom podformula ϕ je interpretovaná práve reláciou $\sigma_{\$i=a}(\rho)$. Po interpretácii všetkých atomárnych podformúl môžeme použiť operácie relačnej algebry, aby sme interpretovali celú formulu Φ podľa postupu z definíície 2.14.

Načrtli sme, ako možno interpretovať predikátové symboly tabuľkami a výrazy logiky operáciami relačnej algebry. Máme teda návod, ako rozšíriť interpretáciu predikátových symbolov na interpretáciu otázok v relačnom kalkule. Každá takáto interpretácia dáva korektnú odpoveď na danú otázku. Nasledujúci príklad to ilustruje.

Príklad 2.15 Nasledujúca otázka je z príkladu 2.4:

$$\{X \mid \exists Z(rodi\check{c}(Z, eva) \wedge mu\check{z}(Z) \wedge rodi\check{c}(X, Z) \wedge \check{z}ena(X))\}.$$

---

[12] Pripomíname opäť, že $k = m$, aby operácia mala jasný databázový zmysel.

[13] ⟨ prienik: dodatok E ⟩

[14] Pozorný čitateľ vidí, že v prípade kartézskeho súčinu $T_1 \times \cdots \times T_k$ s nekonečnou mohutnosťou a konečnej tabuľky U_1 dostávame nekonečný rozdiel, čím sme sa dostali do rozporu s databázovou intuíciou, ktorá vyžaduje konečnosť tabuliek. Dôkladná teória relačných databáz na tento problém myslí, odkazujeme tu na [Ull 88].

Nech pomocou (konkrétnych) tabuliek $RODI\check{C}$, $MU\check{Z}$, $\check{Z}ENA$ interpretujeme predikáty *rodič, muž, žena*. Odpoveď na túto otázku môžeme získať vyhodnotením nasledujúceho výrazu relačnej algebry

$$\pi_{\$1}(\sigma_{\$1=\$3=\$5}((\sigma_{\$2}=evaRODI\check{C}\times MU\check{Z})\times\sigma_{\$1=\$3}(RODI\check{C}\times\check{Z}ENA))).$$

Teda, z kartézskeho súčinu $RODI\check{C}\times MU\check{Z}$ vyselektujeme iba tie riadky, v ktorých eva je dieťaťom. Z kartézskeho súčinu $RODI\check{C}\times\check{Z}ENA$ vyselektujeme tie riadky, v ktorých sa hodnoty 1. a 3. stĺpca zhodujú. Urobíme kartézsky súčin oboch uvedených kartézskych súčinov, vyselektujeme z nich iba tie riadky, kde sa hodnoty 1., 3. a 5. stĺpca zhodujú. Napokon, projekciou vyberieme iba prvý stĺpec. ■

Základné operácie relačnej algebry sú dobre definované matematické pojmy (z tohto dôvodu by sme v súlade s [Llo 87] mohli povedať, že relačná algebra poskytuje *deklaratívnu* sémantiku pre Cn_{RA}). Navyše, existujú efektívne algoritmy, na základe ktorých možno počítať operácie (vyhodnotiť výrazy)[15] relačnej algebry (*operačná* sémantika podľa [Llo 87]).

Treba zdôrazniť, že operácie relačnej algebry, aplikované na relácie, produkujú znovu relácie, na ktoré sú aplikovateľné ďalšie operácie (atď.). Táto operačná homogénnosť je inšpiratívna aj z hľadiska reprezentácie znalostí. Všetky relácie predstavujú nejakú množinu znalostí - bez ohľadu na to, či sú zapamätané explicitne, alebo či sú vypočítané pomocou relačných operácií.

Špecifikovali sme (pomocou nejakého vzoru) databázovú inferenčnú mašinu Cn_{RA} pomocou relačnej algebry. Celkom prirodzená otázka je, či v relačnom kalkule možno vyjadriť každú otázku, ktorej sémantickú špecifikáciu poskytuje relačná algebra a či každá otázka v relačnom kalkule má „sémantiku" definovanú v relačnej algebre. Odpoveď je pozitívna. Relačný kalkul a relačná algebra sú ekvivalentné (pri istých spresneniach, obmedzeniach, potrebu ktorých sme už naznačili). Podrobný (a relatívne zdĺhavý) výklad je však mimo našej pozornosti. Možno ho nájsť v [Ull 88].

Dôležité je však uvedomiť si jadro problému: Špecifikovali sme (sémantickými prostriedkami) správanie Cn_{RA}. Cn_{RA} považujeme za odvodzovací stroj, ktorý dokáže odpovedať na otázky relačného kalkulu. Ekvivalencia relačnej algebry a relačného kalkulu ukazuje, že je možná korektná implementácia Cn_{RA}.

Na záver by nás mohlo zaujímať, či táto špecifikácia je adekvátna. Odpoveď závisí od našich ambícií. Môžeme sa však bez výčitiek svedomia pýtať, čo nie je vyjadriteľné (reprezentovateľné) v štandardnej databáze.

Príklad 2.16 V štandardnej relačnej databáze nie je vyjadriteľná neúplná informácia.

[15] Ak ϕ je výraz relačnej algebry, ktorý obsahuje mená relácií $\rho_1,\ldots,\rho_k$, potom vyhodnotením tohto výrazu je relácia, vypočítaná podľa pravidiel z definície 2.14.

Databázové vety sme stotožnili so základnými atómami jazyka predikátovej logiky. V reálnych databázach niektoré konštanty (hodnoty niektorých stĺpcov v nejakých riadkoch) nemusia byť známe. Napríklad pri schéme

$$zamestnanec(ID\#, Meno, Adresa, Plat)$$

nemusíme poznať adresu každého zamestnanca. Aj v našej formalizácii potrebujeme poňatie databázovej vety trochu zliberalizovať tak, aby sme nemuseli používať iba základné atómy (nebudeme mať napríklad vždy k dispozícii konštantu, ktorá je hodnotou atribútu *Adresa*). ■

Ďalším príkladom otázok nevyjadriteľných pomocou Cn_{RA} sú otázky o dôveryhodnosti databázových viet.

Príklad 2.17 V štandardnej relačnej databáze nie sme schopní vyjadriť mieru našej dôvery v databázové vety. Každú vetu v databáze sme nútení považovať za rovnako pravdivú (prípadne môžeme pripustiť, že nie je spoľahnutie na niektoré z nich, ale – keďže nespoľahlivosť nemáme nijako zaznamenanú – naša skepsa sa vlastne týka všetkých viet z databázy). ■

Príklad 2.18 V relačnej algebre nemožno konečným výrazom vyjadriť tranzitívny uzáver binárnej relácie.[16] Výrazy relačnej algebry sú iba konečné. Pre náš príklad 2.6 to znamená, že nemôžeme konečným výrazom relačnej algebry vyjadriť otázku o všetkých predkoch (alebo potomkoch) nejakej osoby. Teda, existujú nejaké znalosti, ktoré sú implicitne obsiahnuté v štandardnej databáze, ale nedokážeme ich vyjadriť (reprezentovať), kým Cn (a s ňou aj reprezentáciu implicitných znalostí) špecifikujeme pomocou relačnej algebry. ■

Teda, ak by medzi naše ambície patrilo reprezentovať znalosti o tranzitívnom uzávere binárnej relácie, špecifikácia inferenčného stroja pomocou relačnej algebry by nebola adekvátna (podobne pre prípad neúplnej a nie celkom spoľahlivej informácie). Ak máme väčšie ambície, potrebujeme zosilniť vyjadrovacie schopnosti jazyka J a/alebo odvodzovacie schopnosti Cn.

Pojem implicitnej obsiahnutosti (implicitnej znalosti) používame v rôznych zmysloch.

Ak je implicitnou znalosťou relácia, ktorú nenájdeme priamo v databáze, ale získame ju vyhodnotením nejakého výrazu relačnej algebry, potom relačný kalkul je úplný. Dokáže totiž vyjadriť každú implicitnú informáciu.

Pojem otázky možno však rozumným spôsobom zovšeobecniť. Otázka sa dá chápať ako vypočítateľné zobrazenie z množiny databáz danej schémy do ľubovoľnej relácie definovateľnej nad typmi danej schémy. Myšlienka, ktorá je za tým: pýtame sa na relácie, ktoré sa dajú vypočítateľným spôsobom definovať nad danou databázou.

[16] ⟨ tranzitívny uzáver: dodatok E ⟩

Definícia 2.19 Majme databázovú schému $S = \{$

$$\begin{aligned} p_1(A_{1,1} : T_{1,1}, \ldots, A_{1,k_1} : T_{1,k_1}), \\ \ldots, \\ p_m(A_{m,1} : T_{m,1}, \ldots, A_{m,k_m} : T_{m,k_m})\}. \end{aligned}$$

Nech pre každé i je predikátový symbol p_i interpretovaný reláciou ρ_i. Nech Ω je množina relácií, interpretácií predikátových symbolov schémy S. Nech ďalej Σ je množina relácií, podmnožín ľubovoľného kartézskeho súčinu $T_{i,j} \times \cdots \times T_{m,k}$, kde komponentami súčinu sú niektoré typy zo schémy S.

Potom *otázka* Q je čiastočne rekurzívna funkcia z Ω do Σ. ■

Podrobnejšie pozri v [ChH 85]. V časti 5.1.1 si ukážeme, že pojem otázky môžeme zovšeobecniť ešte viac.

Predstava o implicitnej znalosti, ktorá nie je odpoveďou na žiadnu otázku relačného kalkulu, dostáva teraz jasné kontúry. Existuje totiž otázka (o tranzitívnom uzávere), ktorá je otázkou podľa uvedenej definície (do Σ patrí aj podmnožina kartézskeho súčinu $Osoba \times Osoba$, ktorá pre danú databázu obsahuje dvojice späté vzťahom „byť predkom“), pričom v relačnej algebre ju nemožno vyjadriť (a teda ani vyhodnotiť). V tomto zmysle slova je relačný kalkul neúplný.

Ak by sme chceli k uvedenému všeobecnému pojmu otázky skonštruovať zodpovedajúci operátor odvodenia (s predstavou, že výsledkom odvodenia budú všetky možné odpovede na takto definované otázky), museli by sme postupovať inak ako pri špecifikácii Cn_{RA}. (Pozri sekciu 2.3 alebo [ChH 85].) O implicitne reprezentovaných znalostiach teda má zmysel hovoriť iba vzhľadom na nejaký operátor odvodenia a reprezentačný jazyk.

Vyššie uvedená zbežná úvaha naznačuje, že možno definovať nejaké usporiadania na špecifikáciách operátorov odvodenia (prípadne aj na jazykoch, kvôli jednoduchosti však budeme spomínať iba operátory odvodenia). Je zrejmé, že implementáciou slabšieho operátora neodvodíme niektoré vety, ktoré sú v databáze implicitné na základe špecifikácie silnejšieho operátora. Môžeme hovoriť, že takéto usporiadanie na operátoroch odvodenia indukuje usporiadanie na reprezentáciách znalostí.

Tu je vhodné poznamenať: nemožno očakávať od žiadneho jazyka na reprezentáciu znalostí, že v ňom možno vyjadriť všetko.[17] Existujú hranice vyjadriteľnosti (reprezentovateľnosti).

2.2 Neúplná a neistá informácia v databázach

Nasleduje zopár menej systematických poznámok na tému databáz s neúplnou informáciou a s neistou (nie celkom spoľahlivou) informáciou. Tieto poznámky

[17] Nedomnievam sa, že to možno očakávať od prirodzeného jazyka, ale táto poznámka už patrí do filozofie.

sú motivované príkladmi 2.16 a 2.17. Načrtneme možnosti zovšeobecnenia a zosilnenia vyjadrovacích schopností databázového jazyka.

Predpokladajme, že máme k dispozícii jazyk, zavedený v sekcii 2.1, menovite, máme definovaný pojem databázovej vety ako základného atómu jazyka predikátovej logiky. Teraz si pojem databázovej vety zovšeobecníme.

Definícia 2.20 Ak V je databázová veta, potom $\exists X V'$ je databázová veta, pričom V' získame z V tak, že nahradíme jednu konštantu premennou X. ■

Príklad 2.21 V príklade 2.16 sme sa stretli so schémou

$$zamestnanec(ID\#, Meno, Adresa, Plat)$$

a predpokladali sme, že nemusíme poznať adresu každého zamestnanca.

Keď nemáme k dispozícii hodnotu atribútu *Adresa*, môžeme za databázovú vetu považovať aj $\exists X\ zamestnanec(7208017676, juro, X, 5500)$. ■

Očividne, toto rozšírenie pojmu databázovej vety umožňuje formálnu reprezentáciu neúplnej informácie – nie je potrebné, aby v každej vete boli známe všetky argumenty (konštanty).[18]

Poznamenajme, že definíciu 2.20 možno aplikovať rekurzívne, takže databázová veta môže obsahovať viac neznámych hodnôt. Budeme ich nazývať *nulovými* hodnotami. Samozrejme, na používanie nulových hodnôt je primerané klásť nejaké obmedzenia. Databázová veta, pozostávajúca zo samých nulových hodnôt zrejme nemá nijaký zmysel. Ďalej, ak chceme zachovať databázové intuície, je prirodzené, keď zabránime tomu, aby nulové hodnoty nadobúdali niektoré z atribútov (napríklad kľúčové,[19] i ďalšie s dôležitým informačným významom).

Ak v databázovej vete $\exists X\ p(a_1, \ldots, a_{i-1}, X, a_{i+1}, \ldots, a_k)$ odstránime existenčný kvantifikátor a viazanú premennú nahradíme doposiaľ nepoužitou konštantou ω, dostaneme formulu tvaru $p(a_1, \ldots, a_{i-1}, \omega, a_{i+1}, \ldots, a_k)$. Pri každom ďalšom výskyte existenčného kvantifikátora sa zavedie vždy ďalšia nová konštanta. Táto procedúra sa nazýva *skolemizáciou* a ω *Skolemovou konštantou. Skolemovou normálnou formou* sa nazýva formula, ktorá je v prenexnej normálnej forme a v ktorej všetky existenčné kvantifikátory sú nahradené Skolemovými konštantami.[20] V databázovom svete sa nepoužíva termín Skolemove konštanty, zaviedol sa termín *nulové hodnoty.* Tie však plnia presne tú úlohu, čo Skolemove konštanty – nulová hodnota je neznáma hodnota. Nejaká existuje, ale nie je známe, aká.

[18] V tejto knihe sa používajú rôzne pojmy neúplnosti. Na týchto miestach rozumieme pod neúplnou informáciou iba a presne tie databázové vety, ktoré nie sú základnými atómami.

[19] Kľúčové atribúty jednoznačne identifikujú databázovú vetu: Množina atribútov $A_{i_1}, \ldots, A_{i_j}$ tvorí kľúč relačnej schémy $R(A_1, \ldots, A_k)$, ak je podmnožinou množiny $A_1, \ldots, A_k$ a v žiadnom modeli tejto schémy neexistujú dve rôzne k–tice, ktoré sa zhodujú v hodnotách atribútov $A_{i_1}, \ldots, A_{i_j}$.

[20] ⟨ dodatok D ⟩

Príklad 2.22 Databázová veta z príkladu 2.21 so Skolemovou konštantou (nulovou hodnotou): $zamestnanec(7208017676, juro, \omega, 5500)$. ■

Rozšírenie databázového jazyka tak, aby bol schopný vyjadriť neúplnú informáciu a zaznamenať ju v E, nespôsobuje nijaké problémy. Čo je podstatne zložitejšie, je adekvátna modifikácia odvodzovacieho operátora – označme ho Cn_{idb}. Cieľom a ohraničením pritom je, aby odvodzovanie v takomto jazyku (za prítomnosti neúplnej informácie) zodpovedalo našim intuíciam.

Nebudeme sa podrobne venovať detailom špecifikácie Cn_{idb}, tento problém nie je v centre našej pozornosti. Základné idey iba načrtneme. Najprv si na veľmi jednoduchej formalizácii ukážeme základný problém a spektrum jeho možných riešení, potom stručne zreferujeme, ako sa tento problém rieši v literatúre.

Predstavme si, ako by sme skomplikovali relačnú algebru, aby primerane špecifikovala Cn_{idb}. Zrejme by sme nemuseli nič meniť na kartézskom súčine: „násobili“ by sme prvky relácií aj s nulovými hodnotami tak, ako by to boli regulárne hodnoty. Podobne pri projekcii: jej hodnotou by boli navzájom rôzne „fragmenty“ pôvodných k-tic (aj s nulovými hodnotami). Azda jediným problémom sú k-tice, patriace do projekcie a pozostávajúce zo samých nulových hodnôt.

Podľa všetkého by sme museli modifikovať definíciu zjednotenia, rozdielu a selekcie. Nie je vylúčené, že užitočné by boli nejaké nové operácie, ktoré nemajú zmysel pri databáze s úplnou informáciou. Problémy so zjednotením a rozdielom obídeme a budeme sa venovať iba selekcii.

Zavedieme – postupne – definíciu usporiadania na prvkoch relácií. Najprv: Skolemovu konštantu budeme považovať za horšiu (vzhľadom na definované usporiadanie) ako obvyklú konštantu.

Definícia 2.23 Ak ω je Skolemova konštanta a t_i je hodnota typu T_i, potom $\omega < t_i$. Relácia $\leq$ je reflexívny uzáver relácie $<$: ak $\omega < t_i$, potom $\omega \leq t_i$; okrem toho, pre každé x platí $x \leq x$. ■

Nejakú k-ticu konštánt v budeme považovať za zúplnenie k-tice u, ak v žiadnom z „rozmerov“ nie je v horšia ako u. (Toto usporiadnie rozšírime prirodzeným spôsobom aj na atomárne formuly.) Formálne:

Definícia 2.24 (Zúplnenie) Ak máme k-tice $u = (u_1, \ldots, u_k)$ a $v = (v_1, \ldots, v_k)$, kde u_i, v_i sú buď hodnoty typu T_i alebo Skolemove konštanty, potom $u \leq v$ práve vtedy, keď pre každé i platí $u_i \leq v_i$.

Podobne, ak máme atómy $U = p(u_1, \ldots, u_k)$ a $V = p(v_1, \ldots, v_k)$, potom $U \leq V$, práve vtedy, keď pre každé i platí $u_i \leq v_i$. ■

Uvažujme selekciu $\sigma_\phi\ r$, kde ϕ je formula, v ktorej sa vyskytuje nejaký atribút A_i, ale nevyskytujú sa v nej nulové hodnoty a v relácii r sú niektoré z hodnôt atribútu A_i nulové. Ako interpretovať (a vyhodnotiť) $\sigma_\phi\ r$? Ponúkajú sa dve krajné riešenia (a celé spektrum riešení medzi nimi).

Intuitívne, jedna krajnosť je tá, že podmienku ϕ nepovažujeme za splnenú v nejakom riadku tabuľky r, ak hodnota príslušného atribútu A_i v tomto riadku je nulová.

Druhú krajnosť predstavuje, že podmienku ϕ považujeme – intuitívne – za splnenú (v riadku z r), ak existuje také nahradenie nulovej hodnoty z tohto riadku konštantou príslušného typu, pre ktorú je ϕ pravdivá.

Príklad 2.25 Použime tabuľku T z príkladu 2.1, iba ju trochu zmodifikujme.

RODIČ	*DIEŤA*
anna	ω
karol	miro
karol	zuza
milan	štefan
zuza	eva
ondrej	eva

V prvom riadku, v druhom stĺpci je nulová hodnota. Predpokladajme, že odpoveďou na otázku $\sigma_{DIE\check{T}A=miro}(T)$ bude tabuľka, obsahujúca prvé dva riadky. Interpretujeme to tak, že nie je vylúčené, že *miro* je dieťaťom *anny* (existuje zodpovedajúce nahradenie nulovej hodnoty). Ak by sme prijali opačnú krajnú polohu, odpoveďová tabuľka bude obsahovať iba druhý riadok (nemôžeme zaručiť, že *miro* je dieťaťom *anny*). ■

Formálnejšie:

Definícia 2.26 (Zúplnenie relácií) Nech $r, r' \subseteq T_1 \times \cdots \times T_k$ sú relácie. Budeme hovoriť, že $r \leq r'$, ak $\forall u \in r\ \exists v \in r'\ u \leq v$. ■

Poznámka 2.27 Zdanlivo je táto definícia veľmi slabá: Stačí totiž, ak v r' existuje jediná veta, ktorá je zúplnením všetkých viet z r.

Keď si však uvedomíme, že nejaká veta je zúplnením inej vety iba vtedy, keď je s ňou zhodná alebo sa od nej líši iba tým, že má miesto Skolemových konštánt (nulových hodnôt) regulárne koštanty, vidíme, že informačný obsah takejto r by bol veľmi slabý. ■

Predstavme si teraz nejakú reláciu r s nulovými hodnotami. Túto reláciu možno zúplniť rôznym spôsobom. Množinu možných zúplnení r označme ako W, pričom predpokladáme, že W obsahuje iba relácie bez nulových hodnôt. W teda predstavuje nejaké spektrum alternatív, zodpovedajúcich neúplnej informácii z r. Ak $s \in W$, môžeme hovoriť o nejakom vzťahu medzi r a s: s je možným zúplnením r. Tejto predstave o možných zúplneniach relácie s nulovými hodnotami zodpovedá silná a dôležitá sémantická štruktúra. Takáto štruktúra sa nazýva Kripkeho štruktúrou. Teraz zavedieme špeciálny prípad tohto pojmu.

Definícia 2.28 (Kripkeho štruktúra) Nech W je množina relácií (bez nulových hodnôt), ktoré interpretujú predikát $p(A_1, \ldots, A_k)$ a interpretáciou toho istého predikátu je aj relácia s nulovými hodnotami r. *Kripkeho štruktúrou* nazvime trojicu (W, r, ρ), kde ρ je relácia dostupnosti $\{(r, r') \ : \ r' \in W \wedge r \leq r'\}$. ■

Teda, $r' \in W$ je zúplnením r. Prvky z W budeme nazývať aj možnými svetmi.

Prichádzame k jadru problému, k tomu, ako vyhodnocovať selekčné podmienky nad databázou s neúplnou informáciou. Základná intuícia je jednoduchá. Už sme sa s ňou zoznámili a teraz ju preformulujeme: Jeden možný krajný prístup je požadovať, aby selekčná podmienka bola splnená v každom zúplnení. Druhou krajnosťou je, že selekčnú podmienku považujeme za splnenú v relácii s neúplnou informáciou, ak je možné také zúplnenie tejto relácie, kde táto podmienka je splnená

Teraz formálnejšie: Zrekapitulujeme, že databázu si predstavujeme ako množinu základných atómov (niektoré z argumentov môžu byť Skolemove konštanty). Predpokladáme aj nejakú databázovú schému a nejaký jazyk, formulujeme v ňom selekčné podmienky. Predpokladajme ďalej, že vieme rozhodnúť, či je nejaká selekčná podmienka splnená pre nejakú vetu bez nulových hodnôt. (Tento predpoklad je bez problémov splnený v obvyklých relačných databázach.) Najprv zadefinujeme (modálne) operátory $\Box, \Diamond$. $\Box$ môžeme čítať ako *nutne platí* a $\Diamond$ ako *môže platiť*. Základná intuícia je takáto: $\Box\phi$ ($\Diamond\phi$) sa považuje za pravdivé v nejakom možnom svete w, ak ϕ je pravdivé v každom (aspoň niektorom) možnom svete dostupnom z w.

Definícia 2.29 (Operátory $\Box$, $\Diamond$) Nech $\mathcal{K} = (W, r, \rho)$ je Kripkeho štruktúra. Nech ϕ je formula. Hovoríme, že v $\mathcal{K}$ je pre vetu V relácie r splnená formula

$\Box\phi$ ak v každej r' dostupnej z r existuje veta $V' \in r'$ taká, že $V \leq V'$ a ϕ je splnená vo V',

$\Diamond\phi$ ak existuje r' dostupná z r a v nej existuje veta $V' \in r'$ taká, že $V \leq V'$ a ϕ je splnená vo V'.

■

Jeden extrém pri vyhodnocovaní selekčnej podmienky ϕ môžeme chápať ako vyhodnocovanie podmienky $\Box\phi$, druhý ako $\Diamond\phi$. Podmienka ϕ je syntakticky jednoduchšia, ale treba ju vyhodnocovať nad reláciou s nulovými hodnotami. Podmienky $\Box\phi$ alebo $\Diamond\phi$ sú síce syntakticky zložitejšie, ale vyhodnotiteľné nad reláciami bez nulových hodnôt, kde sa vieme pohybovať suverénnejšie.

Prijmeme teraz dve konvencie, ktoré zodpovedajú spomínaným extrémnym možnostiam vyhodnocovania selekčných podmienok. Prvá z nich stotožňuje $\sigma_\phi(r)$ s vyhodnotením $\sigma_{\Box\phi}$, druhá s vyhodnotením $\sigma_{\Diamond\phi}$. Pripomeňme, že

$\sigma_\phi(r)$ je selekčná formula: vyberá z relácie r tie riadky, ktoré spĺňajú podmienku ϕ.

V nasledujúcich konvenciách predpokladáme, že $\mathcal{K} = (W, r, \rho)$ je Kripkeho štruktúra, ϕ, $\Box\phi$ a $\Diamond\phi$ sú selekčné formuly. Obe konvencie zodpovedajú vyššie spomínaným krajným možnostiam vyhodnocovania selekcie v relácii s nulovými hodnotami. Samozrejme, tieto konvencie sú alternatívne – buď prijmeme jednu alebo druhú.

Konvencia 2.30 Dohodneme sa, že $\sigma_\phi(r)$ budeme považovať za splnené pre vetu $V \in r$ práve vtedy, keď je pre V splnené $\sigma_{\Box\phi}$. ■

Konvencia 2.31 Dohodneme sa, že $\sigma_\phi(r)$ budeme považovať za splnené pre vetu $V \in r$ práve vtedy, keď je pre V splnené $\sigma_{\Diamond\phi}$. ■

Zhrňme: V súvislosti s vyhodnocovaním selekcie sme teda na jednom extréme videli prístup, ktorý selekčnú podmienku považuje za pravdivú iba vtedy, keď je určite pravdivá, na druhej strane bol extrém, ktorý ju považuje za pravdivú, keď môže byť pravdivá. Obidve situácie – môžeme ich chápať ako nutnosť a možnosť – sú z hľadiska reprezentácie znalostí relevantné. Formálnym štúdiom nutnosti a možnosti sa zaoberá modálna logika, v tomto texte jej však systematickú pozornosť venovať nebudeme.

Prvá krajnosť predstavuje prístup, ktorý sa zaobíde bez prijímania hypotéz (resp. vyhýba sa mu). Druhá krajnosť je prvým príkladom hypotetického usudzovania, s ktorým sa v tomto texte stretávame: hypoteticky skúšame nulové hodnoty nahradiť nenulovými a skusmo pracujeme s takto zvolenými hodnotami. Priestor medzi týmito krajnosťami by sa dal zúžiť vhodným zúžením relácie dostupnosti z definície 2.28.

Predchádzajúce odstavce nazerajú na problém skôr cez optiku reprezentácie znalostí a slúžili aj ako príležitosť pre prvé oboznámenie so sémantikou modálnej logiky. Vlastné databázové hľadisko nie je pre nás rozhodujúce a dôležité. Medzi dôležitými prácami, ktoré problematiku nulových hodnôt v relačných databázach študujú, sú [Rei 86, Var 86, AKG 91, ImL 84, Lip 81, Lip 77].

Reiter sa v [Rei 86] venuje zovšeobecneniu relačnej algebry na prípad databáz s nulovými hodnotami a algoritmu vyhodnocovania otázok, v nej formulovaných. Ukázal, že tento algoritmus je korektný, t.j. všetko, čo vypočíta ako odpoveď na danú otázku, je skutočne odpoveďou na túto otázku. Nie je však úplný, niektoré odpovede na otázku nedokáže vypočítať. Algoritmus však má niektoré dobré vlastnosti: pre databázy bez nulových hodnôt, pre pozitívne otázky a pre všeobecne kvantifikované pozitívne otázky je úplný.

Vardi [Var 86] dôsledne uplatňuje optiku, podľa ktorej je adekvátne databázu ponímať ako teóriu, ak chceme reprezenovať neúplnú informáciu.[21] Chápať databázu ako teóriu znamená, že na ňu nazeráme ako na množinu

[21] Bežne sa databáza chápe ako interpretácia, či model databázovej schémy a integritných obmedzení. Tak sme to robili aj my v tejto kapitole.

axióm. Odpovede na otázky sa odvodzujú z axióm. Tieto odpovede tvoria implicitný obsah databázy. Jazyk, v ktorom sa príslušné axiómy píšu, obsahuje konečný počet konštánt, konečný počet predikátových symbolov a žiadne funkčné symboly. Axiómy, ktoré tvoria databázu:

1. Atomické základné fakty, databázové vety.
2. Predpoklad jediného mena (UNA, unique name assumption): pre niektoré konštanty c_i, c_j sa prijímajú axiómy tvaru $c_i \neq c_j$.
3. Predpoklad uzavretosti domén (DCA, domain closure assumption): ak $c_1, \ldots, c_n$ sú všetky konštanty jazyka J, potom $\forall y(y = c_1 \vee \cdots \vee y = c_n)$.
4. Axióma uzáveru:
   Nech $p(\overline{t_1}), \ldots, p(\overline{t_s})$ sú všetky atomické základné fakty s predikátovým symbolom p, kde $\overline{t}$ reprezentuje k-ticu termov $t_1, \ldots, t_k$ a k je arita predikátového symbolu p.

   Potom platí $(\forall \overline{y})(p(\overline{y}) \Rightarrow \overline{y} = \overline{t_1} \vee \cdots \vee \overline{y} = \overline{t_s})$.

   Ak neexistuje medzi axiómami žiadny atóm s predikátovým symbolom p, potom sa prijíma, že $\forall \overline{y} \neg p(\overline{y})$.

Axióma uzáveru vlastne vyjadruje predpoklad uzavretého sveta, ktorý budeme podrobnejšie diskutovať v kapitole 3 i neskôr: ak niečo nemáme zaznamenané, potom to neplatí. Ak predpoklad jediného mena formulujeme pre každú dvojicu konštánt z jazyka J, máme databázu s úplnou informáciou. V opačnom prípade nevylučujeme, že niektoré z dvojíc konštánt sú menami toho istého indivídua, teda pripúšťame nulové hodnoty (či Skolemove konštanty) a tým aj neúplnú informáciu. Predstavme si totiž, že platí $p(\omega, b)$ a nemáme prijatú žiadnu axiómu tvaru $\omega \neq a$ pre žiadne a. Potom, samozrejme $p(\omega, b)$ sa nemusí líšiť od $p(a, b)$.

Takáto axiomatická teória má vo všeobecnosti viac modelov, každý predstavuje jeden z možných stavov databázy.

Viac možností, ako reprezentovať databázy s neúplnou informáciou, diskutujú [AKG 91].

Podmienky sémantickej korektnosti definícií relačných operácií pre databázy s nulovými hodnotami sa skúmajú v [ImL 84]. Lipski [Lip 77, Lip 81] sa venuje problému horného a dolného ohraničenia zúplnení databáz s nulovými hodnotami. Ak sme vyššie rozvíjali predstavu, že hypoteticky skúšame nulové hodnoty nahradiť nenulovými a skusmo pracujeme s takto zvolenými hodnotami, môžeme povedať, že Lipski uvažuje isté hranice, v rámci ktorých je rozumné skutočnú hodnotu aproximovať hypotetickými hodnotami.

Ešte sa zmienime o ďalších možnostiach reprezentácie neúplnej informácie v databázach.

Nad konečnou doménou možno vety tvaru $\exists X V(X)$, kde $V(X)$ je atóm, obsahujúci voľnú premennú X, nahradiť *disjunkciami* tvaru

$$V(a_1) \vee \cdots \vee V(a_k),$$

pričom $a_1, \ldots, a_k$ sú všetky hodnoty z príslušnej domény. Samozrejme, disjunkcie môžu vyjadriť aj to, čo existenčná veta uvedeného tvaru nevyjadrí, napríklad $p(a) \vee q(a)$. Disjunkciou môžeme vyjadriť aj viac konkrétnu informáciu ako pomocou existenčnej vety. Napríklad pre schému

$$zamestnanec(Meno, Bydlisko, Plat)$$

je informácia

$$zamestnanec(milan, Bratislava, 6000) \vee zamestnanec(milan, Modra, 6000)$$

konkrétnejšia ako informácia $\exists X\, zamestnanec(milan, X, 6000)$.

Analogicky, existenčné vety môžeme nahradiť vetami, ktoré obsahujú *množinové* alebo *intervalové* hodnoty. Opäť platí, že množinové a intervalové hodnoty umožňujú vyjadriť aj špecifickejšiu informáciu než existenčné vety. Existenčné vety sú vlastne krajným prípadom viet s množinovými hodnotami: existenčne viazaná premenná reprezentuje množinu všetkých možných hodnôt daného atribútu. Ak je množina hodnôt tohto atribútu usporiadaná, môžeme povedať, že existenčná veta je krajným prípadom intervalových hodnôt.

Databázy s neúplnou informáciou sa študujú ako nástroj na zvládnutie tých situácií, keď o niektorých databázových objektoch nemáme k dispozícii všetky zbierané informácie. Hodnoty v niektorých stĺpcoch nie sú s určitosťou známe. Ako nástroje na popísanie tejto okolnosti sa používajú tzv. nulové hodnoty, disjunkcie alebo existenčné kvantifikátory. Nevenovali sme sa pomerne dôležitému rozlíšeniu – neexistencia hodnoty v riadku relácie môže znamenať jednak, že skutočná hodnota je neznáma, ale aj, že žiadna hodnota atribútu pre danú „entitu“ neexistuje. Dobrým príkladom neexistujúcich hodnôt pre niektoré atribúty sú medicínske databázy. Napríklad, nie každý pacient musí mať absolvované všetky typy vyšetrení. Samozrejme, špecifikácia vhodnej inferencie nad takto skomplikovaným problémom je ešte náročnejšia. Nebudeme sa tejto otázke venovať.

Teraz prejdime iba letmo k príkladu 2.17, ktorý ukazuje, že niekedy môže byť zaujímavé v databáze vyjadriť naše presvedčenie o hodnovernosti údajov. Asi najjednoduchší spôsob reprezentácie je ten, ktorý každú relačnú schému rozšíri o jeden stĺpec, ktorého hodnoty budú vyjadrovať váhu, ktorú pripisujeme platnosti danej vety. Oveľa drahší spôsob by priraďoval ku každej hodnote v každom stĺpci každého riadku jej dôveryhodnosť.

Samozrejme, prosté rozšírenie jazyka nespôsobuje nijaké závažné komplikácie. Problémy vzniknú vtedy, keď chceme špecifikovať operátor odvodenia nad takto rozšíreným databázovým jazykom. Ide predovšetkým o to, že pre operáciu, do ktorej vstupujú dva komponenty s dvoma rôznymi hodnotami dôveryhodnosti treba určiť, akú dôveryhodnosť pripísať výsledku tejto operácie. Tu sa problému venovať nebudeme, je mimo hlavnej pozornosti tohto textu. Rovnako v celom texte ostáva problém neurčitosti a neistoty, stupňov dôveryhodnosti pri odvodzovaní mimo nášho záujmu.

2.3 Deduktívna databáza

Od motivačného príkladu 2.18 (tranzitívny uzáver) prejdeme k ďalšiemu pokusu zosilňovať vyjadrovacie schopnosti relačnej algebry. Toto zosilnenie nás povedie k deduktívnym databázam. V nich budeme schopní vyjadriť otázku z príkladu 2.18.

Rekapitulujme: V časti 2.1 sme si znalosti intuitívne stotožnili s riadkami v tabuľkách (bez ohľadu na to, či priamo zapamätaných alebo vypočítaných).

Vypočítané tabuľky boli odpoveďami na nejaké otázky. Uviedli sme si tri príklady jazykov, v ktorých sa formulujú otázky: QBE, relačný kalkul a relačnú algebru. Relačnú algebru sme si zvolili ako sémantickú špecifikáciu stroja odpovedajúceho na otázky databázového jazyka nad relačnou databázou.

Aby sme sa dostali ďalej, zavedieme si ešte jeden typ formalizácie databázového jazyka. Je založený na jazyku logického programovania (trochu okliešteným – o funkčné symboly).[22]

Výhodou tohto jazyka je, že otázky, integritné obmedzenia a fakty vyjadríme v tom istom jazyku. Ďalej, prirodzene v ňom zavedieme dedukciu. Navyše, pripravíme si pôdu pre komplikovanie predstavy o databáze, ktoré bude nasledovať v kapitole 3.

Skôr než prejdeme k charakterizácii tohto jazyka, drobné upozornenie. Budeme používať implikáciu orientovanú opačne, než je to obvyklé. Prijímame tým konvenciu, známu z logického programovania. Idea, ktorá je za týmto zápisom: problém (výpočet) A sa redukuje na problém G. Ak mám vypočítať A a platí $A \leftarrow G$, musím prv vypočítať G. Nech však píšeme $A \leftarrow G$, či $G \rightarrow A$, vždy tomu rozumieme takto: „ak platí G, potom platí A“. Ak je G prázdne, výpočet A je triviálny, A je fakt. Fakt $A \leftarrow$ môžeme čítať ako „platí A“. Deklaratívnemu a procedurálnemu čítaniu implikácií tvaru $A \leftarrow G$ (klauz) je venovaná poznámka 2.40.

Pripomeňme, že E predstavuje v ľubovoľnej reprezentácii znalostí množinu viet. V štandardnej databáze so schémou S môže E obsahovať vety nasledovného typu:

fakty $A \leftarrow$, kde A je základný atóm bez funkčných symbolov, pričom predikátový symbol z A sa vyskytuje v S,

otázky $\leftarrow A_1, \ldots, A_n$, kde A_i sú atómy (spravidla nie základné), použité predikátové symboly sa buď vyskytujú v S alebo patria medzi symboly jazyka, v ktorom sa formulujú selekčné kritériá,[23]

integritné obmedzenia syntakticky sa nelíšia od otázok; integritné obmedzenia budeme chápať ako otázky, na ktoré sa neočakáva pravdivá odpoveď; ak by sa taká odpoveď našla, svedčí o tom, že databáza nie je v „rozumnom, zdravom“ stave.

---

[22] ⟨ dodatok D ⟩

[23] Napríklad $<$, $=$ a pod. Ako príklad selekčného kritéria si môžeme uviesť $Plat > 5000$.

Aj teraz databázu stotožníme s množinou nejakých viet a operácií na týchto vetách. Odpovede na všetky možné otázky budeme považovať za znalosti, ktoré sú reprezentované v databáze.

Vrátime sa k nášmu príkladu rodinnej databázy. V nasledujúcom príklade si vyjadríme databázu z príkladu 2.6 a súčasne dodáme jedno integritné obmedzenie, ktoré hovorí, že v rozumnej databáze by nemal byť ani jeden objekt, ktorý je súčasne muž i žena.

Príklad 2.32 Fakty:

$$\begin{array}{lll}
rodi\check{c}(anna, miro) \leftarrow & mu\check{z}(miro) \leftarrow & \check{z}ena(anna) \leftarrow \\
rodi\check{c}(anna, zuza) \leftarrow & mu\check{z}(\check{s}tefan) \leftarrow & \check{z}ena(zuza) \leftarrow \\
rodi\check{c}(anna, \check{s}tefan) \leftarrow & mu\check{z}(karol) \leftarrow & \check{z}ena(eva) \leftarrow \\
rodi\check{c}(karol, miro) \leftarrow & mu\check{z}(milan) \leftarrow & \\
rodi\check{c}(karol, zuza) \leftarrow & mu\check{z}(ondrej) \leftarrow & \\
rodi\check{c}(milan, \check{s}tefan) \leftarrow & & \\
rodi\check{c}(zuza, eva) \leftarrow & & \\
rodi\check{c}(ondrej, eva) \leftarrow & &
\end{array}$$

Integritné obmedzenie: $\leftarrow \check{z}ena(X), mu\check{z}(X)$. ■

Teraz by sme sa mohli začať podrobnejšie venovať pravidlám. V klasickej databáze je istá asymetria medzi explicitnými a implicitnými znalosťami. Na vzťah starého rodičovstva sme sa v príklade 2.4 pýtali iba nepriamo (bez použitia tohto pomenovania). Podobne, na databázu uvedenú v príklade 2.32 nemôžeme priamo adresovať (s použitím vhodných predikátových symbolov) otázky na také rodinné vzťahy, ako sú otec, matka, brat, starý otec, strýko atď. Dosiahneme to však, keď rozšírime náš jazyk o pravidlá.

Príklad 2.33 Otázku z príkladu 2.4 možno preformulovať takto: X je starým rodičom Y, ak existuje Z také, že X je jeho rodičom a súčasne Z je rodičom Y. Táto formulácia má tvar pravidla: ak platia nejaké predpoklady, tak platí nejaký dôsledok.

Pravidlo si zapíšeme v práve zavádzanom formalizme:

$$star\acute{y}\_rodi\check{c}(X, Y) \leftarrow rodi\check{c}(X, Z), rodi\check{c}(Z, Y).$$

■

Nasledujúci príklad ukazuje, že otázke adresovanej relačnej databáze zodpovedajú pravidlá. Každú otázku možno preformulovať na pravidlo. A naopak, každému pravidlu zodpovedá nejaká otázka adresovaná klasickej databáze. Pozri aj [Ull 88].

Príklad 2.34 Pomocou pravidiel možno pohodlne definovať zjednotenie relácií p, q (predikát $zj_{p,q}$), ich kartézsky súčin ($cart_{p,q}$), rozdiel ($rozd_{p,q}$), selekciu (sel_p) a projekciu ($proj_p$).

$$\begin{array}{rcl} zj_{p,q}(X_1,\ldots,X_k) & \leftarrow & p(X_1,\ldots,X_k) \\ zj_{p,q}(X_1,\ldots,X_k) & \leftarrow & q(X_1,\ldots,X_k) \end{array}$$

Zmysel uvedenej dvojice pravidiel možno vyjadriť takto: Predikát $zj_{p,q}$ je pravdivý pre nejakú k-ticu ξ, ak je pre ξ pravdivé p (podobne pre q).

$$cart_{p,q}(X_1,\ldots,X_k,Y_1,\ldots,Y_m) \leftarrow p(X_1,\ldots,X_k), q(Y_1,\ldots,Y_m)$$

Predikát $cart_{p,q}$ je pravdivý pre nejakú $k+m$-ticu ξ, ak pre jej prvých k argumentov je pravdivé p a pre zvyšných m argumentov je pravdivé q.

$$rozd_{p,q}(X_1,\ldots,X_k) \leftarrow p(X_1,\ldots,X_k), not\ q(X_1,\ldots,X_k)$$

Predikát $rozd_{p,q}$ je pravdivý pre nejakú k-ticu ξ, ak je pre ξ pravdivé p a nepravdivé q.

$$sel_p^{\phi}(X_1,\ldots,X_k) \leftarrow p(X_1,\ldots,X_k), \phi,$$

kde ϕ je ľubovoľná selekčná podmienka. Predikát sel_p^{ϕ} je pravdivý pre nejakú k-ticu ξ, ak je pre ξ pravdivé p a ϕ.

$$proj_p^{i_1,\ldots,i_l}(X_{i_1},\ldots,X_{i_l}) \leftarrow p(X_1,\ldots,X_k),$$

kde $(i_1,\ldots,i_l)$ je podpostupnosť postupnosti $(1,\ldots,k)$, pričom $l < k$. Predikát $proj_p^{i_1,\ldots,i_l}$ je pravdivý pre nejakú l-ticu ξ, ak ξ tvoria komponenty $i_1,\ldots,i_l$ k-tice splňujúcej p. ■

Môžeme si doplniť definíciu jazyka, ktorý budeme používať. Okrem faktov, otázok a integritných obmedzení pripustíme aj pravidlá:

pravidlá $A \leftarrow A_1,\ldots,A_n$, kde A, A_i sú atómy (spravidla nie základné), použité predikátové symboly sa vyskytujú v S alebo sú predikátmi jazyka, v ktorom sa vyjadrujú selekčné podmienky.

Jazyk, ktorý obsahuje predikátové symboly, konštanty, premenné a možno v ňom formulovať podľa uvedenej syntaxe fakty, pravidlá, integritné obmedzenia a otázky, nazývame *datalog*. Zvlášť poznamenajme, že neobsahuje funkčné symboly a dodajme, že zatiaľ budeme používať datalog bez negácie. To znamená, že nepripustíme také pravidlá, ako bola definícia rozdielu z príkladu 2.34.

Príklad 2.35 Do databázy môžu patriť okrem faktov uvedených v príklade 2.32 aj nasledujúce pravidlá:

$$\begin{aligned} otec(X,Y) &\leftarrow rodi\check{c}(X,Y), mu\check{z}(X),\\ matka(X,Y) &\leftarrow rodi\check{c}(X,Y), \check{z}ena(X),\\ brat(X,Y) &\leftarrow matka(M,X), matka(M,Y), mu\check{z}(X),\\ str\acute{y}ko(X,Y) &\leftarrow rodi\check{c}(R,Y), brat(X,R). \end{aligned}$$

Implicitnými znalosťami teraz budú aj všetky základné fakty s predikátovými symbolmi *otec, matka, brat, strýko*, ktoré vyplývajú z explicitných znalostí. Implicitné znalosti môžeme stotožniť s deduktívnymi dôsledkami. Túto predstavu ešte upresníme. Teraz iba toľko, že oveľa prirodzenejšie ako doteraz môžeme za znalosti implicitné v našej databáze považovať znalosti o tom, že karol je mirovým otcom, anna zuzinou matkou, miro zuziným bratom, eviným strýkom a pod. (keď chceme, aj s prevalcovaním syntaxe slovenčiny syntaxou prologovskou).[24] ■

Terminológia: Základná predstava o datalogovskej (deduktívnej) databáze je taká, že v jej podloží existuje obvyklá relačná databáza. Každej tabuľke tejto databázy zodpovedá predikát, ktorý nazývame *extenzionálny* (budeme používať skratku EDB-predikát). Predikáty, ktorým nezodpovedajú tabuľky, sú definované nejakými pravidlami (ako napríklad vyššie $otec, matka, brat$). Nazývajú sa intenzionálne predikáty (IDB-predikáty). Ich význam sa vypočítava. K spôsobom ich výpočtu sa ešte vrátime.

To, čo sme zatiaľ videli, je iba kozmetická zmena. Zodpovedá databázovým pohľadom. Databázový pohľad si ilustrujeme príkladom:

Príklad 2.36 V príklade 2.35 sme mali pravidlo

$$str\acute{y}ko(X,Y) \leftarrow rodi\check{c}(R,Y), brat(X,R).$$

Môžeme si predstaviť, že máme tabuľku $STR\acute{Y}KO$. Táto tabuľka však nie je priamo zapamätaná v databáze, nazývame ju *pohľadom* a vypočítava sa z tabuliek $RODI\check{C}$ a $BRAT$. Tabuľka $RODI\check{C}$ je priamo zapamätaná v databáze, tabuľku $BRAT$ tiež vypočítavame (napríklad podľa pravidla z príkladu 2.35).

[24] Každé pravidlo vyjadruje nejakú znalosť – napríklad o tom, čo to je starý rodič. Dá sa očakávať námietka, že medzi touto znalosťou a znalosťou, že Anna je rodičom Zuzy, je značný kvalitatívny rozdiel. Nechcem to popierať, ale zavedením rozlišovania medzi kvalitou resp. hodnotou znalosti by sme sa dostali do sotva prekonateľných ťažkostí. Preto ľubovoľná veta ľubovoľného reprezentačného jazyka bude vyjadrovať nejakú znalosť. Medzi rôznymi druhmi znalostí budeme rozlišovať iba vtedy, keď budeme mať na to jasné syntaktické alebo sémantické dôvody. Na základe syntaktického kritéria už rozlišujeme medzi faktami a pravidlami. Toto rozlíšenie však pre nás zatiaľ nie je dôležité z nijakého ďalšieho hľadiska. Ak by sme chceli rozlíšiť dáta a znalosti tak, že dáta sú vyjadrené ako fakty a znalosti ako nedegenerované pravidlá, dostali by sme sa do závozu: Takto poňaté znalosti môžeme získať aj bez používania pravidiel. Videli sme totiž, že to, čo dokážeme vyjadriť pravidlami, dokážeme vyjadriť aj pomocou otázok v relačných databázových jazykoch. Až zavedenie rekurzie zosilní operátor inferencie, a tým aj vyjadrovacie možnosti.

Pravá strana pravidla, definujúceho strýka, brata atď., je vlastne otázka. Pohľad teda umožňuje užívateľovi relačnej databázy pracovať s „virtuálnymi" tabuľkami, s množinami odpovedí na nejakú otázku adresovanú databáze. ■

Znalosti, ktoré sú implicitné v štandardnej databáze, sme pochopili ako odpovede na nejaké otázky. Videli sme aj, že implicitné znalosti treba chápať relatívne k databázovému jazyku: niektoré nie sme schopní získať, pokiaľ disponujeme limitovaným databázovým jazykom, napríklad tak limitovaným ako relačný kalkul. Ukážeme si však, že po zosilnení databázového jazyka ich však už získať vieme.

Podstatný vyjadrovací rozdiel môžeme v datalogu (v porovnaní s relačnou algebrou) dosiahnuť až vtedy, keď dovolíme písať skupiny rekurzívnych pravidiel. Rekurzívne pravidlo obsahuje v tele ten istý predikátový symbol, čo sa vyskytuje aj v jeho hlave. Relácia, ktorou sa interpretuje tento symbol, sa postupne konštruuje, dodefinováva. V nasledujúcom príklade sa relácia *predok* definuje takouto konštrukciou.

Príklad 2.37 V datalogu dokážeme vyjadriť aj tranzitívny uzáver binárnej relácie:

$$\begin{array}{rcl} predok(X,Y) & \leftarrow & rodi\check{c}(X,Y), \\ predok(X,Y) & \leftarrow & rodi\check{c}(X,Z), predok(Z,Y). \end{array}$$

■

Videli sme teda na konkrétnom príklade, že naša schopnosť reprezentovať nejaké znalosti vzrástla vďaka tomu, že sme zaviedli bohatší jazyk. Jazyk, ktorý umožňuje rekurziu.

Teraz môžeme prejsť k presnejšiemu vyjadreniu štvorice (J, E, Cn, Ind) pre datalogovské databázy. Predpokladajme, že Ind je prázdne. Predpokladajme ďalej, že J je vymedzené nejakými množinami predikátových symbolov a konštánt. Možno v ňom tvoriť fakty, pravidlá, integritné obmedzenia a otázky. E budeme charakterizovať po pomocnej definícii:

Definícia 2.38 *Literál* je atóm alebo jeho negácia. *Klauza* je disjunkcia literálov. *Hornovská klauza* je klauza, ktorá obsahuje nanajvýš jeden pozitívny literál. *Definitná klauza* je hornovská klauza, ktorá obsahuje presne jeden pozitívny literál.

Základná klauza neobsahuje žiadnu premennú. ■

Klauzy môžeme zapisovať aj ako implikácie tvaru $A_1 \vee \cdots \vee A_k \leftarrow B_1, \ldots, B_n$, kde $n \geq 0$, $k \geq 0$ A_i, B_i sú atómy.

Ľahko to možno preveriť pomocou výrokovej logiky: $A_1 \vee \cdots \vee A_k \leftarrow B_1, \ldots, B_n$ upravíme pomocou $(p \leftarrow q) \equiv (p \vee \neg q)$ na $A_1 \vee \cdots \vee A_k \vee \neg(B_1 \wedge \cdots \wedge B_n)$ a podľa $\neg(p \wedge q) \equiv (\neg p \vee \neg q)$ na $A_1 \vee \cdots \vee A_k \vee \neg B_1 \vee \cdots \vee \neg B_n$.

Klauza je *hornovská*, ak $k \leq 1$. *Definitná* je, ak $k = 1$. V integritnom obmedzení a otázke je $k = 0$, v pravidle je $n > 0$, vo fakte je $n = 0$. Ak máme hornovskú klauzu $A \leftarrow B_1, \ldots, B_n$, potom A nazývame *hlavou* klauzy, jej *telom* je $B_1, \ldots, B_n$.

Poznámka 2.39 Venujme pozornosť pojmu integritného obmedzenia. Podľa práve uvedených manipulácií môžeme integritné obmedzenie $\leftarrow B_1, \ldots, B_n$ vyjadriť ako $\neg B_1 \vee \cdots \vee \neg B_n$. Táto formula je splnená vtedy, keď aspoň jedno B_i je nepravdivé. Napríklad, formula $\leftarrow mu\check{z}(X), \check{z}ena(X)$ je pre konkrétnu inštanciu X splnená vtedy, keď buď $mu\check{z}(X)$ alebo $\check{z}ena(X)$ je nepravdivé (pre túto inštanciu). Ak sú všetky B_i pravdivé, integritné obmedzenie je narušené. Teda, ak by boli aj $\check{z}ena(X)$ aj $mu\check{z}(X)$ pravdivé pre danú inštanciu X, databáza by nebola v rozumnom stave. ■

Poznámka 2.40 Dôležité je rozlíšiť deklaratívne a procedurálne čítanie definitných kláuz. Procedurálne čítanie: ak chceme vypočítať (zistiť, či) A, musíme najprv vypočítať (zistiť, či) B_1 a B_2 a $\ldots B_n$. Deklaratívne čítanie: ak je pravdivé B_1 a B_2 a $\ldots B_n$, potom je pravdivé aj A. Pre sémantickú charakterizáciu je relevantné deklaratívne čítanie. ■

Môžeme sa vrátiť k vymedzeniu E pre datalogovské databázy. Datalogovská databáza je konečná množina hornovských kláuz.[25]

Ostáva nám charakterizovať Cn pre datalogovské databázy. Budeme ho charakterizovať ako operátor (limitovanej) dedukcie. Základná intuícia je, že nejaký záver je dedukovateľný z nejakej množiny predpokladov, ak pravdivosť predpokladov zaručuje pravdivosť záveru. Začneme príkladom. Predstavíme si situácie, keď nejaká veta nie je dedukovateľná z predpokladov.

Príklad 2.41 Vyplýva veta *„Niektoré zvieratá sú hnedé“* z nižšie uvedených predpokladov?

- Niektoré medvede sú hnedé.
- Niektoré zvieratá sú medvede.

Zaručuje v našom príklade pravdivosť predpokladov pravdivosť záveru? Ak sa nám podarí nájsť takú interpretáciu, pri ktorej sú predpoklady pravdivé a záver nepravdivý, môžeme tvrdiť, že nie. A budeme to považovať za demonštráciu toho, že záver nevyplýva z predpokladov. Na to si však najprv musíme problém sformalizovať:

[25] Vidíme, že jazyk datalogu je limitovaný. Nepripúšťa plné vyjadrovacie možnosti predikátovej logiky. Keďže v hlave pravidla nemôže byť nič iné, iba atóm, nedá sa odvodiť, a teda vyjadriť, ani $p \vee q$. Máme príklad ďalšej reprezentácie znalostí, neumožňujúcej odvodiť vetu, ktorá môže byť v databáze implicitná z hľadiska inej reprezentácie znalostí.

$$\frac{\begin{array}{c}\exists X(m(X) \wedge h(X)) \\ \exists X(z(X) \wedge m(X))\end{array}}{\exists X(z(X) \wedge h(X))}$$

Ak predikát m interpretujeme množinou prvočísel, z množinou párnych a h množinou nepárnych čísel, máme skonštruovaný príklad, ktorý ukazuje, že usudzovaním podľa uvedenej schémy môžeme získať z pravdivých predpokladov nepravdivé závery. Týmto zistením sa nevyvracia pravdivosť vety „Niektoré zvieratá sú hnedé“. Konštatuje sa iba toľko, že táto veta nie je deduktívnym dôsledkom uvedených predpokladov. ■

Teda: veta vyplýva z množiny predpokladov, ak nemožno nájsť *protipríklad* (pre odvodenie tej istej formy), pri ktorom sú všetky predpoklady pravdivé a záver nepravdivý.

Na presný výklad v tejto chvíli potrebujeme definície splňania, pravdivosti, modelu. Naším hlavným cieľom je však rýchly postup do tretej kapitoly, uspokojíme sa preto s intuíciami.[26] *Model* nejakej množiny viet M je taká intepretácia, v ktorej je pravdivá každá veta z M. Uvedieme si aj prvú predstavu o herbrandovskej(om) interpretácii(modeli). Herbrandovskú interpretáciu si v tejto chvíli predstavíme ako množinu základných atómov. Každý atóm, ktorý sa v tejto množine nachádza, sa považuje za pravdivý (pri danej interpretácii). Ten, čo sa v danej množine nevyskytuje, je nepravdivý. Herbrandovský model nejakej množiny viet je taká herbrandovská intepretácia, pri ktorej je každá veta z danej množiny pravdivá.

Môžeme si už priblížiť (zatiaľ iba na intuitívnej úrovni) ďalší spôsob sémantickej špecifikácie operátora inferencie (inferenčného stroja). Kým v prípade relačných databáz sme použili špecifikáciu pomocou nejakého vzoru (relačnej algebry), teraz využijeme prístup na základe teórie modelov, na základe definície vyplývania (využívajúcej pojem modelu).

Budeme hovoriť, že základný atóm ϕ *logicky vyplýva* z nejakej množiny E datalogovských kláuz práve vtedy, keď je pravdivý v každom modeli E (značenie: $E \models \phi$). To znamená: ak sú v nejakej interpretácii pravdivé všetky vety z E, tak je v tejto interpretácii pravdivý aj základný atóm ϕ. Treba pripomenúť, že zatiaľ definujeme iba vzťah vyplpývania pre základné atómy: $\neg p \vee q$, ani $\neg q \rightarrow \neg p$ nevyplýva z $p \leftarrow q$ (v teraz zavedenom zmysle slova).

Pojem logického vyplývania (dedukcie) možno zovšeobecniť tak, aby sa nevzťahoval iba na základné atómy a datalogovskú syntax. Podstatné však zostáva: nejaký dôsledok logicky vyplýva (je dedukovateľný) z predpokladov, ak je pravdivý vo všetkých modeloch množiny predpokladov.

Špeciálny prípad operátora Cn, operátor deduktívneho uzáveru Cn_{ded} nad jazykom Hornovských kláuz budeme používať v tomto zmysle:

[26] ⟨ dodatky B a D ⟩

Definícia 2.42 Nech B_J je množina základných atómov jazyka J, E je množina hornovských kláuz v tomto jazyku.

Potom $Cn_{ded}(E) = \{\phi \in B_J : E \models \phi\}$. ■

Teda, špecifikovali sme Cn_{ded} sémanticky, a to pomocou pojmu logického vyplývania. Ak máme sémanticky definovaný ľubovoľný vzťah vyplývania Ω, môžeme si definovať korektnosť a úplnosť operátora odvodenia (vzhľadom na Ω). Základná intuícia: Odvodenie Cn je korektné, ak všetko, čo odvodíme z ľubovoľnej množiny viet, z nej vyplýva podľa Ω. Naopak, ak všetko, čo vyplýva z ľubovoľnej množiny viet (podľa Ω), dokážeme z nej odvodiť pomocou Cn, hovoríme, že odvodenie je úplné.

Definícia 2.43 Operátor odvodenia Cn je *korektný* vzhľadom na Ω práve vtedy, keď pre $\phi \in Cn(E)$ platí $E \; \Omega \; \phi$.

Cn je *úplný* vzhľadom na Ω práve vtedy, keď pre ľubovoľné E a ϕ, ktoré spĺňajú $E \; \Omega \; \phi$, platí $\phi \in Cn(E)$. ■

Teda, špecifikáciu operátora odvodenia Cn vzťahom vyplývania Ω možno chápať takto: Ak sme definovali Ω, cieľom je navrhnúť a realizovať Cn tak, aby bol korektný, prípadne úplný, vzhľadom na Ω.[27]

2.4 Komentáre

Herbrandovské interpretácie V celom tomto texte používame herbrandovské interpretácie alebo ich varianty. Iba niekedy naznačujeme všeobecnejší pojem modelu.

Používanie herbrandovských interpretácií a modelov je založené na nasledujúcej teoréme:

Tvrdenie 2.44 ([ChL 73]) *Množina kláuz S je nesplniteľná práve vtedy, keď S nie je splnená pre žiadnu herbrandovskú interpretáciu množiny S.*

Základná myšlienka dôkazu: Z nesplniteľnosti ľahko dostaneme nesplniteľnosť pre herbrandovské interpretácie. Naopak: ak by S bola nesplniteľná pri herbrandovských interpretáciách a splnená pri nejakej (všeobecnej) interpretácii $\mathcal{I}$, možno ľahko skonštruovať herbrandovskú interpretáciu, ktorá zodpovedá $\mathcal{I}$ a spĺňa S. ■

Uvedené tvrdenie teda umožňuje v mnohých kontextoch používať iba herbrandovské interpretácie. V teórii logického programovania a deduktívnych databáz je obmedzovanie sa na herbrandovské interpretácie bežné.

[27] Sémantická špecifikácia sa formuluje pomocou pojmov modelu (spĺňania, pravdivosti) a logického vyplývania. Implementácia operátora Cn (na základe danej špecifikácie) sa realizuje syntakticky a výpočtovo. Logická syntax sa konštruuje nad pojmami dobre utvorenej formuly, odvodenia, dôkazu. Príslušné výpočtové procedúry operujú nad týmito syntaktickým konštrukciami.

Implementácia operátora Cn_{ded} Operátor Cn_{ded} môžeme implementovať pomocou SLD-rezolvencie. Z teórie logického programovania je známe, že táto implementácia je korektná a úplná vzhľadom na danú špecifikáciu (pozri dodatok D alebo podrobne v [Llo 87]).

SLD-rezolvencia počíta od cieľa (od otázky, ktorú treba zodpovedať). Postupne sa dostáva k faktom, ktoré na zodpovedanie možno použiť. Miesto SLD-rezolvencie môžeme datalogovský výpočet (teda operátor Cn_{ded}) realizovať aj metódami výpočtu zdola nahor, od faktov k cieľu. Výpočty zhora nadol i zdola nahor majú svoje prednosti i nevýhody. Preto sa navrhujú a študujú metódy, ktoré kombinujú výpočet zdola nahor a zhora nadol. Tieto metódy sú z hľadiska databázových aplikácií výhodnejšie ako SLD-rezolvencia.

Predpokladajme, že špecifikáciou Cn_{ded} zamýšľame určiť, čo sú adekvátne odpovede na otázky adresované deduktívnej databáze. Potom pomocou správnej implementácie operátora Cn_{ded} získame ako odpovede na otázky presne tie základné atómy, ktoré z E logicky vyplývajú. A naopak, všetky základné atómy, ktoré logicky vyplývajú z E, získame pomocou takejto implementácie.

Efektívnosť Vyvažovanie vyjadrovacej sily a výpočtovej efektívnosti je jednou z kľúčových úloh pri reprezentácii poznatkov. Databázový kontext je zaujímavý aj z tohto hľadiska.

V relačných databázach sa používa jednoduchý, limitovaný jazyk na reprezentáciu poznatkov. Výhodou za toto „uskromnenie sa" je výpočtová efektívnosť inferencie nad vetami tohto jazyka.

V tejto kapitole sme si ukázali, ako možno zosilňovať vyjadrovaciu silu relačných databázových jazykov. Problému zosilnenia vyjadrovacích schopností relačného kalkulu sa venovali [ChH 85, ChH 82, ChH 80]. Ukázali, že jazyk, ktorý obohacuje relačný kalkul o operátor najmenšieho pevného bodu, umožňuje vyjadriť presne tie otázky ako jazyk hornovských kláuz (logického programovania). Toto zosilnenie jazyka je z výpočtového hľadiska slušné:

Tvrdenie 2.45 *[Imm 82] Predpokladajme (bez ujmy na všeobecnosti), že je daná databázová schéma, ktorej všetky atribúty nadobúdajú jediný typ hodnôt. Majme jazyk, ktorý zosilňuje relačný kalkul o operátor najmenšieho pevného bodu a o relačný symbol totálneho usporiadania na doméne.*

Potom všetky otázky, ktoré sú vyjadriteľné v tomto jazyku, sú vypočítateľné v polynomickom čase.

Dôsledok 2.46 ([Pap 93]) *Programy v jazyku logického programovania bez funkčných symbolov odpovedajú na otázky v polynomickom čase.*

Videli sme, že reprezentačný formalizmus, založený na hornovských klauzách má solídne výpočtové vlastnosti. Preto neprekvapuje nasledujúce zistenie. Kým vo všeobecnosti je problém splniteľnosti formúl výrokového počtu

NP-úplný, pre hornovské klauzy existuje lineárny algoritmus [DGa 84]. Podobne, úlohou SLD-rezolvencie je zefektívniť výpočet v porovnaní so všeobecnou rezolvenciou [GeN 87].

Za tento zisk z hľadiska výpočtovej efektívnosti sa platí nižšou odvodzovacou silou: videli sme už, že v jazyku hornovských kláuz neodvodíme všetky dôsledky, ktoré možno odvodiť v neobmedzenej predikátovej logike prvého rádu. Keďže odvodzovanie je súčasťou reprezentácie poznatkov, platíme nižšou vyjadrovacou silou reprezentačného formalizmu.

Polynomický algoritmus 2.47 preveruje splniteľnosť množiny hornovských výrokovologických kláuz (problém HSAT).[28] Tento algoritmus skúša konštruovať model množiny kláuz Φ. Ak je Φ splniteľná, podarí sa mu to, v opačnom prípade konštatuje nesplniteľnosť. Použitá technika: Štartuje sa z prázdnej interpretácie. Pridávajú sa všetky fakty z Φ, ak nejaké sú. Udržiavajú sa dve verzie interpretácie: T a pracovná verzia I. Postupne sa do T pridávajú hlavy kláuz nesplnených pri danom stave interpretácie I (to znamená, že telo klauzy je pri I splnené, ale hlava nie). Po pridaní týchto hláv do T sa môže stať, že v T budú splnené telá niektorých ďalších kláuz, ale nie ich hlavy. Preto sa v procese rozširovania interpretácie pokračuje dovtedy, kým sa nedosiahne pevný bod (kým nedosiahneme, že $I = T$). Vtedy sú všetky klauzy s neprázdnou hlavou splnené pri skonštruovanej interpretácii. Napokon sa preverí, či intepretácia T spĺňa klauzy s prázdnou hlavou: iba tie môžu byť príčinou nesplniteľnosti (čitateľ iste vidí súvislosť s integritnými obmedzeniami).

Algoritmus ukončí svoju činnosť: T rastie pri každom kroku, hornou medzou tohto rastu je (konečný) počet výrokových premenných vo Φ.

Korektnosť: T určite zaručuje splniteľnosť kláuz tvaru $y \leftarrow x_1, \ldots, x_n$. Okrem toho, pre každé $\leftarrow x_1 \wedge \cdots \wedge x_n$, teda pre $\neg x_1 \vee \cdots \vee \neg x_n$ z Φ platí, že keď táto klauza nie je splnená pri T, Φ nie je splniteľná: T je najmenšia[29] interpretácia, ktorá spĺňa všetky klauzy z Φ s neprázdnou hlavou. Nemôžeme preto vypustiť z T žiadnu z premenných x_i.

Zložitosť: uvedený algorimus je polynomický; lineárnosť sa dá dosiahnuť pri vhodných údajových štruktúrach.[30]

Zovšeobecnenie predstavy o deduktívnej databáze Lloyd v [Llo 87] zavádza všeobecnejšiu predstavu o deduktívnej databáze. Databázové vety sú tvaru $A \leftarrow W$, kde W je ľubovoľná formula predikátového počtu prvého rádu. Otázky a integritné obmedzenia sú tvaru $\leftarrow W$, pričom W má ten istý význam ako vyššie. Výpočtová procedúra spočíva v redukcii do tvaru hornovských kláuz a v následnom využití výpočtovej procedúry pre hornovské klauzy.

[28] Čitateľa, ktorý nie je zvyknutý čítať zápisy algoritmov, odkazujeme na dodatok E. Umiestnenie textov algoritmov ponechávam na LaTeX. Ten ich umiestni v prostredí zvanom tabuľka tak, aby boli súvisle čitateľné na jednej strane. Algoritmus 2.47 je v tabuľke 2.1.

[29] ⟨ dodatok D ⟩

[30] ⟨ zložitosť: dodatok H ⟩

Algoritmus 2.47
vstup: Φ – *množina hornovských kláuz (v jazyku výrokovej logiky) s konečným počtom premenných*
výstup: *rozhodnutie, či* Φ *je splniteľná*

begin
$T := \emptyset$
(T obsahuje zoznam premenných – je to zamýšľaný model množiny* Φ **)*
repeat
    $I := T$
    pre každú klauzu $y \leftarrow x_1, \ldots, x_n$ *z* Φ *nesplnenú pri* I
        $T := T \cup \{y\}$
until $I = T$
test, či všetky klauzy s prázdnou hlavou sú splnené pri T
end

Tabuľka 2.1: HSAT – splniteľnosť hornovských kláuz

Rozšírenie deduktívnych databáz o neúplnú informáciu možno nájsť napríklad v [Gel 94]. Hlavná myšlienka spočíva v zavedení modálnych operátorov do jazyka logického programovania.

V tejto kapitole sme nazerali na databázy z hľadiska reprezentácie znalostí. Ukázali sme si, že možno hovoriť o znalostiach, reprezentovaných v relačnej databáze. Vyjadrovacie schopnosti relačnej databázy sme rozšírili viacerými spôsobmi.

Najprv sme uvažovali o možnostiach vyjadriť neúplnú informáciu (a odvodzovať implicitnú informáciu z databázy s neúplnou informáciou). Potom sme sa zmienili o databáze s vetami rozmanitej hodnovernosti.

Ďalší smer, ako zaviesť silnejší jazyk a silnejšiu odvodzovaciu mašinu, poskytuje koncept deduktívnej databázy.

Kapitola 3

Databázy s hypotézami

V tejto kapitole zavŕšime postupné komplikovanie predstáv o znalostiach, ktoré sú explicitne alebo implicitne reprezentované v databáze. Jedným z cieľov kapitoly je pripraviť a zdôvodniť širšie poňatie implicitnosti: V nejakej množine viet sú v istom zmysle slova implicitné aj také znalosti, ktoré z nej nemožno vydedukovať. Existuje množstvo príkladov využívania tohto typu implicitnosti pri každodennom usudzovaní. Vďaka nemu nachádzame riešenia zaujímavých a dôležitých problémov, nemá však silu (nevyvrátiteľnosť) dedukcie. Je nepostrádateľné, keď nemáme k dispozícii úplné poznanie danej oblasti. Vtedy sme nútení prijímať dobre zdôvodnené hypotézy. Môžeme ich považovať za poznatky implicitné v neúplnom poznaní. V neúplnom poznaní sú implicitné isté alternatívy jeho zúplňovania.

Pre výpočtové modelovanie inteligencie je dôležité modelovať schopnosť prijímať hypotézy, ktoré zúplnia naše poznanie aspoň dočasne, schopnosť rozpoznať konflikt medzi prijatou hypotézou a ostatnými akceptovanými poznatkami a schopnosť riešiť tento konflikt. Preto si zavedieme druh odvodzovania, ktorý nazveme hypotetickým odvodzovaním. Ukážeme si jeho jednoduché príklady. Budeme analyzovať – na abstraktnej úrovni – jeho základné vlastnosti.

Tento druh usudzovania je charakteristický pre tzv. zdravý ľudský rozum, alebo common sense, ako sa hovorí v anglosaských krajinách a používa v odbornej terminológii. Výskum v oblasti hypotetického, na kontexte závislého usudzovania, odvodzovania záverov z neúplného poznania a za prítomnosti nekonzistentností sa považuje za kľúč k pochopeniu common sensu.

V nasledujúcich kapitolách sa potom budeme venovať vlastne iba hypotetickej inferencii a problémom s ňou súvisiacim.

Začneme intuíciami, príkladmi a základnými definíciami. Pred prvým príkladom žiadam čitateľa o veľkorysosť: bude prostoduchý, naznačí však dôležitú vec.

Príklad 3.1 Doplňme databázu z príkladov 2.32 a 2.35 o klauzu

$$manželia(X, Y) \leftarrow rodič(X, Z), rodič(Y, Z), muž(X), žena(Y).$$

Táto klauza umožňuje odvodiť, že anna a karol sú manželia. Rovnako sme však schopní odvodiť aj fakt, že anna a milan sú manželia. ■

Uvedená veta je príkladom všeobecného tvrdenia, ktoré je pravdivé až na nejaké výnimky. Napriek týmto výnimkám je účelné vety tohto typu používať, často fungujú spoľahlivo. Podobným správaním sa vyznačujú vety, ktoré zvykneme považovať za vieryhodné alebo približné.

Samozrejme, pojmy, ktoré sme práve použili (vieryhodnosť, približnosť, pravdivosť až na nejaké výnimky – a dali by sa pridávať ďalšie), sú veľmi intuitívne. Ďalej ich budeme upresňovať. Od viet však pozornosť presunieme k odvodzovaniu a k operátorom odvodenia Cn.

Doposiaľ sme si odvodzovanie reprezentovali abstraktným operátorom Cn. Každý takýto operátor je však nejakým spôsobom realizovaný, „implementovaný“. Zaužívaný spôsob jeho implementácie predstavujú odvodzovacie pravidlá.

Príklad 3.2 Príkladom takéhoto pravidla je *modus ponens* : ak platí veta ϕ a ak platí $\phi \rightarrow \psi$, potom môžeme odvodiť ψ. ■

Definícia 3.3 Nech X je nejaká konečná množina viet jazyka J a ϕ je veta jazyka J. Budeme hovoriť, že dvojica (X, ϕ) je *odvodzovacím pravidlom*[1] (nad jazykom J). X nazývame *predpokladmi* tohto odvodzovacieho pravidla a ϕ jeho *záverom*.

Pravidlá tvaru (X, ϕ) budeme obvykle zapisovať buď ako

$$\frac{X}{\phi}$$

alebo ako $X \ / \ \phi$. ■

Príklad 3.4 Modus ponens z príkladu 3.2 zapíšeme ako

$$\frac{\phi,\ \phi \rightarrow \psi}{\psi}$$

[1]Neskôr si pojem pravidla zovšeobecníme, teraz ešte nemáme pre to dostatočnú motiváciu.

Modus ponens zodpovedá tejto – vždy pravdivej – implikácii: $(\phi \wedge (\phi \rightarrow \psi)) \rightarrow \psi$. Nie je protiintuitívne, keď ku každej implikácii $\phi \rightarrow \psi$, ktorej veríme, skonštruujeme odvodzovacie pravidlo $\phi \,/ \psi$. Špeciálnym prípadom takýchto implikácií sú pravidlá, ktoré sme spoznali v kapitole 2, teda definitné klauzy. ■

Intuície o hypotetickom usudzovaní si ešte doplníme. Keďže sme si uvedomili, že každej implikácii môžeme priradiť odvodzovacie pravidlo, príklad 3.1 naznačuje, že je účelné používať pravidlá, ktoré zväčša dávajú pravdivé dôsledky z pravdivých predpokladov, niekedy nás však môžu sklamať: z pravdivých predpokladov dajú nepravdivý záver (teda, odvodenie nie je deduktívne).

Usudzovanie, riadené takýmto druhom pravidiel, môžeme nazvať *hypotetickým usudzovaním*. Závery, získané takýmto usudzovaním môžeme považovať za *hypotézy*.

Príklad 3.5 Implikácii z príkladu 3.1 zodpovedá nejaké odvodzovacie pravidlo. Toto pravidlo niekedy dáva z pravdivých predpokladov nepravdivý záver. Závery získané s jeho pomocou, napríklad $manželia(anna, karol)$, považujeme za hypotézy.

Treba dodať, že aj v prípade, keď pracujeme s oveľa rozumnejšou definíciou manželského vzťahu, môžeme sa dostať do úskalí hypotetického usudzovania. Preverovanie, či nejaká Anna a nejaký Karol sú manželia, môže spôsobovať vážne problémy (vrátane pátrania po matrikách tohto sveta). ■

Inferenčný stroj, ktorý má schopnosť odvodzovať (navrhovať, generovať) hypotézy, nazveme generátorom hypotéz. Databázu (množinu viet), nad ktorou je definovaný generátor hypotéz, budeme nazývať *databázou s hypotézami*. Pojmy z tohto odstavca onedlho upresníme.[2]

Je zrejmé, prečo nás zaujíma hypotetické usudzovanie a hypotézy v súvislosti s výpočtovým modelovaním inteligentného správania: Ambicióznejší model inteligentného správania, ktorý je založený na počítaní so znalosťami, nevystačí s dedukciou (tak, ako človek nevystačí s dedukciou). Pri odvodzovaní nemožno očakávať, že sú vždy k dispozícii všetky vety, ktoré umožňujú vydedukovať relevantné závery (závery, potrebné pre rozhodovanie, pre úspešné

[2] Uvedený prípad (výsledkom odvodenia je nejaká hypotéza) treba odlíšiť od nasledujúceho: Veľmi zaujímavá a dôležitá úloha je presvedčiť sa, k akým záverom vedie prijatie nejakých hypotéz. Tento typ problémov („what – if questions") je kľúčový pre analýzu trendov, simuláciu, predikciu. Samozrejme, pozornosť sa mu venuje aj z hľadiska informatického. Predmetom pozornosti je vkladanie hypotéz do databázy a štúdium ich dôsledkov. Samozrejme, toto vkladanie a odvodzovanie treba špecifikovať (a realizovať) tak, aby existujúcu databázu (faktov) nedeštruovalo. Pozri napríklad [Bon 97], staršia práca, venovaná problému je napríklad [GaR 84]. V [Bon 97] sa venuje pozornosť aj hypotetickému odstraňovaniu viet z databázy. Gabbay s Reylem upozorňujú na súvislosť s kontrafaktuálnym odvodzovaním: je to odvodzovanie z viet, o ktorých je známe, že nie sú pravdivé. Analýza takýchto viet je dôležitá pre pochopenie teoretického poznania. Kontrafaktuálne usudzovanie analyzoval z hľadiska umelej inteligencie [Gin 86].

akcie, pre riešenie nejakého problému). Systém, ktorého cieľom je pracovať s poznatkami, by nebol užitočný, keby prestal pracovať vo chvíli, keď nemá dostatok informácií. Znakom inteligencie je v takýchto situáciách vytvárať domnienky. Preto je nutné prijímať isté hypotézy, ktoré doplnia znalosti a „poznanie systému" – aspoň dočasne – zúplnia. Robia tak vedci, lekári, detektívi, robí tak ktorýkoľvek človek v bežnom živote. Zrejme by takúto vlastnosť mal mať aj ambicióznejší výpočtový model inteligencie.

Pochopiteľne, formulovanie hypotéz môže viesť k rozporom. Iba systém, ktorý si poradí s rozpormi, na ktoré narazil, dokáže riešiť netriviálne úlohy, náležiace do oblasti výpočtového modelovania inteligencie. Špeciálnym prípadom tohto problému je asimilácia nových poznatkov a adaptácia starých.

Intuitívne úvahy ukončíme tým, že sa pokúsime predstaviť inferenčný stroj, ktorý dokáže dedukovať, ale aj generovať hypotézy. Ak sa mu zo známych viet nedarí dedukovať relevantné závery, prichádza na rad generátor hypotéz. V priaznivom prípade sa mu môže podariť vygenerovať takú hypotézu, z ktorej by relevantný záver bol dedukovateľný.

Príklad 3.6 Predstavme si, že sme ráno bez problémov naštartovali auto. Jazdili sme s ním po veľmi hrboľatom teréne, odstavili sme ho asi na hodinu, reflektory sme nenechali rozsvietené. Potom sme už nedokázali naštartovať. Možno vygenerovať hypotézu, že počas jazdy sa uvoľnili kontakty batérie. Táto hypotéza je implicitná v poznatkoch o danej situácii. V komplikovanejších prípadoch sú možné alternatívne hypotézy. Teda priliehavá je predstava o alternatívnosti hypotetického usudzovania. ■

Niektoré hypotézy môžu byť natrvalo zaradené do bázy poznatkov E, iné možno po čase odmietnuť, ak do E prijmeme vety, ktorým budeme dôverovať viac a ktoré im odporujú. Samozrejme, je to veľmi subtílny a komplikovaný problém. O jeho jednoduchších aspektoch je vlastne takmer celý zvyšok tejto knihy.

Prv než prejdeme k známym jednoduchým generátorom hypotéz, ktoré sa študujú v umelej inteligencii, niekoľko prípravných definícií. Najprv si zavedieme pojem interpretácie (nie celkom štandardne). Pomerne pevné intuície, späté s týmto pojmom sme získali v kapitole 2, štandardnú definíciu (a súvisiace definície splňovania, pravdivosti, modelu) možno nájsť v dodatku B. V nasledujúcej definícii spomedzi všetkých základných atómov nejakého jazyka vyberieme dve množiny. Jedna z nich, T, bude predstavovať množinu všetkých pravdivých základných atómov. Druhá, F, množinu nepravdivých. Interpretácia určuje, čo budeme považovať za pravdivé a čo za nepravdivé.

Definícia 3.7 Nech $\mathcal{A}$ je množina základných atómov jazyka J. Nech *výber* je ľubovoľná podmnožina $\mathcal{A}$. *Interpretácia* je akákoľvek dvojica výberov (T, F). *3-interpretácia* je taká interpretácia, pre ktorú platí $T \cap F = \emptyset$. *2-interpretácia* je taká 3-interpretácia, pre ktorú $T \cup F = \mathcal{A}$. ■

Poznámka 3.8 Tu definovaná interpretácia určuje iba pravdivostnú hodnotu základných atómov. Na základe danej interpretácie základných atómov môžeme priradiť k ľubovoľnej vete (daného konkrétneho jazyka) nejakú pravdivostnú hodnotu, ak máme dobre definované logické spojky. ■

Keďže nemáme v tejto chvíli definovaný nejaký konkrétny jazyk, prijmeme iba konvenciu, ktorá nám dovolí pre existujúcu interpretáciu základných atómov vysloviť predpoklad, že určuje pravdivostnú hodnotu zložitejších viet predpokladaného jazyka:

Konvencia 3.9 Predpokladajme triedu jazykov, ktoré spĺňajú túto podmienku: Ak je daná interpretácia (T, F) jazyka J, môžeme špecifikovať nejakú množinu pravdivostných hodnôt a funkciu $\mathcal{I}_{(T,F)}$, ktorá každej vete jazyka J priradí nejakú pravdivostnú hodnotu.

Ďalej budeme predpokladať, že medzi pravdivostnými hodnotami sú také hodnoty ako pravda a nepravda. $\mathcal{I}_{(T,F)}$ umožňuje hovoriť, že nejaká formula je pravdivá, nepravdivá atď. v (T, F).

Intuitívne, vety sú také výrazy jazyka, ktoré môžu byť pravdivé. Interpretácia je zariadenie, ktoré umožňuje odlíšiť pravdivé a nepravdivé atómy. Na jej základe, pomocou funkcie $\mathcal{I}_{(T,F)}$ možno určiť pravdivostnú hodnotu viet. Najvšeobecnejšia z uvedených je štvorhodnotová interpretácia, ktorá nevylučuje, že niektoré vety (pri danej interpretácii) nie sú ani pravdivé ani nepravdivé a nevylučuje tiež, že existujú sporné vety, ktoré sú aj pravdivé aj nepravdivé. 3-interpretácia nepripúšťa sporné vety. A pre 2-interpretáciu je každá veta buď pravdivá alebo nepravdivá.

Definícia 3.10 Nech X je ľubovoľná množina viet. Hovoríme, že interpretácia (T, F) je *modelom* X, ak každé $\phi \in X$ je pravdivé v (T, F). ■

To, čo podľa tejto definície možno vysloviť o jednoprvkových množinách, budeme vyslovovať aj o ich prvkoch – teda budeme hovoriť aj o modeli vety. Teraz zavedieme veľmi liberálny pojem hypotézy a generátora hypotéz. Neskôr budeme hľadať nejaké podmienky, ktoré by rozumným spôsobom túto veľkú voľnosť obmedzovali.

Definícia 3.11 Odvodzovacie pravidlo nazveme *generátorom hypotéz*, ak existuje model jeho predpokladov, ktorý nie je modelom jeho záveru. Ľubovoľný záver generátora hypotéz nazývame *hypotézou.* ■

Tejto definícii vyhovujú aj veľmi zlé generátory hypotéz. Prijmeme však aj existenciu zlých generátorov hypotéz. Nemôžeme ju totiž vylúčiť. Neskôr sa budeme venovať niektorým lepším generátorom a budeme stručne informovať aj o výskumoch, ktorých cieľom je charakterizovať lepšie generátory pomocou nejakých abstraktných vlastností.

Teraz, po uvedení základných intuícií a definovaní východiskových pojmov, si zbežne priblížime niektoré elementárne typy inferencie, na ktoré môžeme nazerať ako na generátory hypotéz.

Začneme defaultami. V bežnom uvažovaní sa často riadime tým, že sa vieme orientovať v štandardných situáciách. Domnievame sa, že v prípade, keď sa nestane nič neobvyklého, platí to, čo sa štandardne očakáva. Pokiaľ nemáme dôvod predpokladať, že obvyklý stav a chod vecí sa niečím naruší, prijímame obvyklé riešenia. Takýto typ hypotetického usudzovania predstavujú aj defaultové pravidlá, pozri [Rei 80].

Príklad 3.12 Príklad 3.1 si v tvare defaultového pravidla môžeme vyjadriť takto:

$$\frac{rodi\check{c}(X,Z), rodi\check{c}(Y,Z), mu\check{z}(X), \check{z}ena(Y) : man\check{z}elia(X,Y)}{man\check{z}elia(X,Y)}.$$

■

Formuly, ktoré stoja pred dvojbodkou v riadku nad čiarou sa nazývajú predpokladmi pravidla. Strážami tie, čo sú za dvojbodkou.[3] Pod čiarou je záver. Pravidlo je pri odvodzovaní použiteľné vtedy, keď

- predpoklady pravidla sú splnené a
- nie je známe, že by stráže neboli splnené (nič nevylučuje platnosť stráží; je konzistentné predpokladať ich platnosť).[4]

Ak je pravidlo použiteľné, potom môžeme odvodiť zodpovedajúcu inštanciu záveru.

Príklad 3.13 Vráťme sa k príkladu 3.12: Pokiaľ sa nevyskytne nič, čo by vylučovalo, že X a Y sú manželia (hypotéza, že X a Y sú manželia nie je vyvrátená) a ak sú splnené predpoklady, môžeme prijať záver (hypotézu), že X a Y sú manželia.

Pokiaľ veta $man\check{z}elia(karol, zuza)$ z príkladu 3.1 je konzistentná so zvyškom databázy, odvodíme pomocou defaultového pravidla záver *manželia(karol, zuza)*. Ak by v modifikovanej databáze platilo $\neg man\check{z}elia(karol, zuza)$, potom už neodvodíme $man\check{z}elia(karol, zuza)$. ■

Usudzovanie pomocou defaultových pravidiel je klasickým príkladom tzv. nemonotónneho usudzovania. Môže sa totiž stať, že z nejakej množiny viet nedokážeme odvodiť to, čo z jej podmnožiny. Je ľahko možné, že nejaká množina

[3] Anglický termín je *justification*, odchýlili sme sa od neho.

[4] Obe podmienky v súčinnosti sú charakteristické pre spôsob, ako ľudia argumentujú – niektoré ich argumenty podopierajú závery usudzovania, iné argumenty môžu vyvracať tieto závery; ak vyvracajúce argumenty chýbajú alebo nie je nám známe, že by platili, chápeme to ako zdôvodnenie záveru.

viet E nevylučuje platnosť stráží defaultového pravidla, ale jej nadmnožina E' už stráže vyvracia.

Vo všeobecnosti, pri nemonotónnom usudzovaní sa môže stať, že nejaká množina viet E vytvára kontext, v ktorom je možné odvodiť nejakú množinu záverov (hypotéz). Avšak v kontexte E', kde $E \subset E'$, niektorá zo spomínaných hypotéz už nemusí byť prijateľná (odvoditeľná).

Definícia 3.14 Operátor Cn nazývame *nemonotónnym*, ak pre nejaké $E_1 \subseteq E_2$ platí, že $Cn(E_1) \not\subseteq Cn(E_2)$. ■

Odvodzovanie charakterizované nemonotónnym operátorom Cn, je nemonotónne.

Teraz sme už pripravení a motivovaní na zavedenie všeobecnejšieho pojmu pravidla.

Definícia 3.15 ([MNR 90]) Nech X a Y sú konečné množiny viet jazyka J a ϕ je veta jazyka J.

Odvodzovacie pravidlo je trojica (X, Y, ϕ). X nazývame *predpokladmi* tohto odvodzovacieho pravidla, Y *globálnymi strážami* a ϕ jeho *záverom*. ■

Aj pri tomto všeobecnejšom pojme ostáva v platnosti vlastnosť, ktorou definujeme generátor hypotéz: je ňou podmienka existencie modelu jeho predpokladov, ktorý nie je modelom záveru (hypotézy). Pojem pravidla z definície 3.3 je špeciálnym prípadom: predpokladá sa, že Y je prázdna množina.

Poznámka 3.16 Globálnosť stráží dobre ilustruje diskusia v príklade 3.13: Tam podmienkou prijatia hypotézy bolo, že sa nevyskytne nič, čo by vylučovalo, že *karol* a *zuza* sú manželia. Očividne, toto „nič, čo by vylučovalo" odkazuje na nejaký globálny kontext, na všetko, čo je v danej chvíli známe.

Čitateľ získal pri pohľade na defaultové pravidlo istú intuíciu o tom, ako sa uplatňujú globálne stráže pri odvodzovaní. ■

Prv než nadhliadneme ponad defaultové pravidlá, potrebujeme mať zavedený pojem konzistentnosti. Ten sa dá rozumne zaviesť vtedy, keď platí, že pravdivosť ϕ a $\neg\phi$ sa vylučuje. Interpretáciu, ktorá túto vlastnosť zabezpečuje, nazveme klasickou. Neskôr spoznáme aj interpretácie, ktoré nie sú klasické. Ďalej budeme predpokladať formuly bez voľných premenných, t.j. *uzavreté* formuly.[5]

Definícia 3.17 Interpretácia $\mathcal{I}$ jazyka J je *klasická*, ak pre každú uzavretú formulu ϕ z J platí: ϕ je pradivá v $\mathcal{I}$ práve vtedy, keď $\neg\phi$ je nepravdivá v $\mathcal{I}$. ■

[5]⟨ dodatok B ⟩.

Poznámka 3.18 2-interpretácia je nutne klasická, 3-interpretácia môže, ale nemusí byť klasická, naša štvorhodnotová interpretácia nebude klasická. Interpretácie, ktoré nie sú klasické, sa však v tomto texte budú vyskytovať a ich úloha nebude bezvýznamná. Už v časti 3.2 budeme diskutovať alternatívnu cestu k zavedeniu hypotetického usudzovania. Použitie neklasických interpretácií bude pritom kľúčové. Preto sme predpoklad o klasickosti interpretácie formulovali explicitne. ■

Definícia 3.19 Nech X je množina viet jazyka J. Ak existuje klasická interpretácia (T, F), ktorá je modelom X, potom hovoríme, že X je konzistentná. ■

Globálny aspekt defaultových pravidiel znamená, že stráž pravidla ostáva v poriadku (ostáva nenarušená) dovtedy, kým jej popretie nevyplýva z nejakého globálneho kontextu (databázy, bázy znalostí). Nakoľko vzťah logického vyplývania[6] nie je rozhodnuteľný, defaultové pravidlá a ich praktická použiteľnosť budia vážne pochybnosti.[7]

To vrhá tieň na možnosti rozumnej formalizácie nemonotónneho usudzovania.

Predstava hypotetickej databázy nie je až natoľko bláznivá, ako by sa mohlo zdať z predchádzajúceho výkladu. S formalizmom logického programovania (alebo datalogovských databáz), je tiež spriahnuté nemonotónne usudzovanie (veľmi jednoduchého typu).

Doposiaľ sme sa vyhýbali zavedeniu negácie do databáz a datalogu. Teraz sa pozastavme pri probléme negácie. V databázach sa negatívne fakty nereprezentujú priamo. Ide vlastne o ekonómiu myslenia a vyjadrovania: Vždy je tých ľudí, čo nie sú zamestnacami nejakého konkrétneho podniku oveľa viac ako tých, čo sú (atď). Preto sú v databáze negatívne fakty reprezentované implicitne. Dohodou, ktorá toto postihuje, je predpoklad uzavretého sveta (Closed World Assumption, CWA).

Definícia 3.20 (Predpoklad uzavretého sveta) Pre databázu D je

$$CWA(D) = \{\neg\phi | D \not\models \phi\},$$

kde ϕ je základný atóm (jazyka databázy D). ■

CWA si možno predstaviť aj ako odvodzovacie pravidlo: ak $D \not\models \phi$, potom akceptuj $\neg\phi$.[8]

[6] A teda ani konzistentnosti, lebo $M \models \phi$ práve vtedy, keď $M \cup \{\neg\phi\}$ je nekonzistentné.

[7] ⟨ rozhodnuteľnosť : dodatok H ⟩

[8] Aby nevznikol dojem, že prijímanie (generovanie) hypotéz sa tu chápe ako nejaká silná mystická vlastnosť „inteligentného systému“, treba pripomenúť, že CWA vlastne tiež vedie k prijímaniu istých hypotéz: hypoteticky prijímame platnosť akýchsi negácií dovtedy, kým nebudú vyvrátené nejakými pozitívnymi zisteniami. Teda, naša predstava o generovaní hypotéz pripúšťa čo najjednoduchšie prípady. CWA je „generátor hypotéz“. V istom zmysle slova zúplňuje reprezentáciu poznatkov.

CWA je teda pre databázy nástroj reprezentácie negatívnych faktov. Ani CWA však nie je výpočtovo realizovateľný – už pred chvíľou sme zdôrazňovali, že vzťah vyplývania, použitý v definícii, je nerozhodnuteľný.

Preto v logickom programovaní sa miesto CWA používa negácia ako konečné zlyhanie: ak v konečnom čase zlyhá dôkaz faktu p, prijmeme $\neg p$.[9] Datalogovské databázy s negáciou ako konečným zlyhaním vlastne nie sú deduktívne:

Príklad 3.21 Deduktívna databáza obohatená o negáciu ako konečné zlyhanie je hypotetická databáza. Z databázy $\{p(a) \leftarrow, q(b) \leftarrow\}$ odvodíme $\neg p(b)$, hoci z nej logicky nevyplýva. ■

Predpoklad uzavretého sveta, resp. negácia ako konečné zlyhanie sú vhodným nástrojom na reprezentáciu znalostí. Pritom oba vedú k nemonotónnemu odvodzovaniu. (Keď do databázy z príkladu 3.21 doplníme $p(b)$, viac už neodvodíme $\neg p(b)$.) Môže vzniknúť otázka, či táto nemonotónnosť nie je náhodná a závadná.

Ukážeme si však, že zavedenie negácie do definitných logických programov (a teda aj do databáz) nutne vedie k nemonotónnosti. Odvodíme z toho niektoré dôsledky pre nemonotónnu inferenciu a tým aj pre reprezentáciu znalostí.

Skôr však preložíme do reči operátorov odvodenia niektoré pojmy, ktoré sme zaviedli pre odvodzovacie pravidlá. (Aby sme mohli pokračovať na tej úrovni abstrakcie, ktorou sme začali.)

Doposiaľ sme hovorili o pravidlách, ktoré sú generátormi hypotéz. Teraz tento prívlastok prenesieme aj na operátory odvodenia. Keď hovoríme o operátoroch odvodenia a študujeme ich vlastnosti, nadväzujeme na tradíciu, ktorú v klasickej logike založil Tarski. Alternatívny prístup, venovaný štúdiu odvodzovacích pravidiel v klasickej logike, sa spája s Gentzenom.

Definícia 3.22 Operátor odvodenia Cn je *korektný*, ak pre každú množinu viet E a pre každý model M množiny E platí, že M je modelom $Cn(E)$.

Operátor odvodenia Cn_{hyp} je *generátor hypotéz*, ak nie je korektný. Budeme hovoriť aj, že je *nie-korektný*. ■

Nebude nás však zaujímať ľubovoľný nie-korektný operátor odvodenia. Budeme sa zaujímať o nie-korektné operátory so „slušnejším správaním“. Pokúsime sa takéto správanie charakterizovať a operátory s takýmto správaním budeme považovať za dôveryhodnejšie generátory hypotéz.

[9] V tomto kontexte použitý symbol negácie nie je zhodný s negáciou klasicky študovanou vo výrokovej logike. K sémantickej charakterizácii negácie v logickom programovaní sa dostaneme v kapitole 6.

3.1 Charakterizácia nemonotónneho odvodzovania

Najprv definície niekoľkých dôležitých pojmov. Naším cieľom je nejako rozumne obmedziť nemonotónne operátory odvodenia.[10] Ľubovoľný nemonotónny operátor nie je príliš užitočný a zaujímavý. Predmetom výskumu sú práve tie nemonotónne relácie odvodenia, ktoré spĺňajú isté slušné vlastnosti.

Pripomeňme si, že operátor Cn je reflexívny, ak platí $X \subseteq Cn(X)$ pre každú množinu viet X.

Definícia 3.23 Nech E je ľubovoľná konzistentná množina formúl, ϕ je ľubovoľná konzistentná formula a operátor Cn je reflexívny.

Potom hovoríme, že Cn je *slabo korektný*, ak pre $\phi \in Cn(E)$ je $E \cup \{\phi\}$ konzistentná.

Cn zachováva konzistentnosť, ak je $Cn(E)$ konzistentná. ■

Ak Cn zachováva konzistentnosť, potom je slabo korektný. Opačný vzťah však neplatí. Splnená lokálna vlastnosť (slabá korektnosť) nezaručuje splnenie globálnej vlastnosti (zachovávanie konzistentnosti).

Príklad 3.24 Príklad Cn, ktoré je slabo korektné, ale nezachováva konzistentosť: Nech J je jazyk výrokovej logiky, E je konzistentná množina formúl z J, ϕ je literál. Cn definujme takto: $\phi \in Cn(E)$ práve vtedy, keď žiadna formula z E neobsahuje výrokovú premennú z ϕ. Cn je očividne slabo korektný, môže však odvodiť atóm aj jeho negáciu. ■

Ešte potrebujeme zadefinovať a na príklade ozrejmiť jeden technický pojem:

Definícia 3.25 Nech E je množina definitných kláuz. Budeme hovoriť, že *herbrandovská báza* množiny E je množina všetkých základných atómov tvaru $p(t_1, \dots, t_k)$, kde p je predikátový symbol vyskytujúci sa v E a t_i je základný term z E. Budeme ju značiť ako B_E. ■

Príklad 3.26 Nech $E = \{p(a) \leftarrow; q(b) \leftarrow\}$. Potom

$$B_E = \{p(a), p(b), q(a), q(b)\}.$$

Ak $E = \{nat(0) \leftarrow; nat(s(X)) \leftarrow nat(X)\}$, potom

$$B_E = \{nat(0), nat(s(0)), nat(s(s(0))), \dots\}.$$

■

[10]Táto a nasledujúca sekcia je založená na [Sef 91].

Poznámka 3.27 Treba upozorniť na rozdiel medzi množinou B_J, zavedenou v kapitole 2 a herbrandovskou bázou. Ak E, datalogovská databáza nad jazykom J, obsahuje všetky predikátové symboly, všetky funkčné symboly, všetky konštanty jazyka J, potom $B_E = B_J$. Vo všeobecnom prípade však iba časť jazyka J je použitá na zaznamenanie viet z E, vtedy $B_E \subset B_J$, teda herbrandovská báza je vlastnou podmnožinou B_J. V logickom programovaní sa J zvykne určiť na základe toho, čo je použité na zápis E, preto $B_J = B_E$. ■

Teraz môžeme vysloviť a dokázať zaujímavé tvrdenie s vážnymi dôsledkami.

Tvrdenie 3.28 ([Apt 88]) *Predpokladajme, že Cn je reflexívny operátor odvodenia. Nech ďalej E je množina definitných kláuz a $\phi \in Cn(E)$, kde ϕ je negácia nejakého základného atómu z B_E. Potom*

- *Cn nie je korektný,*
- *ak je Cn slabo korektný, potom je nemonotónny.*

Dôkaz :
a) B_E je modelom E. V B_E však nie je splnená žiadna negácia základného literálu z B_E. To znamená, že Cn nie je korektný.
b) Nech ϕ je $\neg\psi$ a Cn je monotónny. Z predpokladu $\neg\psi \in Cn(E)$ vyplýva, že $\neg\psi \in Cn(E \cup \{\psi\})$. Teda Cn nie je slabo korektný. ■

Poznámka 3.29 Každá množina definitných kláuz je konzistentná. Neplatí to o datalogovských databázach, pokiaľ obsahujú integritné obmedzenia – ak sú integritné obmedzenia v danej datalogovskej databáze narušené, databáza nemá model, preto nie je konzistentná.

Integritné obmedzenia neplnia pri dedukcii nijakú úlohu, nemožno ich použiť pri žiadnom odvodzovacom kroku. Slúžia iba na kontrolu konzistentnosti danej množiny datalogovských databázových viet. ■

Videli sme, že nemonotónnosť *CWA* nie je náhodná. Nemožno ju obísť, ak sa pohybujeme v jazyku definitných kláuz a chceme zaviesť negáciu.

Ľahko možno zovšeobecniť: nemonotónnosti sa nevyhneme vtedy, keď pracujeme s nie-korektnými inferenčnými pravidlami (teda, s generátormi hypotéz) a usilujeme sa zachovať konzistentnosť. Abstrahujme však od existencie inferenčných pravidiel a zaujímajme sa iba o správanie operátora Cn.

Tvrdenie 3.30 *Uvažujme jazyk J a jeho klasické interpretácie. Nech Cn je definovaný na množinách viet jazyka J a je reflexívny.*

Ak Cn nie je korektný, ale zachováva konzistentnosť, potom je nemonotónny.

Dôkaz: Nech E je konzistentná množina predpokladov. Na základe nie-korektnosti Cn môžeme predpokladať, že pre nejaké $\phi \in Cn(E)$ existuje model M množiny formúl E, v ktorom ϕ nie je splnená. Potom $\neg\phi$ je splnená v klasickej interpretácii M. Teda $E \cup \{\neg\phi\}$ je konzistentná. Z reflexívnosti Cn dostávame $\neg\phi \in Cn(E \cup \{\neg\phi\})$. Nakoľko Cn zachováva konzistentnosť, $\phi \notin Cn(E \cup \{\neg\phi\})$. Teda Cn je nemonotónny. ■

Na tomto zistení nie je nič prekvapujúceho. Opustenie podstatnej vlastnosti dedukcie, korektnosti, vedie k nemonotónnosti. V tomto zmysle slova všetky druhy nespoľahlivej inferencie, tradične študované (napríklad indukcia, analógia) sú nemonotónne.

Znenie tvrdenia naznačuje, že nie-korektnosť odvodzovacích pravidiel však sama o sebe k nemonotónnosti nestačí. Ukazuje to aj nasledujúci klasický príklad.

Príklad 3.31 Po tisícke pozorovaní len bielych labutí máme v každodennom naivnom uvažovaní dôvod prijať záver, že každá labuť je biela. Predpokladajme, že máme operátor Cn, pomocou ktorého tento záver dostaneme a predpokladajme aj, že Cn má všeobecne známu vlastnosť, umožňujúcu zo sporu odvodiť čokoľvek. Keď zaregistrujeme čiernu labuť a na základe istých terminologických konvencií prijmeme do našej teórie tvrdenie o existencii labute, ktorá nie je biela, nič nás (okrem zdravého rozumu) nenúti k nemonotónnosti: množinou dôsledkov našej teórie by mohli byť všetky výroky jazyka. ■

Keď analyzujeme uvedený príklad, zistíme, že nemonotónne odvodzovanie dostaneme až vtedy, keď sa usilujeme ošetrovať nekonzistentnosti. V terminológii nášho tvrdenia 3.30: aby operátor Cn bol nemonotónny, k nie-korektnosti Cn musí pristúpiť podmienka zachovávania konzistentosti. (V takom prípade sa zriekneme dôsledku "každá labuť je biela", ktorý sme odvodili z tisícich pozorovaní, ale už ho nechceme odvodiť z tisícjeden pozorovaní, lebo je nekonzistentný so zistením existencie labute, ktorá nie je biela. Dôvod, prečo sa zriecť práve tohto tvrdenia a nie nového pozorovania, je zrejmý, ale leží mimo našej hlavnej pozornosti.)

Druhá možnosť, ako riešiť problém naznačený v príklade 3.31, je zriecť sa logiky, pomocou ktorej zo sporu možno odvodiť čokoľvek. Túto vlastnosť majú tzv. parakonzistentné logiky, onedlho sa im budeme venovať.

Príklad 3.32 Zachovávanie konzistentnosti nie je nutnou podmienkou nemonotónnosti. Ukážeme si príklad operátora odvodenia, ktorý nezachováva konzistentnosť a súčasne je nemonotónny. Opustíme jazyk hornovských kláuz. Majme explicitnú databázu $E = \{p \vee q\}$ a prijmime CWA. Zrejme, $CWA(E) = \{\neg p, \neg q\}$.

Ľahko vidno, že $CWA(E)$ vedie spolu s E k nekonzistentnosti, ak predpokladáme klasickú dedukciu. Popri $\neg p$ odvodíme p a popri $\neg q$ odvodíme q. Pri $E' = \{p \vee q, q\}$ je $CWA(E') = \{\neg p\}$. Nemonotónne sa teda môže správať aj taký operátor odvodenia, ktorý nezachováva konzistentnosť. ■

CWA nevedie k nekonzistentnosti, ak sa obmedzíme iba na databázy definitných kláuz.

Tvrdenie 3.33 *Ak D je množina definitných kláuz, potom $D \cup CWA(D)$ je konzistentná množina viet.*

Dôkaz priamo z definície. ■

Tento fakt má všeobecnejší význam: Reštrikciou jazyka sa môžeme dostať k rozumným vlastnostiam daného Cn. Vhodnosť nemonotónnych inferenčných pravidiel závisí aj od jazyka, v ktorom sú reprezentované vety explicitnej databázy. Je užitočné, ak dokážeme rozumne ohraničiť jazyk, v ktorom reprezentujeme znalosti tak, že máme pri danom Cn automaticky zaručené zachovávanie konzistentosti.

Pozorný čitateľ pravdepodobne zbadal, že nemonotónna inferencia súvisí s revíziami. Predpokladajme množiny viet A, B také, že $A \subseteq B$, pričom $Cn(A) \not\subseteq Cn(B)$. To znamená, že existuje ϕ také, že $\phi \in Cn(A)$, ale $\phi \notin Cn(B)$. Okolnosť, že ϕ nie sme schopní odvodiť po rozšírení A na B, môžeme interpretovať ako (vynútenú) revíziu. Túto intiuíciu si teraz vyslovíme v presných pojmoch.

Najprv si však musíme uvedomiť komplikácie, spôsobené potrebou revidovať. Revízie sa stávajú zaujímavým a zložitým problémom až kvôli vzťahom medzi vetami v báze znalostí. Komplikácie presvedčivo ilustruje nasledujúci jednoduchý problém:

Príklad 3.34 Ak máme výrokovologickú bázu dát

$$p \wedge q \to r$$
$$p$$
$$q$$

a vložíme do nej $\neg r$, stáva sa nekonzistentnou. Ak sa s tým nechceme zmieriť, mali by sme situáciu napraviť nejakou modifikáciou. Máme však viac možností, ako to urobiť. ■

Teda, vo všeobecnosti ťažko očakávať, že k danej množine viet existuje presne jediná revidovaná množina viet, ktorá nás uspokojí.

Nech S je množina všetkých viet jazyka J a 2^S nech je množina všetkých podmnožín S.

Definícia 3.35 *Relácia revízie* je ternárna relácia $\rho \subseteq 2^S \times 2^S \times 2^S$ taká, že pre každé $(A, B, C) \in \rho$ platí $A, C \neq \emptyset \neq B \setminus A$, pričom existuje $\phi \in A$ také, že $\phi \notin C$. ■

Intuícia, ktorá je v pozadí za touto definíciou: k množine formúl A pridáme B, nejakú množinu formúl, ktorá obsahuje v porovnaní s A niečo nového. Výsledkom tohto pridania je množina C, ktorá neobsahuje nejaké vety z A.

Samozrejme, relácia revízie je z hľadiska uskutočnenia revízie nedostatočne určitá. Ideálne je, keď pre konkrétnu aplikáciu máme k dispozícii funkciu (operátor) revízie. Jednoducho, budeme predpokladať existenciu nejakej funkcie, ktorú sme získali istou selekciou z relácie. Ak máme pre dané A, B viac C takých, že trojice (A, B, C) patria do relácie revízie ρ, vždy z nich vyberieme jediné C. Budeme teda predpokladať existenciu čiastočného operátora revízie $R : 2^S \times 2^S \longrightarrow 2^S$. R je definovaný pre dvojice množín formúl A a B takých, že $A \neq \emptyset \neq B \setminus Cn(A)$. Vtedy pre $R(A, B) = C$ $(C \neq \emptyset)$ existuje $\phi \in A$ také, že $\phi \notin C$. Plne definovaný operátor revízie môžeme zaviesť pomocou nejakého špeciálneho symbolu, ktorý je hodnotou pre všetky ostatné prípady. Podrobnejšiu informáciu o konštrukcii funkcie revízie poskytuje [GRo 95].

Dostávame sa k formálnemu vyjadreniu toho, čo intuitívne pociťujeme: nemonotónne odvodzovanie nejak súvisí s revíziami.

Definícia 3.36 Nech Cn je nemonotónny operátor odvodenia.

Potom *operátor revízie* R nazývame *asociovaným* s nemonotónnym operátorom odvodenia Cn, ak pre všetky množiny viet A, B také, že $A \subseteq B, Cn(A) \not\subseteq Cn(B)$, $B \setminus Cn(A) \neq \emptyset$ platí $R(Cn(A), B) = Cn(B)$. ■

Operátor revízie, zavedený v tejto definícii vlastne simuluje správanie nemonotónneho operátora Cn pre „tie body“, v ktorých je Cn nemonotónny.

Ukážeme si, že k dôležitej podtriede nemonotónnych operátorov odvodenia zaručene existuje operátor revízie s ním asociovaný.

Podmienky $A \subseteq B$, $Cn(A) \not\subseteq Cn(B)$ a $B \setminus Cn(A) \neq \emptyset$ nemusia byť vo všeobecnosti súčasne splnené. To znamená, že podmienky, ktoré má spĺňať operátor revízie nie sú automaticky garantované. Preto si zadefinujeme vlastnosť operátora odvodenia, ktorá zaručí súčasné splnenie týchto podmienok. Táto vlastnosť sa nazýva kumulatívnosťou.

Definícia 3.37 ([Mak 89]) Cn je *kumulatívny*, ak pri predpokladoch $A \subseteq B \subseteq Cn(A)$ platí $Cn(A) = Cn(B)$. ■

Opakovaná aplikácia Cn na množinu viet B neprinesie nič nového, pretože B obsahuje iba Cn-dôsledky množiny viet A a všetky vety z A patria do B. Takto definovaný Cn možno považovať za slušný operátor odvodenia.

Fakt 3.38 *Nech Cn je nemonotónny a kumulatívny operátor odvodenia a pre $A \neq B$ nech platí $A \subseteq B \wedge Cn(A) \not\subseteq Cn(B)$. Potom $Cn(A) \neq \emptyset \neq B \setminus Cn(A)$.*

Dôkaz: Prázdne $Cn(A)$ je podmnožinou každej množiny. Teda $Cn(A) \neq \emptyset$. Z kumulatívnosti dostávame $Cn(A) \neq Cn(B) \Rightarrow (A \not\subseteq B \vee B \not\subseteq Cn(A)$. Predpokladali sme $Cn(A) \not\subseteq Cn(B)$ a $A \subseteq B$. Teda $Cn(A) \neq Cn(B)$ je splnené a platí $B \not\subseteq Cn(A)$. To znamená, že $B \setminus Cn(A) \neq \emptyset$. ■

Dôsledok 3.39 *Ak Cn je nemonotónny a kumulatívny, potom existuje operátor revízie R asociovaný s Cn.*

Dôkaz: Na základe faktu 3.38 sú splnené podmienky definície 3.36, teda môžeme definovať $R(Cn(A), B)$ ako $Cn(B)$. ■.

3.2 Odvodzovanie a nekonzistentnosť

Jedným zo základných predpokladov predchádzajúcej sekcie (ale vlastne aj celého doterajšieho výkladu) je, že pracujeme s klasickými interpretáciami. Teda, veta ϕ je splnená v interpretácii I práve vtedy, keď $\neg\phi$ v nej nie je splnená. Tento predpoklad opustíme.

Podobne, doteraz sme (mlčky) predpokladali, že operátor Cn je *triviálny* v nasledovnom zmysle slova: pre každú množinu viet A, ktorá obsahuje ϕ aj $\neg\phi$, platí $Cn(A) = S$. S sa nazýva aj *triviálna teória*. Pripomeňme, že S je množina všetkých viet nášho jazyka J.[11]

Operátor Cn, ktorý je triviálny a nezachováva konzistentnosť, sa chová defektným spôsobom: dokáže z konzistentnej množiny formúl vyprodukovať triviálnu teóriu.

Toto defektné správanie možno opraviť buď tým, že berieme do úvahy iba operátory Cn, ktoré zachovávajú konzistentnosť (videli sme, že v prípadoch, keď nie sú monotónne a sú kumulatívne, je s nimi asociovaný nejaký operátor revízie) alebo operátory, ktoré nie sú triviálne.

Revízia je vo všeobecnosti drahá operácia. Pozrime sa preto na druhý možný prístup, na používanie operátora inferencie Cn, ktorý nie je triviálny. Takýto operátor umožňuje pokračovať v odvodzovaní aj za prítomnosti nekonzistentností. Z praktického hľadiska ho môžeme považovať za opatrnú inferenciu kombinovanú s odkladaním revízií na neskôr.

Najprv stručná charakteristika formalizácie [Bel 76], ktorá túto vlastnosť realizovala ako jedna z prvých:

Každej atomickej formule sa priradí ako hodnota nejaká podmnožina priestoru $\{$*truth, false*$\}$, ide teda o množinu hodnôt $\{\{t\}, \{f\}, \{t, f\}, \emptyset\}$. Prvky tejto množiny podmnožín môžeme označiť ako t, f, c (contradictory), u (undefined). Konfúzia z nerozlišovania t a $\{t\}$ nehrozí – všetky pravdivostné hodnoty sú množiny.

Na týchto hodnotách zavedieme čiastočné usporiadanie takto: $u \leq t \leq c$ a $u \leq f \leq c$.

---

[11] Na prípade triviálneho Cn si môžeme priblížiť dôležitosť dvoch rovín pri verifikovaní nejakej špecifikácie. Ak máme špecifikovaný triviálny Cn, môžeme

- jednak preverovať, či jeho implementácia zodpovedá špecifikácii,
- alebo skúmať, či daná špecifikácia zodpovedá požiadavkám modelovania inteligentného správania.

Z hľadiska druhého problému možno konštatovať, že triviálnosť treba v mnohých kontextoch hodnotiť ako neintuitívnu a nevhodnú vlastnosť. Inteligentný človek z nekonzistentných predpokladov odvodzuje – aspoň nejaké – relevantné závery (a rozhodne neakceptuje všetky vety jazyka).

Logické spojky možno definovať viacerými spôsobmi. Pre konjunkciu a disjunkciu si demonštrujeme jeden z nich (nie je to spôsob, ktorý použil [Bel 76]). Hodnota konjunkcie (disjunkcie) $p \wedge q$ (resp. $p \vee q$) je infimum (supremum) hodnôt p, q.[12] Uvedená možnosť slúži iba ako ilustrácia, nižšie sa k problému interpretácie zložitejších formúl ešte vrátime. Opustíme aj historický exkurz a začneme sémantickú špecifikáciu netriviálneho Cn.

Obmedzíme sa na výrokovologický jazyk J. Atomárnymi výrokmi jazyka J budú symboly ako p, q, p_i, ale aj $maco(brum\_brum)$ (aby sme v kapitole 4.2 mohli prirodzenejšie zapisovať niektoré príklady). Budeme používať obvyklé výrokovologické spojky. Vety v explicitnej báze znalostí E sú vetami jazyka J a B_J je množina všetkých základných atómov jazyka J. Nad týmto jazykom budeme predpokladať množinu interpretácií, teda dvojíc (T, F), kde $T, F \subseteq B_J$. Zrejme nebudeme vyžadovať ani $T \cap F = \emptyset$, ani $T \cup F = B_J$.

Intuitívne, atómy z T sa považujú za pravdivé, atómy z F za nepravdivé. Prikročíme k definícii funkcie val_Q, ktorá priraďuje klauzám alebo množinám kláuz podmnožiny priestoru $\{t, f\}$.

Definícia 3.40 Nech $Q = (T, F)$ je interpretácia.

- Ak A je atóm, potom
  - $t \in val_Q(A)$ práve vtedy, keď $A \in T$,
  - $f \in val_Q(A)$ práve vtedy, keď $A \in F$.
- Ak A je $\neg B$, kde B je atóm, potom
  - $t \in val_Q(A)$ práve vtedy, keď $B \in F$,
  - $f \in val_Q(A)$ práve vtedy, keď $B \in T$.

■

Definícia 3.41 Ak A je $B \vee C$, potom

- $t \in val_Q(A)$ práve vtedy, keď $t \in val_Q(B)$ alebo $t \in val_Q(C)$,
- $f \in val_Q(A)$ práve vtedy, keď $f \in val_Q(B)$ a zároveň $f \in val_Q(C)$.

Nech X je množina kláuz. Potom:

- $t \in val_Q(X)$ práve vtedy, keď $(\forall \phi \in X)\ t \in val_Q(\phi)$,
- $f \in val_Q(X)$ práve vtedy, keď $(\exists \phi \in X)\ f \in val_Q(\phi)$.

■

[12] Infimum hodnôt p, q je ich najväčšie dolné ohraničenie, ich supremom je ich najmenšie horné ohraničenie, pozri aj $\langle$ dodatok E $\rangle$.

Alternatívne, ale ekvivalentne: namiesto atómov z F berieme do úvahy zodpovedajúce negatívne literály a interpretáciu budeme reprezentovať jedinou množinou literálov. Teda, napríklad interpretáciu $(\{p\}, \{q\})$ môžeme zapísať aj ako $(p, \neg q)$. Formálnejšie: Ak máme interpretáciu (T, F), budeme ju zapisovať aj ako $\mathcal{I} = T \cup \{\neg\phi : \phi \in F\}$. V tejto časti, ale i na iných miestach, budeme používať tento zápis. V prípade 2-interpretácie vystačíme s množinou T pozitívnych literálov, lebo F možno vypočítať z T. Pre 2-interpretácie totiž platí $F = S \setminus T$.

Príklad 3.42 Ak $Q = (p, \neg p)$, potom $val_Q(p) = c$. Pripomeňme, že $(p, \neg p)$ je iba skratka za (T, F), kde $T = (p)$ a $F = (p)$. „Vypočítame“ c: $t \in val_Q(p)$, pretože $p \in T$ a súčasne $f \in val_Q(p)$, lebo $p \in F$. ■

Definícia 3.43 Z množiny kláuz E *(kvázi)vyplýva* klauza ϕ, ak pre každú interpretáciu Q, v ktorej $t \in val_Q(E)$, platí, že $t \in val_Q(\phi)$ (značenie: $E \models_{QRI} \phi$).

Nech E je množina kláuz. Potom interpretácia $\mathcal{Q}$ je *(kvázi)model* množiny E, ak pre všetky klauzy ϕ z E platí, že $t \in val_Q(\phi)$. ■

Pomocou „kvázi“ chceme vyjadriť, že ide o veľmi neštandardné vyplývanie. Ďalej však budeme hovoriť jednoducho iba o vyplývaní. Podobne pre „kvázimodel“.

Teraz môžeme ukázať charakteristické črty zavedenej sémantiky.

Príklad 3.44 Kontradiktorické (v klasickom zmysle slova) množiny kláuz majú model: Napríklad $(p, \neg p, \neg q)$ je modelom $\{p, q \vee \neg p, \neg q\}$. Funkcia val priraďuje pri tejto interpretácii hodnotu $c = \{t, f\}$ klauzám p a $q \vee \neg p$. Hodnotou klauzy $\neg q$ pri tejto interpretácii je $\{t\}$. Ak by sme chceli formálne zadefinovať množinu kláuz kontradiktorickú pri interpretácii $\mathcal{Q}$, požadovali by sme, aby aspoň jedna klauza z tejto množiny mala pri intepretácii $\mathcal{Q}$ hodnotu c. ■

Príklad 3.45 Z $E = \{p, \neg p\}$ nevyplýva q: interpretácia $(p, \neg p)$ je modelom E, ale nie je modelom q. Teda, zadefinované (kvázi-)vyplývanie nie je triviálne. ■

Uvedená sémantika umožňuje budovať logiku, ktorá implementuje netriviálny operátor Cn. Nazvime si ju QRI, podľa jej (trochu odlišného) vzoru RI [KLo 89]. Pre RI existuje aj dôkazová procedúra. Počítanie v QRI nás bude zaujímať iba pre špeciálny prípad (ďalej v časti 4.2).

Na záver tejto časti ukážeme, ako sa na báze QRI opäť dostávame k nemonotónnej inferencii.

Najprv si ukážeme, že vyplývanie v QRI má nepríjemnú vadu.

Príklad 3.46 Z $E = \{p, \neg p \vee q\}$ nevyplýva q. Doložiť to možno opäť interpretáciou $(p, \neg p)$. To znamená, že modus ponens neplatí pre naše (kvázi)vyplývanie. ■

Nižšie definovaný pojem minimálneho vyplývania[13] zachováva platnosť modus ponens. Neskôr však uvidíme, že to platí iba v niektorých kontextoch.

Definícia 3.47 Dvojica základných literálov $\{\phi, \neg\phi\}$ je *konfliktný pár* interpretácie $\mathcal{Q} = (T, F)$, ak $\phi \in T$ a $\phi \in F$. ■

Na interpretáciach možno definovať tranzitívnu reláciu $\prec$:

Definícia 3.48 Nech $\mathcal{Q}_1, \mathcal{Q}_2$ sú interpretácie. *Stupeň* interpretácie $\mathcal{Q}$ je počet konfliktných párov v $\mathcal{Q}$ (kardinalita množiny $T \cap F$). Potom $\mathcal{Q}_1 \prec \mathcal{Q}_2$, ak stupeň $\mathcal{Q}_1$ je menší ako stupeň $\mathcal{Q}_2$.

$\mathcal{Q}$ je *najmenej konfliktná* (najkonzistentnejšia) interpretácia v množine Π interpretácií, ak neexistuje $\mathcal{Q}'$ v Π také, že $\mathcal{Q}' \prec \mathcal{Q}$.

Z množiny kláuz E *minimálne vyplýva* klauza ϕ (značenie: $E \models_{min} \phi$) práve vtedy, keď ϕ je pravdivé v každom najmenej konfliktnom modeli E. ■

Príklad 3.49 Z $\{\neg p \vee q, p\}$ minimálne vyplýva q, pretože (p, q) je jediný najmenej konfliktný model $\{\neg p \vee q, p\}$. ■

Fakt 3.50 *Minimálne vyplývanie $\models_{min}$ je nemonotónne.*

Dôkaz: Nech $E = \{\neg p \vee q, p\}$. Platí, že $E \models_{min} q$, ale $E \cup \{\neg q\} \not\models_{min} q$. Dva najmenej konfliktné modely $E \cup \{\neg q\}$ sú $(p, \neg p, \neg q)$ a $(p, q, \neg q)$, v prvom z nich q nie je pravdivé. ■

Poznámka 3.51 Opäť vidíme, že nemonotónna inferencia je závislá na kontexte. (Dá sa povedať, že v niektorých kontextoch je modus ponens použiteľný, v iných nie.) ■

Fakt 3.52 *Ak $E \models_{min} q$ a $\mathcal{Q}$ je najmenej konfliktný model E, potom je $\mathcal{Q}$ aj najmenej konfliktný model $E \cup \{q\}$.*

3.3 Počítanie hypotéz

Doposiaľ sme hovorili o generovaní hypotéz dosť abstraktne a s ľahkou bezstarostnosťou. Výpočtové aspekty sme si všímali málo. Vytýčili sme si však cieľ analyzovať hypotetické usudzovanie tak, aby táto analýza bola užitočná pre výpočtové modelovanie inteligencie. Treba teda už v tejto chvíli aspoň naznačiť, čo obnáša počítanie hypotéz.

V počiatkoch umelej inteligencie sa veľká pozornosť venovala klasickej logike a automatickému dokazovaniu teorém. Časom sa však zistilo, že to nie sú adekvátne nástroje na modelovanie inteligencie. Nie sú ani efektívne

[13] Dôležitá poznámka: Minimálne vyplývanie sa považuje za vhodný všeobecný prostriedok sémantickej charakterizácie nemonotónnej inferencie.

z výpočtového hľadiska. Preto sa ozvali hlasy, ktoré preferovali iný druh odvodzovania: využíva defaulty, je nemonotónny, umožňuje pracovať s neúplnou a nekonzistentnou informáciou, dovoľuje rýchle skoky k nejakým predbežným záverom. Mohli by sme povedať, že volali po rýchlom hypotetickom usudzovaní.

Žiaľ, analýza tohto druhu usudzovania ukazuje, že nádeje neboli celkom opodstatnené. V zásade možno hovoriť o dvoch hlavných príčinách výpočtovej náročnosti modelov hypotetického usudzovania:

1. rešpektovanie nejakého globálneho kontextu a práca s ním,
2. preferovanie istých informácií pred inými.

Globálny kontext je dominantný pre hypotetické usudzovanie už na intuitívnej úrovni. Inteligentný a kritický človek prijíma nejakú domnienku vtedy, keď dobre zapadá do sústavy jeho poznatkov. Keď si uvedomuje, že v jeho poznatkoch sú nejaké medzery alebo rozpory, usiluje sa „poupratovať" vo svojich poznatkoch a prijať domnienky, ktoré vytvorený poriadok umožňujú alebo podporujú.

Niektoré príklady formalizácií hypotetického usudzovania robia to isté. Predpoklad uzavretého sveta prijíma také negatívne literály, ktoré nemožno vyvrátiť na základe daného stavu globálnej databázy. Defaultové pravidlo nie je aplikovateľné, ak v danom globálnom kontexte možno vyvrátiť jeho stráže. Formalizácia defaultových teórií predstavuje tento globálny kontext ako nejakú množinu viet $\mathcal{E}$ (uzavretú na dedukciu). $\mathcal{E}$ obsahuje dôsledok každého defaultového pravidla, aplikovateľného na $\mathcal{E}$. Teda – treba odhadnúť nejaký globálny kontext, nejakú množinu viet, presvedčení, poznatkov tak, aby táto množina bola nasýtená a koherentná. Nasýtená v tom zmysle slova, že aplikácia všetkých defaultových pravidiel nič nezmení na $\mathcal{E}$. Pridanie nového pravidla alebo odobratie niektorého z pravidiel by ju mohlo „deštruovať" (rozšíriť alebo zúžiť), odstrániť „poriadok", ktorý ona zavádza. Pod koherentnosťou si možno predstaviť vlastnosť dovoľujúcu byť presvedčeným o platnosti všetkých viet z danej množiny, o tom, že predstavuje „dobre poupratované poznatky". Technické detaily si priblížime v časti 4.1.

Zo sémantického hľadiska je kľúčovým pre pochopenie takýchto konštrukcií pojem pevného bodu nejakého operátora. Ak na množine viet definujeme nejaký operátor Φ, množina viet S je nasýtená, ak je pevným bodom tohto operátora, to znamená $\Phi(S) = S$. Koherentná je vtedy, keď operátor má rozumné vlastnosti.

Z výpočtového hľadiska je problém nájdenia takejto množiny viet veľmi ťažký. Po prvé, treba z veľkého množstva potenciálnych riešení (spomedzi všetkých podmnožín množiny viet formulovateľných v nejakom jazyku) vybrať jednu alebo niekoľko vhodných (spravidla neexistuje konštruktívna metóda na ich hľadanie). Po druhé, treba preveriť ich koherentnosť. V existujúcich formalizáciách to znamená preverenie konzistentnosti (alebo vyplývania), čo sú z výpočtového hľadiska opäť veľmi ťažké problémy. V časti 4.3 budeme venovať pomerne mnoho priestoru výpočtovým aspektom defaultových teórií. Aj na iných miestach sa dotkneme problémov vypočítateľnosti a výpočtovej zložitosti v súvislosti s „počítaním hypotéz“.

Preferovanie niektorých informácií je zdôvodniteľné tým, že spomedzi všetkých mysliteľných hypotéz (ktorých je nesmierne mnoho) si ceníme iba relevantné. Zavedenie preferencií má byť nástrojom na špecifikovanie relevantnosti. V časti 3.2 sme používali vzťah vyplývania $\models_{min}$, ktorý pri práci s nekonzistentnými modelmi umožňoval preferovať tie minimálne nekonzistentné. Sémantická špecifikácia preferencie sa obvykle zakladá na nejakej relácii definovanej na triede modelov. Opäť ide o ťažký výpočtový problém. Spomedzi všetkých modelov treba nájsť minimálne (najlepšie vzhľadom na danú preferenciu) a v minimálnych preverovať nejakú podmienku.

I ďalšie typy výpočtov, súvisiacich s generovaním hypotéz sú náročné: revízie, generalizácie, vysvetľovanie atď.

Hlavný dôraz tohto textu sa kladie na sémantickú špecifikáciu rôznych odvodzovacích operátorov, realizujúcich hypotetické usudzovanie. Problémy súvisiace s vypočítateľnosťou však nepodceňujeme. Dodatok H o vypočítateľnosti a výpočtovej zložitosti svojím rozsahom podstatne presahuje ostatné dodatky. Jedným z jeho cieľov je načrtnúť výpočtové problémy, charakteristické pre výpočtové modelovanie inteligencie. Možno ho považovať aj za komplement tejto časti kapitoly o databázach s hypotézami.

Papadimitriou v [Pap 93] zdôrazňuje výpočtovú náročnosť problémov umelej inteligencie a logiky. Výpočtová zložitosť podľa neho hovorí o probléme, o tom, ako je daná oblasť aplikácií ťažká z hľadiska konceptuálneho. Ukazuje, že je nedostatočne štrukturovaná. Zložitosť problému sa nezmenší, keď na jeho riešenie použijeme iný nástroj. Situáciu môžeme zjednodušiť tak, že zjednodušíme problém alebo ho budeme riešiť tak, že „spustíme“ z nárokov, t.j. uspokojíme sa s nejakými aproximáciami, nevyžadujeme nájdenie všetkých riešení atď. Každopádne, nazeranie na problémy hypotetického usudzovania optikou výpočtovej náročnosti prispieva k lepšiemu pochopeniu tohto usudzovania.

3.4 Komentáre

Zaviedli sme jednoduchú definíciu hypotetického usudzovania, uviedli sme si jeho príklady a vyšetrovali sme niektoré jeho základné vlastnosti. Teraz sa pokúsime tento úvod do hypotetického usudzovania uzavrieť.

Na príklade logického programovania a databáz sme si ukázali, že existujú jazyky, ktorých (prirodzené) rozšírenie vyžaduje zavedenie nemonotónneho operátora Cn. Alebo inak: v niektorých reprezentáciach znalostí sa nemonotónnosti Cn nemožno vyhnúť. Logické programovanie poskytuje pre štúdium a pochopenie nemonotónneho odvodzovania pevné intuície a solídny formálny aparát. Navyše, v tomto formálnom aparáte je priamo zahrnutý výpočet. Popri sémantickej špecifikácii reprezentačných formalizmov budeme venovať pozornosť aj možnostiam efektívnej vypočítateľnosti.

Videli sme, že pre nemonotónny Cn sú charakteristické:

- nie-korektnosť,
- zachovávanie konzistenosti,
- revízie.

Alternatívou k zachovávaniu konzistentnosti je nie-triviálnosť (operátora Cn), t.j. vlastnosť, ktorá nedovoľuje z nekonzistentnosti odvodiť všetky vety jazyka. Ukázali sme si, ako možno nad sémantikou nie-triviálneho Cn zaviesť istý prípad minimálneho vyplývania, ktoré je nemonotónne.

Náš prístup k štúdiu nemonotónnosti je charakteristický tým, že sa usilujeme hľadať základnejšie vlastnosti, na ktorých je nemonotónnosť založená.

Medzi základné vlastnosti výpočtových modelov inteligencie by mala patriť schopnosť generovať hypotézy. Nech Cn_{FOL} je operátor odvodenia predikátovej logiky prvého rádu. Predpokladajme, že pre ľubovoľnú konzistentnú množinu predpokladov E platí $Cn_{FOL}(E) \subseteq Cn(E)$. Množinu vygenerovaných hypotéz reprezentuje rozdiel $Cn(E) \setminus Cn_{FOL}(E)$. Samozrejme, dá sa predstaviť generátor hypotéz, ktorý neodvodí všetky deduktívne dôsledky každej množiny predpokladov.

Generovanie hypotéz niekedy vynucuje revízie dovtedy akceptovaných poznatkov. Obe tieto dominanty bežného ľudského usudzovania a získavania poznatkov hovoria o jedinej veci – naše poznanie je dynamické, jeho dynamika je ovplyvňovaná získavaním nových faktov, prijímaním nových hypotéz na základe týchto faktov, a (niekedy) vynúteným revidovaním predchádzajúcich presvedčení na základe nových poznatkov. Hypotetické a nemonotónne usudzovanie, revízie sú iba symptómami dynamiky a evolutívnosti poznania.

Problematika hypotetického usudzovania reflektuje fakt, že naše poznanie je aproximatívne, že sa dopúšťame chýb a omylov, že k našim zisteniam nachádzame výnimky, že naše poznatky sú vieryhodné v rôznom stupni. Štúdium týchto problémov je zaujímavé a dôležité. Na druhej strane, je komplikované a v súčasnom stave neposkytuje také odpovede, ktoré by boli dostatočné (najmä nie z hľadiska výpočtového modelovania).

Naša pozornosť bude zameraná práve na problémy hypotetického usudzovania, na odvodzovanie vieryhodných záverov z neúplnej informácie, na vieryhodné zúplňovanie neúplnej inofmácie, na narábanie s nekonzistentnou informáciou a s výnimkami. Vo všeobecnosti medzi kľúčové vlastnosti reprezentácie znalostí (a inferencie, ktorá je jej podstatným aspektom) patria neúplnosť, nekonzistentnosť, nemonotónnosť, nie-korektnosť, neurčitosť, predbežnosť, revidovateľnosť, aproximatívnosť, vágnosť, dynamickosť a závislosť na kontexte. Niektorým z týchto (a súvisiacich) problémov budeme venovať väčšiu pozornosť, niektorým menšiu, než by si zasluhovali.

V tejto kapitole sme si vytvorili istú predstavu o hypotetickej databáze.

Odvodzovacia mašinéria, pracujúca s takouto databázou, má schopnosť generovať hypotézy a revidovať stav databázy v prípade, že niektoré z hypotéz boli vyvrátené (nejakým spôsobom sa vysporiadať so vzniknutou nekonzistentnosťou). Venovali sme sa dôležitej vlastnosti hypotetického odvodzovania, nemonotónnosti. Ukázali sme, že hypotetické odvodzovanie zachovávajúce konzistentnosť nutne vedie k nemonotónnosti. Ďalej sme ukázali, že s nemonotónnym odvodzovaním je nutne spriahnuté revidovanie znalostí. Hľadali sme aj alternatívnu cestu: ak hypotetické odvodzovanie si nekladie za cieľ zachovávať konzistentnosť, možno odvodzovať takým spôsobom, že množina dôsledkov sa nezvrhne, nebude totožná s množinou všetkých viet jazyka.

Kapitola 4

Typické prípady a hierarchie

V kapitole 3 sme zaviedli pojem hypotetického usudzovania. Oboznámili sme sa s jeho jednoduchými príkladmi. Analyzovali sme niekoľko jeho nutných alebo vítaných vlastností (nie-korektnosť, nemonotónnosť, zachovávanie konzistentosti, súvislosť s revíziami, odvodzovanie za prítomnosti nekonzistentnosti).

Teraz, v dvoch kapitolách za sebou, je naším cieľom venovať sa niektorým pomerne dobre analyzovaným prípadom hypotetického usudzovania. V tejto kapitole si podrobnejšie predstavíme defaultovú inferenciu a dedenie vlastností v hierarchických sieťach. V jej poslednej časti sa budeme venovať výpočtovým aspektom oboch spomínaných typov odvodzovania.

Hypotetické usudzovanie sa bežne používa v každodenných situáciách. Obvykle sa pohybujeme v dobre známom prostredí a v ňom vieme spoľahlivo odhadovať, čo nás očakáva. V inej terminológii: Prijímame hypotézy o typických udalostiach a situáciách. Často potrebujeme urobiť si poriadok v našich domnienkach: ak narábame s väčším počtom hypotéz, niekoré z nich sa môžu vzájomne vylučovať, preto je dôležité pýtať sa, ktoré z nich dávajú zmysel dohromady. V tejto kapitole si všimneme podrobnejšie tento druh usudzovania.

Ďalší druh bežne používaného hypotetického usudzovania je založený na tom, že poznatky máme organizované hierarchicky. Hierarchie, ktoré vytvárame, však pripúšťajú výnimky a vyskytujú sa v nich konflikty. Aj tu je dôležitou úlohou „upratovanie“ v našich domnienkach.

*Schopnosti, používané v oboch spomínaných druhoch usudzovania sa považujú za základnú výbavu tzv. zdravého sedliackeho rozumu (*common sense*). Analýza a výpočtové modelovanie common sensu sú jednou z centrálnych výskumných tém v oblasti reprezentácie znalostí.*

Podrobnejšie ako v iných kapitolách sa budeme venovať výpočtovým aspektom: niektorým implementáciám, algoritmom a ich zložitostným charakteristikám. I keď hlavným objektom pozornosti v tejto knihe je sémantická špecifikácia jazykov na reprezentáciu poznatkov a najmä odvodzovacích strojov, dôležité je aj poznať, ako náročné výpočty sú takýmto spôsobom špecifikované. Práve na príklade defaultových teórií a hierarchických sietí podrobnejšie ilustrujeme tento aspekt. Táto časť kapitoly má skôr charakter príručky ako systematickej konštrukcie. Napriek tomu verím, že čitateľ môže získať základnú intuíciu o zložitosti výpočtov, potrebných pre nemonotónne odvodzovanie.

4.1 Defaultové teórie

Človek pri usudzovaní úspešne používa poznatky, ktoré platia v „typickom prípade", „skoro vždy", „až na nejaké výnimky". Hypotézy, ktoré generuje defaultová inferencia, vyjadrujú naše očakávania o obvyklom, typickom stave vecí.

Dôležitá otázka je, ako charakterizovať a identifikovať zodpovedajúci očakávaný stav vecí. Odpoveď poskytuje pojem extenzie defaultovej teórie.

Za zaujímavý moment tejto kapitoly možno považovať rozšírenie konceptu inferenčného operátora, ktorý bude umožňovať rôzne, vzájomne sa vylučujúce, alternatívne cesty odvodzovania a alternatívne predstavy o očakávanom stave vecí.

Defaultové teórie sú veľmi jednoduchým prípadom databáz s hypotetickým usudzovaním.[1] Pojem defaultového pravidla už poznáme z kapitoly 3. Intuitívne, defaultová teória je množina axióm obohatená o množinu defaultových pravidiel. Dôsledky sa odvodzujú pomocou defaultových pravidiel, ale aj pomocou štandardných pravidiel dedukcie.

Najprv si uvedieme príklady defaultových pravidiel. Potom sa budeme zaoberať ich naivným používaním. Prejdeme k intuíciám o sofistikovanejšom prístupe k nim. Po takejto motivačnej príprave sa budeme venovať formalizáciám defaultového usudzovania.

Príklady defaultových pravidiel Prvý príklad naznačí, ako defaultové pravidlá prispievajú k zúplneniu poznania.

Príklad 4.1 Pripomeňme si príklad 3.21. Majme databázu $\{p(a) \leftarrow, q(b) \leftarrow\}$ a predpokladajme, že nás zaujíma iba tá časť sveta, ktorú vieme popísať pomocou p, q, a, b. Je tu niekoľko možných a vylučujúcich sa totálnych zúplnení obrazu sveta:

- $\neg p(b), \neg q(a)$,
- $p(b), q(a)$,
- $\neg p(b), q(a)$,
- $p(b), \neg q(a)$.

Samozrejme, zúplnenia nemusia byť totálne. Môžeme napríklad prijať $\neg p(b)$, ale nechať otvorené, či platí $q(a)$ alebo $\neg q(a)$.

[1] Konštrukcia defaultových teórií a prvý výskum ich vlastností pochádzajú od [Rei 80].

Sformulujme si defaultové pravidlá, ktoré umožnia „vygenerovať" uvedené zúplnenia:

$$\frac{:p(b)}{p(b)}, \qquad \frac{:\neg p(b)}{\neg p(b)}, \qquad \frac{:q(a)}{q(a)}, \qquad \frac{:\neg q(a)}{\neg q(a)}\ .$$

Prvé z nich umožňuje odvodiť $p(b)$, ak neplatí $\neg p(b)$. Analogicky pre ostatné pravidlá.

Presnú predstavu o zúplnení poskytuje pojem extenzie. Ten si zavedieme neskôr. V tejto chvíli si ho – pri istom zjednodušení – môžeme ilustrovať na našom príklade: *Každé zo zúplnení spolu s uvedenou databázou tvorí jeden možný variant našich presvedčení*[2] o takomto jednoduchom svete. Neskôr si zavedieme pojem extenzie, ktorý dá presný zmysel spomínaným *variantom našich presvedčení*. Samotné defaultové teórie nemajú prostriedok, ako vyberať medzi rôznymi extenziami (zúplneniami). Každá extenzia je rovnako dobrá. ■

Dôležité je dodať, že defaultové pravidlá sú doménovo–špecifické. Na rozdiel od nich, pravidlá, s ktorými sa obvykle stretáme, sú nezávislé od akejkoľvek domény. Neobsahujú žiadne konštanty a sú aplikovateľné vždy, keď sú splnené nejaké štrukturálne podmienky. Napríklad, modus ponens môžeme aplikovať na nejakú množinu predpokladov vždy, keď sa v nej vyskytuje nejaká implikácia $\phi \to \psi$ i antecedent tejto implikácie ϕ, pričom ϕ a ψ môžu byť ľubovoľné vety. Ani CWA nie je doménovo-špecifické pravidlo (i keď sa podstatne – nevypočítateľnosťou, nelokálnosťou, neštrukturálnosťou,[3] odkazom na globálny kontext – líši od modus ponens). Môžeme ho aplikovať na ľubovoľný základný atóm ϕ, ktorý nevyplýva z databázy. V kontraste k tomu, doménovo–špecifické pravidlá sú aplikovateľné vtedy, keď sa v množine predpokladov vyskytujú nejaké konkrétne vety (a v prípade defaultových pravidiel sa aj špecifikuje, ktoré ďalšie konkrétne vety sa v nej nesmú vyskytovať).

Príklad 4.1 ukazuje, že s defaultovými pravidlami môžeme spájať túto ideu: Ak nám nejaká databáza (alebo teória) poskytuje neúplný obraz danej oblasti záujmu a potrebujeme zúplnenie tohto obrazu, nájdime pravidlá, ktoré by nás rozumne riadili pri zúplňovaní.

Defaultové pravidlá možno vnímať aj ako spôsob zakódovania viet, ktorých platnosť je závislá na nejakých predpokladoch a strážená splnením nejakých globálnych podmienok. Teda niečo ako podmienečne platné vety. Pritom podmienky sú dvojakého druhu: tie, čo musia byť splnené a tie, čo nesmú byť vyvrátené.

[2] Termínom presvedčenie sa pokúšam vyjadriť to, čo sa v anglickej literatúre vyjadruje ako „belief". Možno by bol vhodnejší termín predpoklady, alebo vieryhodné predpoklady, či zdôvodnená viera. Terminologickými záležitosťami sa však nebudeme príliš zaťažovať.

[3] Pod lokálnosťou sa myslí to, že v každom mieste odvodzovania je zrejmé, ktoré odvodzovacie pravidlo možno použiť a možnosť jeho použitia je rozpoznateľná lokálne – iba na základe analýzy tohto miesta. Pod štrukturálnosťou sa myslí to, že možnosť jeho použitia je rozpoznateľná výlučne na základe štruktúry použitých výrazov. Na príklade modus ponens si možno dobre ilustrovať obe vlastnosti.

Samozrejme, nie každé zúplnenie je rozumné. Vyššie uvedené pravidlá predstavujú skôr formálne cvičenie, než podstatnú črtu defaultových pravidiel. Defaultové pravidlá zúplňujú tak, aby to zodpovedalo obvyklému stavu vecí. Keď nie je známe, že by obvyklý stav vecí bol narušený, predpokladáme, že stav je presne taký ako obvykle.

Nasledujúce príklady lepšie charakterizujú motivácie, s ktorými boli defaultové pravidlá zavedené. Ich cieľom je reprezentácia spôsobov uvažovania podľa tzv. common sensu. Podrobnejšiu prezentáciu možno nájsť v [Luk 90].

Príklad 4.2 (Typické prípady) Základná situácia, pre ktorú je defaultové usudzovanie zamýšľané, sú typické (prototypové) prípady. Každý vie, že mačky sú v typickom prípade ostražité. Doslovne podľa pravidla: ak je niečo mačka a nie sú argumenty proti tomu, že by bola ostražitá, tak prijmime, že je ostražitá.

$$\frac{ma\check{c}ka(X) : ostra\check{z}it\acute{a}(X)}{ostra\check{z}it\acute{a}(X)}$$

■

Príklad 4.3 (Obvyklý stav vecí) Kým v príklade 4.2 sa vymedzila nejaká množina a charakterizovala sa jej typická črta, v tomto príklade ilustrujeme pravidlá charakterizujúce obvyklé udalosti.

Napríklad: obvykle mávame seminár v utorok, výnimku tvoria iba tie prípady, keď sa vopred oznámi, že seminár v utorok nebude.

$$\frac{: semin\acute{a}r\_v\_utorok}{semin\acute{a}r\_v\_utorok}$$

■

Príklad 4.4 (Prezumpcia neviny) Obvinený je považovaný za vinného až vtedy, keď sa jeho nevina vyvráti (dovtedy, kým jeho nevinu možno konzistentne predpokladať, ho treba považovať za nevinného).

$$\frac{obvinen\acute{y}(X) : nevinn\acute{y}(X)}{nevinn\acute{y}(X)}$$

■

Príklad 4.5 (Dôsledky neoboznámenosti) Ak neviem o tom, že by som mal sestru, mám vážny dôvod veriť tomu, že sestru nemám. (Niekedy je dôvodom pre prijatie nejakého záveru neinformovanosť o tom, že by platil jeho opak, ak sa pohybujeme v doménach, ktoré dobre poznáme.)

$$\frac{: \neg m\acute{a}m\_sestru}{\neg m\acute{a}m\_sestru}$$

■

Pochopiteľne, odvodzovanie s použitím defaultových pravidiel nie je korektné a monotónne, videli sme to už v kapitole 3. Doposiaľ sme si ešte neupresnili spôsoby manipulácie s defaultovými pravidlami, preto si čisto intuitívne demonštrujeme ich nie-korektnosť a nemonotónnosť.

Príklad 4.6 Uvažujme defaultové pravidlo

$$\frac{: p \vee q}{p}.$$

Predstavme si, že $E = \{\neg p, q\}$. Úlohou je odvodzovať dôsledky z E pomocou uvedeného defaultového pravidla. Naše pravidlo je aplikovateľné, pretože $p \vee q$ nie je vyvrátené v E (a predpokladová časť pravidla je prázdna, preto žiadne predpoklady netreba splňovať). Teda odvodíme p. Dvojhodnotová interpretácia $I = (\{q\}, \{p\})$ ukazuje, že toto pravidlo nie je korektné. Dôsledok odvodenia, p, je nepravdivý v interpretácii I. Predpoklady, z ktorých sa odvodzovalo, t.j. E, sú pri I pravdivé a stráž nie je vyvrátená ($\neg p \wedge \neg q$ nie je v I pravdivé). Pravidlo je pre E a I aplikovateľné, ale jeho záver je v I nepravdivý, teda je nie-korektné. ■

Nemonotónnosť defaultových pravidiel ľahko demonštruje nasledujúci príklad.

Príklad 4.7 Pravidlo

$$\frac{: p}{p}$$

umožňuje odvodiť p z prázdnej množiny predpokladov, neumožňuje ho však odvodiť z $E = \{\neg p\}$. ■

Naivné používanie defaultových pravidiel Uvedené príklady ukázali, že odvodzovanie pomocou defaultových pravidiel je nie-korektné a nemonotónne. Teraz si všimneme, že nekontrolované odvodzovanie môže viesť k nepríjemným záverom.

Naivne sa defaulty používajú asi týmto spôsobom: Nie je známa (teda ani odvoditeľná) hodnota atribútu A pre objekt o. Máme pritom vedomosti, aká býva táto hodnota obvykle – prijmeme preto túto obvyklú hodnotu, default d. Zodpovedajúce defaultové pravidlo by mohlo vyzerať takto:

$$\frac{: A(o) = d}{A(o) = d}.$$

Teraz upresníme predstavu o odvodzovaní pomocou defaultových pravidiel. Potom ukážeme, že nekontrolované odvodzovanie môže viesť k nepríjemným záverom.

Definícia 4.8 Majme množinu E formúl prvorádového jazyka J, α, β, γ sú formuly z J. Nech D je množina defaultových pravidiel tvaru

$$\frac{\alpha : \beta}{\gamma}.$$

Pravidlo $d \in D$ je *aplikovateľné* pri E, ak $E \models \alpha$, $E \not\models \neg\beta$.

Odvodenie z E pomocou D je postupnosť formúl $\langle \phi_0, \phi_1, \ldots, \phi_k \rangle$ taká, že

- existuje $d \in D$ aplikovateľné pri E a jeho dôsledkom je ϕ_0,
- pre každé i: existuje $d \in D$ aplikovateľné pri $E \cup \{\phi_0, \phi_1, \ldots, \phi_{i-1}\}$, jeho dôsledkom je ϕ_i.

■

Podľa tejto definície odvodenie pozostáva z dôsledkov defaultových pravidiel, ktoré sú aplikovateľné na deduktívny uzáver[4] viet z E doplnených o vety prv odvodené pomocou defaultových pravidiel. Teraz môžeme analyzovať nasledujúci príklad.

Príklad 4.9 Platí, že obvykle manželia bývajú spolu. Podobne platí, že človek obvykle býva v mieste svojho pracoviska. Formalizujme si to nasledujúcimi defaultovými pravidlami, kde *bydlisko* a *sídlo* sú funkčné symboly.

(i) $$\frac{man\check{z}elia(X,Y), bydlisko(X) = Z : bydlisko(Y) = Z}{bydlisko(Y) = Z},$$

(ii) $$\frac{zamestnan\acute{y}\_v(X,Y), s\acute{\imath}dlo(Y) = Z : bydlisko(X) = Z}{bydlisko(X) = Z}.$$

Predpokladajme teraz databázu $E = \{$

$$\begin{gathered} man\check{z}elia(eva, fero), \\ bydlisko(eva) = trnava, \\ zamestnan\acute{y}\_v(fero, divadlo\_andreja\_bagara), \\ s\acute{\imath}dlo(divadlo\_andreja\_bagara) = nitra\}. \end{gathered}$$

Odvodením z E pomocou D je aj postupnosť

$$\langle bydlisko(fero) = trnava, bydlisko(fero) = nitra \rangle.$$

Ak sa čitateľovi zdá tento príklad príliš lacný, napríklad vie si predstaviť pravidlá hry, podľa ktorých defaultové pravidlo (ii) nie je aplikovateľné pri

$$E \cup \{bydlisko(fero) = trnava\},$$

doplníme ďalšie defaultové pravidlo:

[4] Definícia aplikovateľnosti je volená tak, aby naozaj išlo o deduktívny uzáver.

(iii) $\dfrac{: bydlisko(X) \neq nitra}{\neg chodieva\_na\_Zobor(X)}.$

V takom prípade, odvodením z E pomocou $\{(ii), (iii)\}$ je napríklad postupnosť

$$\langle \neg chodieva\_na\_Zobor(fero), bydlisko(fero) = nitra \rangle.$$

■

Príklad naznačuje, že ničím nekontrolované (naivné) odvodzovanie pomocou defaultových pravidiel môže viesť k ťažkostiam, ku konfliktom s intuíciou.

Základná myšlienka sofistikovaného ponímania defaultového odvodzovania Časť nepríjemností, spätých s nekontrolovaným používaním defaultových pravidiel môžeme odstrániť, keď k defaultovému odvodzovaniu pristúpime sofistikovanejšie. Prvá formalizácia defaultového usudzovania [Rei 80] je založená na zaujímavejšom poňatí globálneho kontextu, na ktorý sa defaultové pravidlo odvoláva. Týmto kontextom je akási množina súčasne akceptovateľných presvedčení. Defaultové pravidlá sú aplikovateľné vtedy, keď ich stráže sú konzistentné s takouto množinou presvedčení. (V príklade 4.1 sme pre ňu pripravili termín *extenzia*.) Množinu presvedčení vytvára množina známych faktov a množina hypotéz, ktorými vypĺňujeme niektoré naše medzery v neúplnom poznaní. Cieľom je, aby množina presvedčení bola konzistentná.

V centre pozornosti teórie defaultov teda nie je odvoditeľnosť z nejakej množiny viet pomocou nejakej množiny defaultových pravidiel. Smer uvažovania charakterizovateľný definíciou 4.8 nie je kľúčový. Hlavná pozornosť je sústredená na extenzie, množiny presvedčení, ktorým možno veriť, ak máme k dispozícii nejakú množinu viet (explicitnú databázu) a množinu defaultových pravidiel. Týmto pootočením pozornosti budeme schopní charakterizovať defaultové teórie tak, aby zachovávali konzistentosť. Reiterov pohľad na defaultové pravidlá je taký, že ich nepovažuje za odvodzovacie pravidlá, ale za istý typ nedeterministických meta-pravidiel, inštrukcií, ako tvoriť extenzie neúplných teórií.

Zaujíma nás teda, aké sú udržateľné, konzistentné množiny zúplnení explicitnej databázy. Tieto zúplnenia vyjadrujú predstavu o očakávaniach, založených na obvyklom stave vecí. Vo všeobecnosti ide o výber zo všetkých možných očakávaní. Ide pritom o to, aby bolo možné veriť súčasne všetkým vybratým očakávaniam. Tie – ako celok – môžu byť nezlučiteľné. Pekne to ilustruje príklad 4.9.

Vtip je práve v tom, ako definovať (konštruovať) zlučiteľné množiny presvedčení. O tom je ďalšia časť.

4.1.1 Základy teórie

Po motivačných príkladoch môžeme prejsť k definíciám. Na rozdiel od kapitoly 3 nebudeme hovoriť o abstraktnom, nešpecifikovanom jazyku, ale budeme používať jazyk prvorádovej logiky.[5]

Kvôli jednoduššiemu konštruovaniu teórie našu pozornosť zameriame na uzavreté defaultové pravidlá. V príkladoch 4.2 a 4.4 sme používali otvorené defaultové pravidlá. Predpoklady, stráže i dôsledky boli výrazy s voľnými premennými. Čitateľovi je však zrejmé, že uvedené príklady možno transformovať do uzavretého tvaru – miesto daného pravidla zavedieme množinu všetkých jeho inštancií (samozrejme, ak máme mená všetkých objektov z univerza). Prípad otvorených defaultových teórií ostane mimo našej hlavnej pozornosti. Zmienime sa o nich až v závere tejto kapitoly, v komentároch. Toto zjednodušenie je však celkom primerané – zodpovedá doménovo-špecifickému charakteru defaultových pravidiel.

Definícia 4.10 (Defaultové pravidlo) *Defaultové pravidlo* (nad jazykom prvého rádu J) je pravidlo tvaru

$$\frac{\alpha_1, \ldots, \alpha_n : \beta_1, \ldots, \beta_m}{\gamma}.$$

Pritom $n \geq 0$, $m > 0$ a všetky formuly, ktoré sa v pravidle vyskytujú, sú uzavreté formuly jazyka J. Budeme ich nazývať takto: α_i sú predpoklady pravidla, β_j sú stráže a γ je dôsledok. ■

Predpoklad $m > 0$ sa v literatúre obvykle nevyskytuje, chceme však dosiahnuť, aby defaultové pravidlo nezdegenerovalo v obvyklé odvodzovacie pravidlo.

Niekedy budeme používať notáciu $\alpha_1, \ldots, \alpha_n : \beta_1, \ldots, \beta_m / \gamma$.

Ďalej, ak máme defaultové pravidlo d, jeho predpoklady budeme značiť ako $pre(d)$, stráže $just(d)$ a dôsledok $cons(d)$.[6] Teda pre $d = \alpha_1, \ldots, \alpha_n : \beta_1, \ldots, \beta_m/\gamma$ bude $pre(d) = \{\alpha_1, \ldots, \alpha_n\}$, $just(d) = \{\beta_1, \ldots, \beta_m\}$ a $cons(d) = \{\gamma\}$.

Výskyt stráží vo formulácii defaultových pravidiel ukazuje, že tieto pravidlá majú globálny a na kontexte závislý charakter: o aplikovateľnosti pravidla nemožno rozhodnúť lokálne, po rozpoznaní vhodného miesta v odvodzovacej reťazi. Jeho aplikovateľnosť závisí od stavu celej globálnej databázy, od toho, či v nej sú narušené stráže. V jednom kontexte môžu byť narušené, v inom nemusia.

Definícia 4.11 (Defaultová teória) *Defaultová teória* je dvojica (E, D), kde E sú uzavreté formuly jazyka J a D sú defaultové pravidlá. ■

V tejto chvíli môžeme prikročiť k definícii extenzie. Intuitívne, extenzia bude množina formúl $\mathcal{E}$, ktorá

[5] Základné pojmy možno nájsť v dodatkoch B a D.

[6] Opäť pripomeniem anglické termíny: prerequisities, justifications, consequents.

- obsahuje všetky vety z E (E môžeme považovať za bázu znalostí, ktorými disponuje výpočtový model alebo za axiómy „jeho teórie“),
- bude uzavretá vzhľadom na dedukciu; extenzia obsahuje všetky svoje deduktívne dôsledky,
- bude maximálna v tom zmysle slova, že neexistuje $d \in D$ aplikovateľné na $\mathcal{E}$, ktorého dôsledok by do $\mathcal{E}$ nepatril.

V definícii bude kľúčovým operátor Γ:

Definícia 4.12 (Extenzia) Nech (E, D) je defaultová teória. Nech X je množina viet jazyka J a Cn_{FOL} je operátor odvodenia predikátovej logiky prvého rádu. Nech $\Gamma(X)$ je najmenšia (vzhľadom na inklúziu) množina viet jazyka J, ktorá spĺňa nasledujúce podmienky:

- $E \subseteq \Gamma(X)$,
- $Cn_{FOL}(\Gamma(X)) = \Gamma(X)$,
- ak $(\alpha_1, \ldots, \alpha_n : \beta_1, \ldots, \beta_m / \gamma) \in D$, $\alpha_i \in \Gamma(X)$ pre každé i, $\neg\beta_j \notin X$ pre každé j, tak $\gamma \in \Gamma(X)$.

Množina viet $\mathcal{E}$, pre ktorú platí $\mathcal{E} = \Gamma(\mathcal{E})$, sa nazýva *extenziou* defaultovej teórie T. ■

$\mathcal{E}$ je teda penym bodom operátora Γ.[7]

Nasledujúci príklad z [Luk 90] objasní, prečo extenziu nedefinujeme priamo, ale pomocou operátora Γ. Objasní aj, prečo je v definícii extenzie $\alpha_i \in \Gamma(X)$ pre každé i, ale $\neg\beta_j \notin X$ pre každé j.

Príklad 4.13 Majme defaultovú teóriu, ktorej jedinou „axiómou“ je p, jediným defaultovým pravidlom je $p : q/q$. Teda, $E = \{p\}$ a $D = \{p : q/q\}$. Predstavme si, že by sme definíciu extenzie formulovali priamo pre $\mathcal{E}$ (miesto $\Gamma(X)$). To znamená, mali by sme

- $E \subseteq \mathcal{E}$,
- $Cn_{VL}(\mathcal{E}) = \mathcal{E}$ (keďže sa pohybujeme vo výrokovej logike, stačí nám Cn_{VL}),
- ak $\alpha_i \in \mathcal{E}, \neg\beta_j \notin \mathcal{E}$, potom $\gamma \in \mathcal{E}$.

[7] Ak máme operátor Φ a jeho argument A, pričom platí, že $\Phi(A) = A$, hovoríme, že A je pevným bodom operátora Φ. Pozri dodatok E. Konštrukciu pevného bodu budeme používať častejšie.

Dostali by sme dve extenzie $\mathcal{E}_1 = Cn_{VL}(\{p, q\})$ a $\mathcal{E}_2 = Cn_{VL}(\{p, \neg q\})$. Extenzia $\mathcal{E}_2$ nevyhovuje našej intuícii. Intuitívne by bolo, keby extenzia defaultovej teórie obsahovala iba formuly, ktoré možno získať dedukciou z E a z dôsledkov defaultových pravidiel.

Ľahko sa však dá presvedčiť, že $\Gamma(\mathcal{E}_2) \neq \mathcal{E}_2$: $\{p\} \subseteq \Gamma(\mathcal{E}_2)$, pretože podľa definície 4.12 $E \subseteq \Gamma(\mathcal{E}_2)$. Pretože $p : q/q$ nie je aplikovateľné na $\mathcal{E}_2$, q nepatrí do $\Gamma(\mathcal{E}_2)$. Ale ani $\neg q$ nepatrí do $\Gamma(\mathcal{E}_2)$, pretože nie je dôsledkom žiadneho defaultového pravidla aplikovateľného na $\mathcal{E}_2$, ani deduktívnym dôsledkom E. Aplikácia operátora Γ na $\mathcal{E}_2$ teda vedie iba k množine deduktívnych dôsledkov axiómy p.

Podobným postupom sa dá presvedčiť, že $\Gamma(\mathcal{E}_1) = \mathcal{E}_1$.

Tento príklad aj vysvetľuje, prečo je v definícii 4.12 $\alpha_i \in \Gamma(X)$ pre každé i, ale $\neg\beta_j \notin X$ pre každé j. ■

Uvedený príklad však nechce povedať, že defaultová teória nemôže mať viac extenzií, naopak, tento stav nie je neobvyklý.

Príklad 4.14 Ak $E = \emptyset$ a $D = \{(: p/p), (: \neg p/\neg p)\}$, táto defaultová teória má dve extenzie: $\mathcal{E}_1 = Cn_{FOL}(\{p\})$ a $\mathcal{E}_2 = Cn_{FOL}(\{\neg p\})$. ■

Na druhej strane, existencia extenzií defaultových teórií nie je zaručená.

Príklad 4.15 $T = (E, D)$, kde $E = \emptyset$ a $D = \{(: p/\neg p)\}$ nemá žiadnu extenziu. Ako možní kandidáti na extenziu prichádzajú do úvahy $\mathcal{E}_1 = Cn_{FOL}(\{\neg p\})$ (v prípade, že by defaultové pravidlo bolo aplikovateľné) alebo v opačnom prípade $\mathcal{E}_2 = Cn_{FOL}(\emptyset)$. Ak by $\mathcal{E}_1$ mala byť extenziou, muselo by byť na $\mathcal{E}_1$ aplikovateľné defaultové pravidlo. Stráž p defaultového pravidla je však vyvrátená v $\mathcal{E}_1$, teda $\neg p \notin \Gamma(\mathcal{E}_1)$. Preto $\mathcal{E}_1$ nie je pevný bod operátora Γ.

Všimnime si teraz $\mathcal{E}_2$. V $\mathcal{E}_2$ nie je vyvrátená stráž defaultového pravidla, pretože $\neg p \notin Cn_{FOL}(\emptyset)$ ($\neg p$, vyvrátenie stráže, neodvodíme pomocou výrokovej logiky z prázdnej množiny predpokladov). Teda, môžeme aplikovať defaultové pravidlo a $\neg p \in \Gamma(\mathcal{E}_2)$, preto ani $\mathcal{E}_2$ nie je pevný bod operátora Γ. ■

Teraz je čas zamyslieť sa nad charakterizáciou defaultového odvodzovania. Videli sme už, že naivná cesta cez postupné aplikovanie defaultových pravidiel nie je perspektívna. Zaviedli sme pojem extenzie, ktorý istým spôsobom charakterizuje množinu súčasne akceptovateľných presvedčení. Teraz, s pojmom extenzie, môžeme lepšie charakterizovať nedostatočnosť naivného výpočtu. Pri „naivnom výpočte“ sú problémy práve s príslušnosťou predpokladanej defaultovej hodnoty do nejakej extenzie. Nekladú sa otázky typu, či $\mathcal{E}$ ostane konzistentná po rozhodnutí prijať nejakú defaultovú hodnotu d atribútu A pre objekt o, či defaultovo prijatá veta $A(o) = d$ vôbec patrí do nejakej extenzie, či všetky akceptované defaultové vety patria do tej istej extenzie.

Videli sme tiež, že jedna defaultová teória môže mať viacero extenzií. To znamená, že defaultové odvodzovanie nemožno charakterizovať nejakým operátorom odvodenia produkujúcim presne jednu extenziu.

Zavedieme preto nedeterministický operátor defaultového odvodenia.

Definícia 4.16 Nech $T = (E, D)$ je defaultová teória, kde E je množina viet a D je množina defaultových pravidiel. Potom $Cn^D_{def}(E) \in \{\mathcal{E} \mid \mathcal{E}$ je extenziou $T\}$. ■

Poznámka 4.17 Ak $Cn^D_{def}(E) \in \{\mathcal{E} : \mathcal{E}$ je extenziou teórie $T\}$, potom hodnotu $Cn^D_{def}(E)$ možno vybrať spomedzi alternatív (ktorými sú extenzie danej defaultovej teórie).[8]

Tento druh defaultového usudzovania sa nazýva dôverčivý – nejakú alternatívu vyberieme a sme s ňou spokojní. Jeho protipólom je skeptické usudzovanie, ktoré prijme nejaký dôsledok iba vtedy, keď platí v každej extenzii. Dôverčivému a skeptickému usudzovaniu sa budeme venovať v časti 4.2.

Symbol $Cn^D_{def}(E)$ explicitne odkazuje na množinu defaultových pravidiel D. Pretože táto je obvykle zrejmá z kontextu, budeme index D vynechávať.

Množina extenzií nejakej defaultovej teórie (E, D) môže byť prázdna. Vtedy operátor Cn^D_{def} nie je definovaný pre E, „nevypočíta nič" (vyberá z prázdnej množiny). Cn^D_{def} preto budeme považovať za čiastočný nedeterministický operátor. ■

Aby sme zahladili stopy po neporiadnej robote, pojem nedeterministického a čiatočného nedeterministického operátora teraz zadefinujeme (toto poradie sme volili preto, aby sme mali na očiach najprv nejaký užitočný príklad). Obvyklý operátor priraďuje nejakej množine (z akéhosi priestoru množín) nejakú množinu z toho istého priestoru. Nedeterministický operátor nepriraďuje jedinú pevnú množinu (vždy tú istú pre jeden argument), ale vyberá jednu množinu z nejakej množiny množín.

Definícia 4.18 Nech X je nejaká množina. Φ je nedeterministický operátor definovaný na 2^X (na množine všetkých podmnožín X), ak pre $E \subseteq X$ hodnotou $\Phi(E)$ môže byť ľubovoľná množina $y \in Y$, kde $Y \subseteq 2^X$. ■

Poznámka 4.19 Čiastočný nedeterministický operátor nie je definovaný pre niektoré $E \in 2^X$. Tam, kde je definovaný, sa správa presne tak ako nedeterministický operátor. ■

Narazili sme teda na príklad nedeterministického operátora odvodenia, ktorý ukazuje, že nevystačíme s našou pôvodnou definíciou inferenčného stroja (ako operátora, ktorý nejakej množine viet priraďuje pevne určenú množinu viet).

[8] Opäť vidíme, že hypotetické usudzovanie býva alternatívne.

Alternatívnosť extenzií nás doviedla k predstave o nedeterministickom, alternatívnom správaní „operátora“ defaultového (a vo všeobecnosti, hypotetického) odvodenia. Táto vlastnosť nie je celkom očakávaná a príjemná. Obvyklý postoj k odvodzovaniu je taký, že v každom kontexte, v každom behu bude pre pevne danú množinu predpokladov jednoznačne určená množina jej dôsledkov. Príklad defaultového odvodzovania ukazuje, že pri hypotetickom odvodzovaní si budeme musieť odvyknúť od tejto predstavy. Ide však o veľmi dôležitý pohľad na povahu hypotetického usudzovania - hypotetické myslenie býva spravidla alternatívne.

V tejto súvislosti je vhodné upozorniť na nekumulatívnosť defaultového usudzovania.

Príklad 4.20 ([Mak 89]) Pripomeňme si, že operátor odvodenia Cn je kumulatívny, ak platí: $A \subseteq B \subseteq Cn(A) \Rightarrow Cn(A) = Cn(B)$. Kumulatívnosť vyjadruje istý druh uzáverovosti inferencie: ak množina viet B patrí do Cn-uzáveru množiny viet A, potom ďalšie aplikovanie Cn-uzáveru na B neprinesie nič nového v porovnaní s $Cn(A)$.

Predpokladajme defaultové pravidlá : p/p a $p \vee q : \neg p/\neg p$. Označme túto dvojicu pravidiel ako D. Nech ďalej $A = \emptyset$, $B = \{p \vee q\}$. Množinu výrokovologických dôsledkov množiny viet X značíme ako $Cn_{VL}(X)$. Jedinou extenziou defaultovej teórie (A, D) je $Cn_{VL}(\{p\})$, pretože predpoklad $p \vee q$ možno splniť až po aplikácii pravidla : p/p a v takejto situácii už druhé pravidlo nie je aplikovateľné. Označme si túto extenziu ako $Cn_{def}(A)$. Na druhej strane, (B, D) má dve extenzie: $Cn_{VL}(\{p\})$ a $Cn_{VL}(\{\neg p, q\})$. Môžeme to chápať tak, že $Cn_{def}(B)$ sa počíta nedeterministicky.

Predpoklad, vyžadovaný pre kumulatívnosť je splnený: $p \vee q$ vyplýva z p, preto $A \subseteq B \subseteq Cn_{def}(A)$. Pri „nedeterministickom“ výpočte $Cn_{def}(B)$ však môžeme dostať $Cn_{VL}(\{\neg p, q\})$.

Teda kumulatívnosť nie je splnená, $Cn_{def}(B) \neq Cn_{def}(A)$. Naviac, ani pri skeptickom ponímaní Cn_{def} kumulatívnosť neplatí: p nie je pravdivé vo všetkých extenziách teórie (B, D).

Množina viet B môže byť podmnožinou Cn_{def}-uzáveru množiny viet A, ale „opakované“ aplikovanie Cn_{def} na B môže dať výsledok, odlišný od $Cn_{def}(A)$. Toto zistenie teda ukazuje, že defaultové usudzovanie nemá vlastnosť uzáverovosti, obvyklú pri inferencii. ■

Po tomto extempore sa vráťme k pojmu extenzie. Vyššie uvedená definícia extenzie (4.12) nie je konštruktívna. Dáva iba návod na testovanie, či zvolená množina formúl je extenziou. Neukazuje, ako skonštruovať extenziu. Podobne je to aj s nasledujúcim tvrdením, aj keď sugeruje predstavu iterácie.

Tvrdenie 4.21 (O iterácii extenzií [Rei 80]) *Nech $T = (E, D)$ je defaultová teória. Potom $\mathcal{E}$ je extenziou T práve vtedy, keď $\mathcal{E} = \bigcup_{i=O}^{\infty} \mathcal{E}_i$, kde $\mathcal{E}_0 = E$*

a $\mathcal{E}_{i+1} = Cn_{FOL}(\mathcal{E}_i) \cup \{\gamma | (\alpha_1, \ldots, \alpha_n : \beta_1, \ldots, \beta_m / \gamma) \in D,\ \alpha_k \in \mathcal{E}_i$ *pre každé* $k,\ \neg\beta_j \notin \mathcal{E}$ *pre každé* $j\}$.

Idea dôkazu: Treba ukázať, že $\bigcup_{i=O}^{\infty} \mathcal{E}_i = \Gamma(\mathcal{E})$.

Pri inklúzii $\bigcup_{i=O}^{\infty} \mathcal{E}_i \subseteq \Gamma(\mathcal{E})$ nie je žiadny zvláštny problém, v induktívnom kroku z predpokladu $\mathcal{E}_r \subseteq \Gamma(\mathcal{E})$ ľahko odvodíme, že pre $\phi \in \mathcal{E}_{r+1}$ platí $\phi \in \Gamma(\mathcal{E})$. Buď ϕ patrí do deduktívneho uzáveru $\mathcal{E}_r$ a využijeme, že $\Gamma(\mathcal{E})$ je uzavretá vzhľadom na dedukciu. V opačnom prípade ϕ získame pomocou defaultového pravidla, ktorého predpoklady platia v $\mathcal{E}_r$ a stráže nie sú vyvrátené v $\mathcal{E}$. Podľa induktívneho predpokladu, predpoklady platia v $\Gamma(\mathcal{E})$, čo stačí podľa definície 4.12 na odvodenie záveru $\phi \in \Gamma(\mathcal{E})$.

Pre opačnú inklúziu treba ukázať, že $\bigcup_{i=O}^{\infty} \mathcal{E}_i$ spĺňa podmienky kladené v definícii 4.12 na $\Gamma(\mathcal{E})$. Pretože $\Gamma(\mathcal{E})$ je minimálna množina, ktorá tie podmienky spĺňa, musí potom dokazovaná inklúzia platiť. Ľahko vidno, že $E \subseteq \bigcup_{i=O}^{\infty} \mathcal{E}_i$ a $\bigcup_{i=O}^{\infty} \mathcal{E}_i$ je uzavretá vzhľadom na dedukciu.

Treba ešte ukázať, že dôsledok každého aplikovateľného pravidla patrí do $\bigcup_{i=O}^{\infty} \mathcal{E}_i$. Predpokladajme, že pre nejakú množinu X je $\mathcal{E}_{i+1} = Cn_{FOL}(\mathcal{E}_i) \cup \{\gamma | (\alpha_1, \ldots, \alpha_n : \beta_1, \ldots, \beta_m / \gamma) \in D,\ \alpha_k \in \mathcal{E}_i$ pre každé k, $\neg\beta_j \notin \mathcal{E}$ pre každé $j\}$.

Nech $\Gamma(X) = \mathcal{E}$, pričom $\mathcal{E}$ je definované ako $\mathcal{E} = \bigcup\limits_{i=O}^{\infty} \mathcal{E}_i$: Zrejme $\Gamma(X)$ obsahuje všetky dôsledky defaultových pravidiel aplikovateľných na X. $\Gamma(X)$ je extenziou, ak $\Gamma(X) = X = \mathcal{E} = \bigcup\limits_{i=O}^{\infty} \mathcal{E}_i$. ■

Čitateľ už iste postrehol, že defaultové pravidlá nemôžu zavliecť do defaultovej teórie nekonzistentosť:

Tvrdenie 4.22 *Defaultová teória* (E, D) *má nekonzistentnú extenziu práve vtedy, keď* E *je nekonzistentná.*

Idea dôkazu: Ak je E nekonzistentná, žiadne defaultové pravidlo nie je aplikovateľné. Totiž, $\mathcal{E}_0 = E$ a pre nekonzistentné E je $Cn_{FOL}(\mathcal{E}_0)$ množina všetkých viet jazyka. Preto v $Cn_{FOL}(\mathcal{E}_0)$ existuje k stráži každého defaultového pravidla jej negácia. Z toho podľa vety o iterácii extenzií platí, že $\mathcal{E} = Cn_{FOL}(\mathcal{E}_0)$, a teda $\mathcal{E}$ je nekonzistentná.

Predpokladajme, že E je konzistentná, ale $\mathcal{E}$ je nekonzistentná. Potom žiadne defaultové pravidlo nie je aplikovateľné (stráž každého z nich je blokovaná, extenzia totiž obsahuje všetky vety jazyka), to ale znamená, že $\mathcal{E} = Cn_{FOL}(E)$ je konzistentná. ■

Intuíciu minimality extenzie (v zmysle, že extenzia nemôže mať podmnožiny, ktoré sú tiež extenziami) a zároveň maximality (v zmysle akejsi nasýtenosti) presne vyjadruje nasledujúce tvrdenie:

Tvrdenie 4.23 *Ak* $\mathcal{E}, \mathcal{F}$ *sú extenzie defaultovej teórie* T *a* $\mathcal{E} \subseteq \mathcal{F}$, *potom* $\mathcal{E} = \mathcal{F}$.

Hlavná myšlienka dôkazu: indukciou ukázať, že $\mathcal{F}_n \subseteq \mathcal{E}_n$ z predpokladu $\bigcup_{i=O}^{\infty} \mathcal{E}_i \subseteq \bigcup_{i=O}^{\infty} \mathcal{F}_i$. Samozrejme, $\mathcal{F}_0 = \mathcal{E}_0$. Predpokladajme $\mathcal{F}_i \subseteq \mathcal{E}_i$. Nech $\phi \in \mathcal{F}_{i+1}$. To znamená, že

$$\begin{aligned}\phi \in Cn_{FOL}(\mathcal{F}_i) \quad \cup \quad & \{\gamma | (\alpha_1, \ldots, \alpha_n : \beta_1, \ldots, \beta_m / \gamma) \in D, \\ & \alpha_k \in \mathcal{F}_i \text{ pre každé } k, \neg\beta_j \notin \mathcal{F} \text{ pre každé } j\}.\end{aligned}$$

Ľahko dostávame, že $\phi \in \mathcal{E}_{i+1}$. ■

Ku konštrukcii danej extenzie nemusia prispievať všetky defaultové pravidlá danej teórie:

Definícia 4.24 (Generujúce defaulty) *Generujúce defaulty* extenzie $\mathcal{E}$ defaultovej teórie $T = (E, D)$ sú defaultové pravidlá z množiny

$$GD(\mathcal{E}, T) = \{d \in D \ : \ \mathcal{E} \models pre(d) \wedge \forall \beta \in just(d)\ \mathcal{E} \not\models \neg\beta\}.$$

■

Tvrdenie 4.25 *Ak $\mathcal{E}$ je extenzia defaultovej teórie $T = (E, D)$, potom $\mathcal{E} = Cn_{FOL}(E \cup cons(GD(\mathcal{E}, T)))$.*

Idea dôkazu: Inklúzia $\mathcal{E} \subseteq Cn_{FOL}(E \cup cons(GD(\mathcal{E}, T)))$ s využitím rozvoja $\mathcal{E} = \bigcup_{i=0}^{\infty} \mathcal{E}_i$. Nech $\gamma \in \bigcup_{i=0}^{\infty} \mathcal{E}_i$, teda $\gamma \in \mathcal{E}_j$ pre nejaké j. Indukciou cez j: pre $j = 0$ určite $\gamma \in E$. Inak, buď $\gamma \in Cn_{FOL}(\mathcal{E}_{j-1})$ (vtedy použijeme induktívny predpoklad) alebo pre nejaké defaultové pravidlo d je $cons(d) = \gamma$, pričom $pre(d) \subseteq \mathcal{E}_{j-1}$ a pre každé $\beta \in just(d)$ platí $\neg\beta \notin \mathcal{E}$. Z toho dostávame $\gamma \in GD(\mathcal{E}, D)$.

Opačná inklúzia: Nech $\gamma \in Cn_{FOL}(E \cup cons(GD(\mathcal{E}, D))$ a $\gamma \notin Cn_{FOL}(E)$. To znamená, že existuje defaultové pravidlo d také, že $\mathcal{E} \models pre(d)$, $\mathcal{E} \not\models \neg\beta$ pre každé $\beta \in just(d)$. Pritom buď priamo $\gamma \in cons(d)$ alebo γ získame dedukciou z E a z nejakej množiny dôsledkov defaultových pravidiel. To znamená, $\gamma \in \mathcal{E}$. ■

I keď nie každá defaultová teória musí mať extenziu, existujú dobre definované prípady, kedy je existencia extenzie zaručená. Veľmi dôležitým sú normálne defaultové teórie.

Definícia 4.26 (Normálna defaultová teória) Defaultové pravidlo tvaru

$$\frac{\alpha : \beta}{\beta}$$

je *normálne.* Defaultová teória je normálna, ak všetky jej defaultové pravidlá sú normálne. ■

Poznámka 4.27 Čo hovoria normálne defaultové pravidlá o zúplnení? Ak $\alpha : \beta/\beta \in D$, $\alpha \in \mathcal{E}$, $\neg\beta \notin \mathcal{E}$, potom $\beta \in \mathcal{E}$. Inými slovami, pre každé $d \in D$, pre každú extenziu $\mathcal{E}$, v ktorej je splnené $pre(d)$, platí buď $cons(d) \in \mathcal{E}$ alebo $\neg cons(d) \in \mathcal{E}$. ■

Tvrdenie 4.28 *Každá normálna defaultová teória má extenziu.*

Dôkaz nie je priamočiary. Nemožno použiť ani definíciu, ani tvrdenie o iterácii, keďže obe explicitne používajú $\mathcal{E}$, a my máme dokázať existenciu $\mathcal{E}$. Predpokladajme, že E je konzistentná (v opačnom prípade je tvrdenie triviálne). Použijeme nasledujúcu iteratívnu konštrukciu: nech

- $\mathcal{E}_0 = E$,
- $\mathcal{E}_{i+1} = Cn_{FOL}(\mathcal{E}_i) \cup T_i$,

kde T_i je maximálna (vzhľadom na inklúziu) množina splňujúca:

(1) $\mathcal{E}_i \cup T_i$ je konzistentná,

(2) ak $\beta \in T_i$, potom existuje pravidlo $(\alpha : \beta/\beta) \in D$ také, že $\alpha \in Cn_{FOL}(\mathcal{E}_i)$.

Definujme $\mathcal{E} = \bigcup_{i=O}^{\infty} \mathcal{E}_i$.

Na to, aby sme dokázali, že $\bigcup_{i=O}^{\infty} \mathcal{E}_i$ je extenzia normálnej defaultovej teórie, stačí ukázať, že $T_i = \{\beta | (\alpha : \beta/\beta) \in D, \alpha \in Cn_{FOL}(\mathcal{E}_i), \neg\beta \notin \mathcal{E}\}$. Označme si výraz na pravej strane posledného znaku rovnosti ako P.

Nech neplatí $P \subseteq T_i$, teda $\exists \beta(\beta \in P \wedge \beta \notin T_i)$. To znamená, že $\mathcal{E}_i \cup T_i \cup \{\beta\}$ je nekonzistentná, lebo $\mathcal{E}_i \cup T_i$ je maximálna konzistentná množina s vlastnosťou (2). Preto $\neg\beta$ je deduktívnym dôsledkom $\mathcal{E}_i \cup T_i$, čo podľa definície znamená, že $\neg\beta \in \mathcal{E}$, to je však v spore s konštrukciou P. Teda $P \subseteq T_i$.

Opačná inklúzia je ľahká: pre $\beta \in T_i$ podmienka konzistentnosti (1) dáva $\neg\beta \notin \mathcal{E}$. ∎

Zaujímavú a charakteristickú vlastnosť extenzií normálnych defaultových teórií ukazuje nasledujúce tvrdenie.

Tvrdenie 4.29 *Ak normálna defaultová teória má rôzne extenzie $\mathcal{E}, \mathcal{F}$, potom $\mathcal{E} \cup \mathcal{F}$ je nekonzistentná množina formúl.*

Základná myšlienka: Dôkaz z tvrdenia o iterácii a z faktu, že existuje najmenšie i také, že $\mathcal{E}_i \neq \mathcal{F}_i$. Fakt, že $\mathcal{E} \neq \mathcal{F}$ sa dá vysvetliť iba tým, že niektorý default, prispievajúci k $\mathcal{E}$, nie je aplikovateľný na $\mathcal{F}$ alebo naopak. Nech napríklad $\neg\beta \in \mathcal{E}$ a $\neg\beta \notin \mathcal{F}$. To znamená, že $\alpha : \beta/\beta$ je aplikovateľné na $\mathcal{F}$ a nie na $\mathcal{E}$, čiže $\beta \in \mathcal{F}$, $\beta \wedge \neg\beta \in \mathcal{E} \cup \mathcal{F}$. ∎

Poznámka 4.30 $\mathcal{E}$, $\mathcal{F}$ sú „podopreté“ maximálnymi konzistentnými podmnožinami $cons(D) = \bigcup_{d \in D} cons(d)$, ich zjednotenie však musí byť nekonzistentné.

Tvrdenie 4.29 neplatí pre ľubovoľné defaultové teórie: Ak $E = \emptyset$ a $D = \{: p \wedge \neg s/p, : \neg p/s, : p \wedge \neg s/q, : \neg p/r\}$, potom $\mathcal{E}_1 = Cn_{VL}(\{p,q\})$ a $\mathcal{E}_2 = Cn_{VL}(\{s,r\})$, pričom ich zjednotenie nie je nekonzistentné. ∎

Normálna defaultová teória má za istých okolností jedinú extenziu:

Tvrdenie 4.31 *Nech* $T = (E, D)$ *je normálna defaultová teória a* $cons(D) = \bigcup_{d \in D} cons(d)$. *Ak* $E \cup cons(D)$ *je konzistentná, potom* T *má jedinú extenziu.*

Idea dôkazu: Keďže T je normálna, určite má extenziu. Predpokladajme, že má dve rôzne extenzie $\mathcal{E}_1$, $\mathcal{E}_2$. Podľa tvrdenia 4.29 je $\mathcal{E}_1 \cup \mathcal{E}_2$ nekonzistentná. To je však v spore s konzistentnosťou $E \cup cons(D)$. ■

Poznámka 4.32 Podmienka konzistentnosti $E \cup cons(D)$ z tvrdenia 4.31 nie je nutná: Ak $E = \{p\}$ a $D = \{: \neg p / \neg p\}$, existuje jediná extenzia, $Cn_{VL}(\{p\})$, hoci $E \cup cons(D)$ je nekonzistentná.

Ak $E \cup cons(D)$ je nekonzistentná, je dôležité, či existuje maximálne (vzhľadom na inklúziu) X také, že $\emptyset \subset X \subseteq cons(D)$, kde $E \cup X$ je konzistentná. Vtedy $Cn_{FOL}(E \cup X)$ je extenzia. V prípade, že je viac takých maximálnych X, je aj viac extenzií. Ak neexistuje neprázdne X také, že $X \cup E$ je konzistentná, extenziu predstavujú všetky deduktívne dôsledky E.

Tvrdenie 4.31 neplatí pre ľubovoľné defaultové teórie. Príkladom je $E = \{p\}$ a $D = \{p : q/r, p : \neg r/\neg q\}$. $E \cup cons(D)$ je konzistentná, ale teória má dve extenzie: $Cn_{VL}(\{p, r\})$ a $Cn_{VL}(\{p, \neg q\})$. ■

Na začiatku tejto časti sme videli, že priamočiare odvodzovanie pomocou defaultových pravidiel môže viesť k ťažkostiam. Preto sme miesto predstavy odvodzovania v defaultových teóriách zaviedli predstavu množiny rozumných presvedčení, založených na defaultovej teórii. Spresnili sme ju pojmom extenzie. Postupná aplikácia defaultových pravidiel však vedie k extenzii v prípade normálnych defaultových teórií:

Tvrdenie 4.33 (Semi-monotónnosť) *Nech* D, D_1 *sú množiny normálnych uzavretých defaultových pravidiel, pričom* $D_1 \subseteq D$. *Nech* $\mathcal{E}_1$ *je extenziou* $T_1 = (E, D_1)$ *a nech* $T = (E, D)$. *Potom existuje* $\mathcal{E}$, *extenzia teórie* T, *pre ktorú platí*

(i) $\mathcal{E}_1 \subseteq \mathcal{E}$,

(ii) $GD(\mathcal{E}_1, T_1) \subseteq GD(\mathcal{E}, T)$.

Idea dôkazu: Hlavný vtip je uvedomiť si, že normálna defaultová teória $T_2 = (\mathcal{E}_1, D)$ má určite extenziu, označme si ju $\mathcal{E}$.

Treba dokázať, že $\mathcal{E}$ je aj extenziou $T = (E, D)$: Očividne platí $E \subseteq \mathcal{E}_1 \subseteq \mathcal{E}$. Samozrejme, $\mathcal{E}$ je uzavretá vzhľadom na dedukciu. Dôkaz završuje triviálne preverenie faktu, že $\mathcal{E}$ je uzavretá vzhľadom na všetky aplikovateľné defaultové pravidlá z D.

Tvrdenie (ii) je ľahkým dôsledkom. ■

Poznámka 4.34 Semimonotónnosť možno dokázať aj bez odvolávania sa na vetu o existencii extenzie normálnych defaultových teórií:

Nech je daná množina viet E a množina normálnych defaultových pravidiel D. Predpokladajme, že E je konzistentná (v opačnom prípade je tvrdenie triviálne). Nech $D_0 = \emptyset$. Nech ďalej $\mathcal{E}_0$ je extenziou teórie $T_0 = (E, D_0)$. Očividne, $\mathcal{E}_0 = Cn_{FOL}(E)$.

Sledujme teraz pridávanie normálnych defaultových pravidiel. Predpokladajme, že pre nejaké i je $\mathcal{E}_i$ extenziou $T_i = (E, D_i)$. Ak existuje $d \in D \setminus D_i$, ktoré je aplikovateľné na $\mathcal{E}_i$, potom $Cn_{FOL}(\mathcal{E}_i \cup cons(d)) = \mathcal{E}_{i+1}$ je extenziou $T_{i+1} = (E, D_{i+1})$, kde $D_{i+1} = D_i \cup \{d\}$. Pomocou indukcie dostávame semimonotónnosť.

Dôležitosť semimonotónnosti ukazuje tento fakt: Ak pre triedu $\mathcal{T}$ defaultových teórií platí semimonotónnosť, potom každé $T \in \mathcal{T}$ má extenziu. Ľahko to možno dokázať indukciou cez kardinalitu množiny defaultových pravidiel.
■

Na tvrdení o semimonotónnosti je založená teória dôkazu pre normálne defaultové teórie. Táto teória dôkazu rieši úlohu: ak je daná defaultová teória T a formula ϕ, treba určiť, či existuje extenzia $\mathcal{E}$ teórie T taká, že $\phi \in \mathcal{E}$.

Teórii dôkazu sa nebudeme venovať podrobnejšie. Stručne uvedieme iba základné predstavy:

Príklad 4.35 Nech $T = (E, D)$, $E = \{p\}$, $D = \{d_1 = p : q/q, d_2 = q : r/r\}$. Dôkazom formuly r vzhľadom na T je postupnosť $\langle\{d_2\}, \{d_1\}, \{\}\rangle$. Dôkaz je teda postupnosť množín defaultových pravidiel. Jej prvý člen umožňuje odvodiť dokazovanú formulu (d_2 umožňuje odvodiť r). Ďalšie členy postupnosti dávajú dôvody pre platnosť predpokladov prv použitých defaultových pravidiel. Zastavíme na formulách z E. Teda, dôkaz sa konštruuje od cieľa.
■

Definícia 4.36 Nech je daná defaultová teória $T = (E, D)$ a formula β. Konečná postupnosť $\langle D_0, \ldots, D_k\rangle$, kde $k \geq 0$ a D_i sú konečné podmnožiny D, je *defaultový dôkaz* β vzhľadom na T práve vtedy, keď

1. $\beta \in Cn_{FOL}(E \cup cons(D_0))$,
2. $Cnj(pre(D_{i-1})) \in Cn_{FOL}(E \cup cons(D_i))$, kde $Cnj(Y)$ je konjunkcia formúl z Y,
3. $D_k = \{\}$,
4. $E \cup \bigcup_{i=0}^{k} cons(D_i)$ je konzistentná množina.

■

Tvrdenie 4.37 (Úplnosť) *Konzistentná normálna defaultová teória T má extenziu obsahujúcu vetu β práve vtedy, keď existuje defaultový dôkaz β (vzhľadom na T).*

Podrobný dôkaz možno nájsť v [Rei 80]. Pretože teória dôkazu leží mimo centra našej pozornosti, iba pripomenieme, že úplnosť dôkazovej procedúry pre normálne defaultové teórie je založená na ich semimonotónnosti. Semimonotónnosť normálnych defaultových teórií však vedie ku konfliktu s intuíciami. Viac o tom v nasledujúcej časti.

4.1.2 Problémy

Už niekoľkokrát sme si všimli isté ťažkosti, späté s formalizáciou defaultového odvodzovania. Teraz sme už pripravení na to, aby sme sa im venovali detailnejšie.

Vypočítateľnosť

„Procedúra" defaultového dôkazu nie je vo všeobecnosti vykonateľná. V definícii 4.36 sa opakovane vyskytuje pojem prvorádovej konzekvencie (Cn_{FOL}) alebo konzistentnosti. Keďže prvorádové odvodenie nie je rozhodnuteľné (nemožno pre každú dvojicu (X, ϕ) rozhodnúť, či platí $\phi \in Cn_{FOL}(X)$ alebo nie), riešenie problémov tohto druhu naráža na zásadné prekážky. Podobne je to aj s inými možnými výpočtami problémov súvisiacich s extenziami (napríklad ide o problém členstva v extenzii, vo všetkých extenziách, o konštrukciu aspoň jednej extenzie či všetkých extenzií).

Existujú však rozhodnuteľné prípady týchto problémov. Niektorým z nich sa budeme venovať v časti 4.3. Pre tieto prípady si budeme bližšie všímať zložitosť výpočtov v defaultových teóriách.

Interakcia normálnych defaultov

Podľa pôvodných Reiterových predstáv normálne defaultové teórie by mali stačiť na formalizáciu intuícií, ktoré máme späté s usudzovaním na základe predstáv o obvyklom stave vecí: Nejakú hypotézu (default) akceptujeme, ak je konzistentné ju predpokladať. Záujem teoretikov o normálne defaultové teórie bol posilnený ich relatívne slušnými vlastnosťami: zaručená existencia extenzie, semimonotónnosť, existencia „dôkazovej procedúry".

Intuitívna úvaha, ktorá naznačuje, že každé defaultové pravidlo možno transformovať na normálne, vyzerá asi takto: Predpokladajme, že máme pravidlo

$$\frac{\alpha : \beta}{\gamma}.$$

Ak nemáme argumenty proti platnosti β, môžeme si dovoliť vygenerovať hypotézu β. A naopak, ak sme rozumným spôsobom odvodili γ, mali by sme predpokladať, že nie sú nijaké argumenty proti γ. Teda prijateľnými hypotézami (dôsledkami defaultového pravidla) sú aj β aj γ, rovnako obe môžeme považovať za prijateľné stráže. Miesto pôvodného pravidla preto môžeme navrhnúť a používať pravidlo

$$\frac{\alpha : \beta \wedge \gamma}{\beta \wedge \gamma}.$$

Normálne defaultové pravidlá sa však dostávajú do interakcií, ktoré vedú k neintuitívnym dôsledkom. Prvýkrát na to poukázali [ReC 81].

Jednoduchým príkladom neintuitívnosti, spôsobenej interakciami viacerých normálnych pravidiel, je tranzitívnosť.

Príklad 4.38 Majme tieto vety:

- Absolventi gymnázií sú obvykle (v typickom prípade) dospelí.
- Dospelí ľudia sú v typickom prípade zamestnaní.
- Absolventi gymnázií nie sú v typickom prípade zamestnaní.

Skúsme uvedené tri vety reprezentovať nasledujúcimi normálnymi defaultovými pravidlami (značenie: a – „Jožo je absolvent gymnázia“, d – „Jožo je dospelý“, z – „Jožo je zamestnaný“).

$$a : d/d$$
$$d : z/z$$
$$a : \neg z/\neg z$$

Tranzitívnosť (preverili sme ju v príklade 4.35) nie je vlastnosťou, ktorú by sme tu uvítali:

Ak prijmeme, že $E = \{a\}$, dostávame dve extenzie: $\mathcal{E}_1 = Cn_{VL}(\{a, d, z\})$, $\mathcal{E}_2 = Cn_{VL}(\{a, d, \neg z\})$. $\mathcal{E}_1$ sme získali na základe tranzitívnosti pomocou prvých dvoch pravidiel.

Práve $\mathcal{E}_1$ je extenzia, ktorá je v konflikte s intuíciami – s predpokladaným intuitívnym zámyslom pri formulovaní tejto defaultovej miniteórie. Na druhej strane, z formálneho hľadiska je to plne legitimná extenzia. Takéto extenzie – vyhovujúce z formálneho hľadiska, ale neintuitívne[9] – sa nazývajú anomálnymi extenziami. ■

Jedným z možných riešení problémov, spôsobených interakciami normálnych defaultových pravidiel, je použitie semi-normálnych defaultových teórií.

[9] Iba si pripomeňme poznámky o adekvátnosti špecifikácií a možných konfliktoch špecifikácií s intuíciami, kapitola 1.

Definícia 4.39 (Semi-normálna defaultová teória) Defaultové pravidlo tvaru

$$\frac{\alpha_1, \ldots, \alpha_k : \beta_1, \ldots, \beta_m,}{\gamma},$$

je *semi-normálne*, ak $\{\beta_1, \ldots, \beta_m\} \models \gamma$. Pre $m = 1$ predpokladáme, že $\beta_1 \not\equiv \gamma$, aby sa pravidlo nezvrhlo na normálne.

Defaultová teória (E, D) je *semi-normálna*, ak $D = D_1 \cup D_2$, kde D_1 je množina normálnych defaultových pravidiel (môže byť aj prázdna) a D_2 je neprázdna množina semi-normálnych pravidiel. ■

Poznámka 4.40 Najužívanejší tvar semi-normálneho defaultového pravidla je:

$$\frac{\alpha : \beta \wedge \gamma,}{\gamma}.$$

■

Príklad 4.41 Ak v príklade 4.38 nahradíme pravidlo $d : z/z$ pravidlom $d : z \wedge \neg a/z$, dostávame – v súlade s intuíciami – jedinú extenziu $\mathcal{E}_2$. ■

Podobné interakcie môžu vzniknúť medzi E a D (v prípade normálnych pravidiel):

Príklad 4.42 ([ReC 81]) Určite sme ochotní akceptovať, že všetci osemnásťroční sú dospelí. Podobne, v typickom prípade sú dospelí zosobášení. Uvedené predpoklady môžeme vyjadriť defaultovou teóriou:

$$\begin{aligned} E &= \{18 \rightarrow dosp\}, \\ D &= \{dosp : sob/sob\}. \end{aligned}$$

■

Z tejto teórie môžeme pre ľubovoľného osemnásťročného človeka odvodiť záver, že je v typickom prípade zosobášený. Všeobecnejšie, pre ľubovoľnú atypickú okrajovú vrstvu nejakej „populácie“ môžeme dostať neintuitívne závery, ktoré sú typické pre jej nadmnožinu. Problém možno opäť riešiť semi-normálnym defaultovým pravidlom.

V časti 4.2 si všimneme priamočiary – i keď problematický – spôsob, ako pomocou normálnych defaultových pravidiel možno reprezentovať dedičnosť v hierarchických sieťach. Ukážeme si aj, že semi-normálne defaultové teórie túto reprezentáciu zjemnia a umožnia intuitívnejšie odvodzovanie.

Žiaľ, semi-normálne defaultové teórie majú v porovnaní s normálnymi niektoré negatívne vlastnosti.

Semi-normálne defaultové teórie nemusia mať extenziu:

Príklad 4.43 Uvažujme $(\emptyset, \{: p \wedge \neg q/\neg q, : q \wedge \neg r/\neg r, : r \wedge \neg p/\neg p\})$.
Táto defaultová teória nemá extenziu. ■

Semi-normálne defaultové teórie nie sú semi-monotónne:

Príklad 4.44 Uvažujme teóriu $T_1 = (\emptyset, \{: p \wedge q/q\})$. Jej jedinou extenziou je $\mathcal{E}_1 = Cn_{VL}(\{q\})$. Ak pridáme ďalšie defaultové pravidlo, $: \neg p/\neg p$, jedinou extenziou novej, rozšírenej defaultovej teórie T_2 je $\mathcal{E}_2 = Cn_{VL}(\{\neg p\})$.

V prvom prípade, pri T_1, môžeme predpokladať $p \wedge q$ ($E = \emptyset$ ani $\mathcal{E}_1$ túto konjunkciu nevylučujú).

V druhom prípade predpokladajme, že existuje $\mathcal{E}$, extenzia T_2 taká, že $q \in \mathcal{E}$. Tento predpoklad však vedie k sporu:

- ak ani $p \in \mathcal{E}$, ani $\neg p \in \mathcal{E}$, na základe pravidla $: \neg p/\neg p$ musí byť $\neg p \in \mathcal{E}$,
- ak $p \in \mathcal{E}$, potom $: \neg p/\neg p$ nie je aplikovateľné, ale $p \notin \mathcal{E}$, pretože p nezískame z $\emptyset \cup \{q\}$
- ak $\neg p \in \mathcal{E}$, pravidlo $: p \wedge q/q$ nie je aplikovateľné, preto $q \notin \mathcal{E}$.

Teda, semi-monotónnosť pre semi-normálne pravidlá neplatí. ■

Napokon, rozhodnutie vyjadriť interakcie medzi pravidlami explicitne pomocou semi-normálnych pravidiel spôsobuje vážne koncepčné ťažkosti: určite je náročné v čase „tvorby bázy znalostí“ a v čase jej vývoja odhaliť a sformulovať všetky interakcie. K problému sa ešte vrátime v časti 4.2.

Lokálna vs. globálna konzistentnosť

Reiterova formalizácia pracuje s lokálnou požiadavkou konzistentnosti. To znamená, že konzistentnosť stráží každého pravidla sa testuje izolovane. O čo ide, naznačia nasledujúce príklady (pochádzajú z [DSJ 94]).

Príklad 4.45 Uvažujme $T = (\emptyset, \{: p/q, : \neg p/r\})$. Extenziou je $Cn_{VL}(\{q, r\})$, napriek tomu, že q a r by nemali platiť súčasne. ■

V istom zmysle slova to znamená, že extenzia neberie stráže defaultových pravidiel záväzne.[10]

Príklad 4.46 $T = (\emptyset, \{: p/q, : \neg p/r, : \neg q/s, : \neg r/t\})$. Táto teória má jedinú extenziu $Cn_{VL}(\{q, r\})$, napriek tomu, že intuitívne by bolo udržiavať tieto dve množiny presvedčení: $Cn_{VL}(\{q, t\})$ a $Cn_{VL}(\{r, s\})$. Dobré pochopenie podporí táto interpretácia: p – je slnečno, q – pôjdeme na pláž, r – pôjdeme do kina, s – Stano bude šťastný t – Tomáš bude šťastný. ■

[10] V anglickej literatúre sa používa termín „commitment to assumptions“.

Príklad 4.47 $T = (\{\neg p \vee \neg r\}, \{: u \wedge p/u, : q \wedge r/q\})$.

Podľa Reiterovej formalizácie je extenziou $Cn_{VL}(\{\neg p \vee \neg r, u, q\})$. Intuitívne je však veriť buď v $Cn_{VL}(\{u, \neg r\})$ alebo v $Cn_{VL}(\{q, \neg p\})$. ■

Problém s absenciou starostlivosti o globálnu konzistentnosť rieši Constrained Default Logic (*ConDL*) [DSJ 94]. Podstata prístupu je v tom, že zachováva Reiterovu definíciu defaultovej teórie, ale Reiterom zavedený pojem extenzie sa nepovažuje za postačujúci. Na syntaktickej úrovni sa nič nemení, mení sa však predstava o tom, aký je očakávaný stav vecí (či presvedčení), zodpovedajúci danej defaultovej teórii. Inak nazerajú na to, čo možno považovať za súčasne udržateľné presvedčenia pre danú defaultovú teóriu.

Zavádzajú pojem obmedzovanej extenzie. Lokálny test konzistentnosti stráže v aplikovanom defaultovom pravidle nahradzujú globálnou starostlivosťou o konzistentnosť. Základnú myšlienku si môžeme priblížiť tak, že popri vetách, ktorým možno konzistentne veriť pri danej defaultovej teórii sa treba starať aj o množinu integritných obmedzení, ktorú tvoria stráže aplikovaných defaultových pravidiel. Cieľom je udržať obe množiny v konzistentnom stave.

Môžeme prejsť k definícii obmedzovanej extenzie v *ConDL*. V tejto definícii sa pracuje s dvojicou množín viet (S, T). Množina viet S zodpovedá štandardnej extenzii, jej nadmnožina T predstavuje akési ohraničenie, v ktorom sa stráži konzistentnosť:

Definícia 4.48 (Obmedzovaná extenzia) Nech $T = (E, D)$ je defaultová teória nad jazykom J. X nech je množina viet jazyka J. Nech $\Gamma(X)$ je dvojica (S, T) najmenších množín viet jazyka J, ktoré spĺňajú nasledujúce podmienky:

- $E \subseteq S \subseteq T$,
- $Cn_{FOL}(S) = S$, $Cn_{FOL}(T) = T$,
- pre ľubovoľné $(\alpha_1, \ldots, \alpha_n : \beta_1, \ldots, \beta_m/\gamma) \in D$ platí: ak pre každé i je $\alpha_i \in S$ a množina $X \cup \{\beta_j : 1 \leq j \leq m\} \cup \{\gamma\}$ je konzistentná, tak $\gamma \in S$ a $\beta_1 \wedge \cdots \wedge \beta_m \wedge \gamma \in T$.

Dvojica množín viet $(\mathcal{E}, \mathcal{C})$, pre ktorú platí $\Gamma(\mathcal{C}) = (\mathcal{E}, \mathcal{C})$, sa nazýva *obmedzovanou (constrained) extenziou* defaultovej teórie. ■

Operátor Γ sa tu aplikuje na množinu presvedčení, v ktoré možno konzistentne veriť pri danej defaultovej teórii, rozšírenú o množinu integritných obmedzení (stráží aplikovaných defaultových pravidiel). Teda, kontext usudzovania sa explicitne ohraničuje spomínanými integritnými obmedzeniami.

Poznámka 4.49 Kým tradičné integritné obmedzenia hovoria o tom, aká musí byť databáza, „integritné obmedzenia“ v kontexte obmedzovanej extenzie charakterizujú, aké môžu byť rôzne (alternatívne) množiny viet, medzi ktorými sa vyskytujú aj hypotézy. Tieto rôzne alternatívy sa tvoria tak, že sa vyberajú maximálne konzistentné podmnožiny integritných obmedzení. ■

Príklad 4.50 Defaultová teória $(\{\neg q \vee \neg s\}, \{: p \wedge q/p, : r \wedge s/r\})$ má dve obmedzované extenzie

$$(Cn_{FOL}(\{p, \neg q \vee \neg s\}), Cn_{FOL}(\{p, q, \neg s\})),$$
$$(Cn_{FOL}(\{r, \neg q \vee \neg s\}), Cn_{FOL}(\{\neg q, r, s\})).$$

■

Táto modifikácia pojmu extenzie má dve veľmi cenené vlastnosti: je zaručené, že platí semi-monotónnosť a že existujú obmedzované extenzie každej teórie.

Tvrdenie 4.51 (Semimonotónnosť) *Nech $T = (E, D)$ je defaultová teória, nech $D \subseteq D'$. Ak $(\mathcal{E}, \mathcal{C})$ je obmedzovaná extenzia T, potom existuje obmedzovaná extenzia $(\mathcal{E}', \mathcal{C}')$ teórie (E, D') a platí $\mathcal{E} \subseteq \mathcal{E}'$ a $\mathcal{C} \subseteq \mathcal{C}'$.*

Idea dôkazu: Podobne ako v poznámke 4.34 konštruujme postupnosť defaultových pravidiel. Štartujeme z $(\mathcal{E}, \mathcal{C})$. Nájdeme aplikovateľné pravidlo (podľa definície 4.48). K $\mathcal{E}$ pridáme jeho dôsledok, k $\mathcal{C}$ dôsledok spolu so strážami. Takto postupujeme dovtedy, kým túto postupnosť nasýtime (t.j. dovtedy, kým žiadne pravidlo nie je aplikovateľné). Keď zastavíme, dostávame dvojicu $(\mathcal{E}', \mathcal{C}')$. O nej treba ešte preveriť, že je obmedzovanou extenziou. Podrobný dôkaz v [DSJ 94].

Tvrdenie 4.52 *Každá defaultová teória má obmedzovanú extenziu.*

Dôkaz z tvrdenia 4.51.

Príklad 4.53 Obmedzovanou extenziou teórie $T = (\emptyset, \{: \neg p/p\})$ je

$$(Cn_{VL}(\emptyset), Cn_{VL}(\emptyset)).$$

Ak (A, B) je obmedzovaná extenzia teórie T, v B sú stráže a dôsledky defaultových pravidiel (za predpokladu, že sú konzistentné). B však obsahuje prázdnu množinu stráží a dôsledkov: Keďže podľa definície $A \subseteq B$ a A, B sú uzavreté vzhľadom na dedukciu, $(Cn_{VL}(\emptyset), Cn_{VL}(\emptyset))$ je obmedzovanou extenziou. Možno treba pripomenúť, že Cn_{VL}-dôsledkami prázdnej množiny viet sú tautológie výrokovej logiky. ■

Tvrdenie 4.54 *Nech (E, D) je defaultová teória, Ak $(\mathcal{E}, \mathcal{C})$ a $(\mathcal{E}', \mathcal{C}')$ sú jej rôzne obmedzované extenzie, potom $\mathcal{C} \cup \mathcal{C}'$ je nekonzistentná.*

Idea dôkazu: ak by $\mathcal{C} \cup \mathcal{C}'$ bola konzistentná množina, potom podľa definície 4.48 by bolo $\mathcal{C} \cup \mathcal{C}' = \mathcal{C} = \mathcal{C}'$.

O vzťahu štandardných a obmedzovaných extenzií defaultovej teórie hovoria nasledujúce vety.

Tvrdenie 4.55 *Nech (E, D) je normálna defaultová teória. Potom $\mathcal{E}$ je jej extenzia práve vtedy, keď $(\mathcal{E}, \mathcal{E})$ je jej obmedzovaná extenzia.*

Ľahký dôkaz z faktu, že stráže a dôsledky sú pri normálnych defaultových pravidlách totožné.

Uvažovanie po prípadoch

Príklad 4.56 Predpokladajme, že $E = \{\alpha \vee \beta\}$ a $D = \{\alpha : \gamma/\gamma, \beta : \gamma/\gamma\}$. Bolo by intuitívne, keby γ patrilo do extenzie tejto teórie. Žiaľ, podľa Reiterovej teórie to nie je tak. ■

Príklad 4.57 Podobne, pre $E = \emptyset$, $D = \{\alpha : \beta/\beta, \neg\alpha : \beta/\beta\}$.

Keďže E je prázdna, ani jedno z pravidiel nie je aplikovateľné: ani predpoklad α, ani predpoklad $\neg\alpha$ nemôže byť splnený. ■

Klasická (Reiterova) formalizácia defaultového usudzovania teda nedovoľuje uvažovanie po prípadoch. V príkladoch 4.56 a 4.57 sme neboli schopní využiť, že platí nejaká disjunkcia a skúšať, čo dostaneme z predpokladu, že platia jednotlivé disjunkty: obvykle, ak dostaneme ten istý záver v oboch prípadoch, môžeme ho prijať nezávisle na tom, ktorý z disjunktov je pravdivý.

Aj tomuto problému sa úspešne venovali [DSJ 94]. Vyriešili ho tým, že uvažovali iba defaultové pravidlá bez predpokladov

Najprv si uvedieme systém *PfDL* (precondition-free default logic). Budeme sa venovať iba prípadu normálnych teórií. Ak máme normálnu defaultovú teóriu, transformujeme ju na bezpredpokladovú normálnu teóriu tak, že miesto každého pravidla tvaru $\alpha : \beta/\beta$ zavedieme pravidlo tvaru

$$\frac{: \alpha \to \beta}{: \alpha \to \beta}.$$

Pojem extenzie ostáva rovnako definovaný ako pre všeobecné defaultové teórie.

Príklad 4.58 Vráťme sa k príkladom 4.56 a 4.57.

4.56 $E = \{\alpha \vee \beta\}$,
$D' = \{: \alpha \to \gamma/\alpha \to \gamma, : \beta \to \gamma/\beta \to \gamma\}$. Ľahko možno preveriť, že $\mathcal{E} = Cn_{FOL}(\{\alpha \vee \beta, \gamma\})$ je extenzia defaultovej teórie (E, D'): Stráž nie je vyvrátená, $\alpha \wedge \neg\gamma$ neplatí v $\mathcal{E}$. Preto možno predpokladať $\alpha \to \gamma$. Podobne pre $\beta \to \gamma$. Teda, z $\alpha \vee \beta$ dostávame γ výrokovologickým vyplývaním.

4.57 $E = \emptyset$,
$D' = \{: \alpha \to \beta / \alpha \to \beta, : \neg\alpha \to \beta / \neg\alpha \to \beta\}$.

Zrejme $\alpha \vee \neg\alpha \ \in Cn_{FOL}(\emptyset)$. Preto $Cn_{FOL}(\{\beta\})$ je extenziou tejto defaultovej teórie. ■

PfDL má niektoré užitočné vlastnosti implikácie:

Príklad 4.59 Často pri klasickom odvodzovaní využívame to, že $p \to q$ je ekvivalentné s kontrapozíciou $\neg q \to \neg p$.

Defaultová teória $(\emptyset, \{p : q/q, p : \neg q/\neg q\})$ má jedinú extenziu $Cn_{FOL}(\emptyset)$, lebo žiadne z defaultových pravidiel nie je aplikovateľné.

Extenziou bezpredpokladového variantu $(\emptyset, \{: p \to q/p \to q, : p \to \neg q/p \to \neg q\})$ tejto defaultovej teórie je $Cn_{FOL}(\{\neg p\})$.

Aplikovateľné sú obe pravidlá. Kontrapozície ich dôsledkov sú $\neg q \to \neg p$, $\neg\neg q \to \neg p$. Na základe platnosti $\neg q \vee \neg\neg q$ dostávame $\neg p$. (Je zrejmé, že platnosť $\neg p$ zaručuje platnosť $p \to q$ aj $p \to \neg q$.)

Podobne, $(\{\neg q\}, \{p : q/q\})$ má extenziu $Cn_{FOL}(\{\neg q\})$, ale $(\{\neg q\}, \{: p \to q/p \to q\})$ má extenziu $Cn_{FOL}(\{\neg q, \neg p\})$. ■

Samozrejme, aj pri bezpredpokladových defaultových teóriách, je zaujímavé študovať obmedzované extenzie. Syntakticky sa tvar defaultových teórií nelíši od *PfDL* a pojem obmedzovanej extenzie dedíme z *ConDL*. Aj túto verziu obmedzovanej defaultovej logiky, *PfConDL*, študovali [DSJ 94].

4.1.3 Sémantika

Sémantická špecifikácia usudzovania o typických prípadoch (vlastnostiach, stavoch) je možná v rôznych úrovniach.

V istom zmysle slova sa dajú aj extenzie považovať za sémantickú charakterizáciu defaultových teórií. Už vieme, že extenzie predstavujú množinu presvedčení, ktorým možno súčasne veriť, ak akceptujeme nejakú defaultovú teóriu. Extenzia teda v istom zmysle slova predstavuje význam defaultovej teórie. Ľahko si možno predstaviť prístup, keď najprv špecifikujeme množinu (resp. alternatívne množiny) viet, ktoré pokladáme za vieryhodné. Na základe tejto špecifikácie skonštruujeme defaultovú teóriu, pozri [Mar 97], aj časť 4.1.4.

Tu však uprednostníme teoreticko-modelovú sémantiku. To znamená, že v termínoch tejto sémantiky charakterizujeme defaultové teórie a ich extenzie (a tak získame meradlo korektnosti či úplnosti). Prístup orientovaný na teoreticko-modelovú sémantiku zaviedli a preferovali viacerí autori, napríklad [Luk 90, Eth 88, Lif 90, SKa 90, DSJ 94, RoZ 95].

Budeme ďalej predpokladať defaultovú teóriu $T = (E, D)$. Triedu všetkých modelov množiny viet E označíme ako $Mod(E)$. Defaultové pravidlá obvykle rozširujú množinu akceptovateľných viet: k vetám z E pridávajú dôsledky defaultových pravidiel. Rozšíreniu množiny viet zodpovedá zmenšenie množiny modelov: Niektoré modely množiny E prestávajú byť modelmi rozšírenej množiny viet, pretože dôsledky použitých defaultových viet v nich nemusia byť splnené.

$Mod(E)$ zredukujeme na množinu všetkých modelov z $Mod(E)$, v ktorých sú splnené dôsledky defaultových pravidiel. Táto redukovaná množina modelov je v istom zmysle slova „lepšia“, presnejšie zodpovedá defaultovým pravidlám. Zavedieme preto nejakú preferenciu (usporiadanie) na modeloch.

Prechádzame k prezentácii sémantickej charakterizácie defaultových teórií a ich extenzií. Jej základy podal [Eth 88].

Vyjdeme z myšlienky, že defaultové pravidlo $\alpha : \beta/\gamma$ sa dá považovať za rozumné, ak existuje model m taký, že $m \models \alpha$, $m \not\models \neg\beta$ a $m \models \gamma$. Budeme však hovoriť o množinách modelov, pretože extenzia má obvykle celú triedu modelov.

Ponúka sa nám teda myšlienka, že bude rozumné rozlišovať lepšie a horšie množiny modelov. Zadefinujeme reláciu preferencie. Najprv budeme preferovať množiny modelov z hľadiska jediného pravidla d. Výraz $M_1 \leq_d M_2$ môžeme čítať: množina modelov M_1 je vzhľadom na pravidlo d „lepšia“ ako množina modelov M_2. Analogicky pre množinu defaultových pravidiel D a pre označenie $\leq_D$.

Definícia 4.60 Nech $\mathcal{M}$ je ľubovoľná trieda modelov, $\mathcal{M} \subseteq Mod(E)$. Predpokladajme, že sú dané množiny modelov, $M_1, M_2 \in 2^{\mathcal{M}}$ a pravidlo $d = \alpha : \beta/\gamma$ je z D.

Potom $M_1 \leq_d M_2$ práve vtedy, keď

(1) $\forall m \in M_2\ (m \models \alpha)$,

(2) $\exists m \in M_2\ (m \models \beta)$,

(3) $M_1 = \{m \in M_2 : m \models \gamma\}$.
M_1 budeme označovať aj ako $d(M_2)$.

Budeme hovoriť, že defaultové pravidlo $d = \alpha : \beta/\gamma$ je *aplikovateľné* na triedu modelov M, ak sú splnené podmienky (1) a (2). ■

Spomedzi dvoch tried modelov, v ktorých sú splnené podmienky (1) a (2) je lepšia (vzhľadom na defaultové pravidlo d) tá trieda modelov, v ktorej je splnené $cons(d)$.

Definícia 4.61 $M_1 \leq_D M_2$ práve vtedy, keď $\exists d \in D\ (M_1 \leq_d M_2)\ \vee\ \exists M'\ (M_1 \leq_D M' \leq_D M_2)$ ■

Vzťah $\leq_d$ a pojem aplikovateľnosti defaultového pravidla na triedu modelov však trpia tým istým druhom chýb ako naivné odvodzovanie pomocou defaultových pravidiel. Môže sa totiž stať, že $M_1 \leq_d M_2$, ale $\neg\exists m \in M_1\ (m \models just(d))$. Teda, po aplikácii nejakého defaultového pravidla d na množinu modelov M sa môže stať, že stráže $just(d)$ už nie sú splniteľné v $d(M)$. Preto d nie je aplikovateľné na triedu modelov $d(M)$.[11] Túto nepríjemnú vlastnosť nemajú stabilné množiny modelov, s ktorými sa zoznámime ďalej. Semimonotónne teórie sú na tom opäť lepšie: minimum v usporiadaní $\leq_D$ u nich vedie k extenzii, pozri príklad 4.62 a tvrdenie 4.63.

Príklad 4.62 Pripomenieme si príklad 4.38:

$$a : d/d,$$
$$d : z/z,$$
$$a : \neg z/\neg z.$$

Okrem toho prijmime, že $E = \{a\}$.

Obmedzíme sa na Herbrandove interpretácie (a modely). $Mod(E) = \{\{a\}, \{a, d\}, \{a, z\}, \{a, d, z\}\}$, pričom neprítomnosť nejakej výrokovej premennej v interpretácii chápeme ako splnenie jej negácie v danej interpretácii (ide o 2-interpretácie).

Jednoprvkové množiny modelov $\{\{a, d, z\}\}$ a $\{\{a, d\}\}$ sú minimálne vzhľadom na $\leq_D$ (poznamenajme, že $\{a, d\}$ predstavuje $\{a, d, \neg z\}$). Obe sú extenziami uvedenej normálnej defaultovej teórie. Ľahko sa možno presvedčiť, že pre ľubovoľné modely M_1 a M_2 tejto defaultovej teórie a pre žiadne normálne defaultové pravidlo d nemôže súčasne platiť, že $M_1 \leq_d M_2$ a $\neg\exists m \in M_1$ $(m \models just(d))$, pretože $just(d) = cons(d)$. ■

Množinu všetkých viet (uzavretých prvorádových formúl) pravdivých v každom modeli triedy modelov M budme značiť ako $Th(M)$.

Tvrdenie 4.63 *Nech (E, D) je semimonotónna defaultová teória. Ak M_{min} je minimum v triede $Mod(E)$ vzhľadom na usporiadanie $\leq_D$, potom $Th(M_{min})$ je extenziou (E, D).*

Dôkaz je založený na ľahko preveriteľnom fakte, že pre triedu modelov M, kde $Th(M)$ je extenziou semi-monotónnej defaultovej teórie (E, D), platí: ak $d \notin D$, potom $Th(d(M))$ je extenziou $(E, D \cup \{d\})$.

Vo všeobecnom prípade však musíme byť opatrnejší.

Príklad 4.64 V úspornejšej notácii si pripomenieme príklad 4.9. $E = \{m(e, f), b(e) = t, z(f, d), s(d) = n, t \neq n, (b(Y) = Z \wedge b(Y) = X) \rightarrow X =$

[11] Keď prejdeme do reči extenzií, každé defaultové pravidlo, ktoré prispeje k vytvoreniu extenzie, je na ňu aj aplikovateľné. Preto s jednoduchými aplikáciami defaultových pravidiel na triedy modelov nevystačíme pri teoreticko-modelovej charakterizácii extenzií.

$Z\}$.[12] $D =$

$$\begin{aligned}\{d_1 &= \frac{m(X,Y), b(X)=Z : b(Y)=Z}{b(Y)=Z},\\ d_2 &= \frac{z(X,Y), s(Y)=Z : b(X)=Z}{b(X)=Z},\\ d_3 &= \frac{: b(X)\neq n}{\neg c(X)}\}.\end{aligned}$$

Štartujme teraz od $Mod(E)$, od množiny všetkých modelov E. Postupnou aplikáciou defaultových pravidiel redukujme túto množinu modelov. Model budeme stotožňovať s množinou základných atómov pravdivých v tomto modeli. Treba si uvedomiť, že do $Mod(E)$ patria aj také modely m, v ktorých máme voľnosť, pokiaľ ide o pravdivostné hodnoty viet $c(e), c(f), b(f) = t$, $z(e,d)$.

$$M = d_3(Mod(E)) = Mod(E) \setminus \{m : m \not\models \neg c(f)\} \setminus \{m : m \not\models \neg c(e)\}$$
$$M' = d_2(M) = M \setminus \{m : m \not\models b(f) = n\}$$

Ďalej už nemôžeme aplikáciou defaultových pravidiel získať „lepšiu“ (menšiu) triedu modelov. Budú totiž narušené stráže každého defaultového pravidla (okrem d_2, ktoré však nemôže priniesť nič nového). Žiaľ, M' nie je triedou modelov nejakej extenzie vyššie uvedenej defaultovej teórie, hoci je minimálnou množinou modelov. M' obsahuje iba modely, v ktorých platí $b(f) = n$. Súčasne v nich však platí $\neg c(f)$. V žiadnej extenzii ale nemôžu platiť súčasne obe vety. To znamená, že M' nie je triedou všetkých modelov nejakej extenzie. ■

Videli sme teda, že minimálna množina modelov vo všeobecnosti nestačí na charakterizáciu extenzie. Preto zavedieme pojem stability. Základná myšlienka, ktorú sledujeme: budú nás zaujímať triedy modelov, ktoré popri tom, že sú prefererované vzhľadom na $\leq_d$ (alebo $\leq_D$), umožňujú aj aplikáciu d (všetkých pravidiel z D). Takéto triedy modelov budeme považovať za stabilné.

Definícia 4.65 (Stabilná množina modelov) Nech $M \in 2^{Mod(E)}$. M je *stabilná* pre $T = (E, D)$ práve vtedy, keď existuje množina pravidiel D',

[12] Posledné dve vety z E sa v príklade 4.9 nevyskytovali. Tu sú doplnené kvôli korektnej dedukcii z E, čitateľ iste vidí, že sú dostatočne intuitívne. Považujeme ich za skratky všetkých relevantných inštancií (pre konštanty tu použité: e, f, t, d, n). Podobne defaultové pravidlá považujeme za reprezentantov ich rozumných inštancií. „Rozumných“ preto, lebo mlčky predpokladáme nejakú typovú disciplínu, neočakávame atómy ako $z(d,n), b(t) = e, c(t)$ atď.

pričom

$$\begin{array}{rcl} D' & \subseteq & D, \\ M & \leq_{D'} & Mod(E), \\ \forall d \in D' \; \exists m \in M & & (m \models just(d)). \end{array}$$

■

Fakt 4.66 *Nech v (E, D) platí semimonotónnosť. Potom*

- *$Mod(E)$ je stabilná pre (E, D),*
- *ak M je stabilná pre (E, D), $d \in D$ a $M' \leq_d M$, potom M' je stabilná pre (E, D).*

Dôkaz: Uvažujeme prázdnu množinu defaultových pravidiel. Určite platí $Mod(E) \leq_\emptyset Mod(E)$ a $\forall d \in \emptyset \; \exists m \in Mod(E) \; m \models just(d)$. (Stačí zvážiť, že ide o skratku za $\forall d \; \exists m \; (d \in \emptyset \wedge m \in Mod(E) \to m \models just(d))$ a ľavá strana tejto implikácie je vždy nepravdivá.) Teda $Mod(E)$ je stabilná množina modelov pre ľubovoľné (E, D).

Nech M je stabilná pre (E, D), pričom $M \leq_{D'} Mod(E)$. Nech $d(M) = M'$. Teda platí $M' \leq_d M$, preto dostávame $M' \leq_{D' \cup \{d\}} Mod(E)$. Zo semimonotónnosti dostávame $\exists m \in M' \; m \models just(d)$. Inak by totiž extenzia teórie $(E, D' \cup \{d\})$ nebola nadmnožinou extenzie teórie (E, D').

Tvrdenie 4.67 (Korektnosť) *Nech $T = (E, D)$. Ak $\mathcal{E}$ je extenziou T, tak $M_s = \{m : m \models \mathcal{E}\}$ je stabilná pre T a neexistuje množina modelov M stabilná pre T taká, že $M \leq_D M_s$.*

Odkazujeme na podrobný dôkaz v [Luk 90]. Dobrý námet je využiť množinu generujúcich defaultov $GD(\mathcal{E}, T)$ extenzie $\mathcal{E}$ a zvážiť, že

$$\mathcal{E} = Th(E \cup cons(GD(\mathcal{E}, T))).$$

Po zavedení vhodného usporiadania na generujúcich defaultoch dostaneme stabilitu a minimalitu M_s.

Tvrdenie 4.68 (Úplnosť) *Nech $T = (E, D)$, M je stabilná pre T a neexistuje množina modelov M' stabilná pre T taká, že $M' \leq_D M$.*

Potom $M = Mod(\mathcal{E})$, kde $\mathcal{E}$ je nejaká extenzia T.

Opäť odkazujeme na dôkaz v [Luk 90]. Na základe definície stabilnosti platí $\exists D' (D' \subseteq D \;\wedge\; M \leq_{D'} Mod(E))$. Dobré je nájsť minimálne D' s touto vlastnosťou a uvedomiť si, že možno vytvoriť nejaké usporiadanie na D', ktoré zodpovedá reťazi vedúcej od $Mod(E)$ k M.

Príklad 4.69 Nech $T = (\emptyset, \{: \neg p/p\})$. Od $Mod(\emptyset)$ sa pomocou daného defaultového pravidla dostaneme iba k $\{m : m \models p\}$, táto množina modelov však nie je stabilná pre T. ■

Príklad 4.70 Nech $T = (\{p\}, \{p : q \wedge \neg r/q, p : r \wedge \neg q/r\})$. Množiny modelov $\{m : m \models p \wedge q\}$ a $\{m : m \models p \wedge r\}$ sú minimálne a stabilné pre T. ■

Videli sme (tvrdenia 4.67 a 4.68), že pojem stabilnosti nestačí na charakterizáciu extenzie. Iba vtedy, keď nájdeme minimálnu stabilnú množinu modelov, vieme charakterizovať extenziu. Teraz prejdeme k sémantickej charakterizácii obmedzovaných extenzií. Dvojici množín viet $(\mathcal{E}, \mathcal{C})$ zodpovedá dvojica množín ich modelov (M, M'), kde

- M splňuje teóriu, vytvorenú vetami z E a dôsledkami defaultových pravidiel aplikovateľných na $\mathcal{E}$,
- M' navyše splňuje aj stráže defaultových pravidiel.

Takéto dvojice tried modelov sa volajú v [DSJ 94] focused model structures.

Definícia 4.71 Nech $d \in D$ a $\mathcal{M}_1 = (M_1, M_1')$, $\mathcal{M}_2 = (M_2, M_2')$ sú dvojice tried modelov. Voláme ich *zaostrenými modelovými štruktúrami*. Potom

a) $\mathcal{M}_1 \leq_d \mathcal{M}_2$ práve vtedy, keď

$$\begin{array}{rcl} \forall m \in M_2 & & (m \models pre(d)), \\ \exists m \in M_2' & & (m \models just(d) \wedge cons(d)), \\ M_1 & = & \{m \in M_2 : m \models cons(d)\}, \\ M_1' & = & \{m \in M_2' : m \models just(d) \wedge cons(d)\}. \end{array}$$

b) $\mathcal{M}_1 \leq_D \mathcal{M}_2$ práve vtedy, keď

- $\exists d \in D(\mathcal{M}_1 \leq_d \mathcal{M}_2)$ alebo
- $\exists \mathcal{M}_3(\mathcal{M}_1 \leq_D \mathcal{M}_3 \wedge \mathcal{M}_3 \leq_D \mathcal{M}_2)$.

c) *Preferované* zaostrené modelové štruktúry sú $\leq_D$-minimálne dvojice priestoru $2^{Mod(E)} \times 2^{Mod(E)}$.

■

Tvrdenie 4.72 (Korektnosť a úplnosť [DSJ 94]) *Nech $T = (E, D)$ je defaultová teória, (M, M') je dvojica tried modelov, $\mathcal{E} = Th(M)$ a $\mathcal{C} = Th(M')$.*

Potom $(\mathcal{E}, \mathcal{C})$ je obmedzovaná extenzia T práve vtedy, keď (M, M') je $\leq_D$-minimálna v priestore $2^{Mod(E)} \times 2^{Mod(E)}$.

Poznámka 4.73 Pojem stability alebo jeho analógiu nepotrebujeme zavádzať vďaka semi-monotónnosti obmedzovaných extenzií. ■

Zhrňme sémantické konštrukcie:

Boli vcelku priamočiare. Zavedením usporiadania na modeloch defaultových teórií sme získali ďalší príklad minimálnych modelov, resp. minimálneho vyplývania. Ďalej sme získali možnosť adekvátne charakterizovať *otvorené* defaultové teórie.

Doposiaľ sme teoretické výsledky o defaultových teóriách uvádzali iba pre uzavretý prípad (ak sa nejaké premenné vyskytujú, sú viazané kvantifikátormi). Intuitívne sme pracovali aj s otvorenými defaultovými pravidlami, vždy sme však mlčky predpokladali, že ich môžeme považovať za skratky mnohých inštancií týchto pravidiel. Naposledy sme tak postupovali v príklade 4.64.

Lifschitz však v [Lif 90] ukazuje, že tento prístup nie je postačujúci:

Príklad 4.74 V defaultovej teórii $T = (\{p(a)\}, \{: \neg p(X)/\neg p(X)\})$ nie je dokázateľné $\forall X(p(X) \equiv x = a)$.

Podobne $\{\neg na(b, b_1), \ldots, \neg na(b, b_n)\}$ je slabšie ako

$$\forall X \neg na(b, X),$$

pokiaľ neprijmeme predpoklad uzavretosti domén (DCA): $\forall X(X = b_1 \vee \cdots \vee X = b_n))$. ■

Lifschitz [Lif 90] charakterizoval otvorené defaultové teórie teoreticko-modelovými konštrukciami. Predpokladal triedy modelov so spoločným univerzom a navyše predpokladal, že každý objekt z univerza má meno.

4.1.4 Komentáre

Nasledujúce poznámky sú orientované na čitateľa s hlbším záujmom o defaultové teórie. Ich cieľom je uľahčiť orientáciu v problematike, ktorá nebola zahrnutá do predchádzajúcich častí. V tejto časti podám takmer iba bibliografické údaje o výskume a návrhoch zameraných na prekonanie problematických stránok defaultových teórií. Príkladmi takýchto nedostatkov sú napríklad nedostatočná kontrola (globálnej) konzistentnosti či nekumulatívnosť. Trochu väčšiu pozornosť budeme venovať iba vzťahu defaultových teórií k revíziám.

Reprezentatívnou a často citovanou monografiou o nemonotónnom usudzovaní je [MaT 93], zodpovedá stavu zo začiatku deväťdesiatych rokov, základné, klasické výsledky, boli v tom čase už dosiahnuté.

Besnard a Schaub v [BeS 94] zavádzajú Kripkeho štruktúry (pozri časť 2.2, kde sme si zaviedli ich verziu). Vychádzajú pritom z myšlienky zaostrených modelových štruktúr, diskutovaných v časti 4.1.3. Dvojica množín modelov prirodzene vedie k relácii dostupnosti v Kripkeho štruktúre. Takto definovaná sémantika dovoľuje charakterizovať nielen obmedzované extenzie [DSJ 94], ale aj klasický pojem extenzie [Rei 80] a niektoré návrhy riešení neintuitívností Reiterovej formalizácie – Brewkovu kumulatívnu defaultovú logiku [Bre 91] a Lukasiewiczov prístup [Luk 88]. Navyše, dokáže aj porovnať spomínané návrhy.

Nait Abdallah [NaA 89] sa tiež venuje problémom spätým s disjunktívnou informáciou, s analýzou prípadov, s uplatnením kontrapozície v defaultových teóriách. Ako riešenie predkladá originálny, treba zdôrazniť, že monotónny, formalizmus, ktorý zachováva platnosť teorémy dedukcie.

Ďalší prístup k riešeniu týchto ťažkostí ponúkajú Mikitiuk a Truszczynski [MiT 93, MiT 95]. Navrhujú racionálnu defaultovú logiku. Táto sa pre semi-normálne defaultové teórie zhoduje s obmedzovanou defaultovou logikou. Pojem racionálnej extenzie aplikovali aj na analýzu disjunktívnych logických programov.

Defaultové usudzovanie možno chápať ako špeciálny prípad riešenia nekonzistentností (revízie sú jedným z jeho aspektov, čo nás už neprekvapuje). Zaujímavý pohľad na defaultové usudzovanie poskytuje [Bre 89], ktorý rozvíjal motívy z [Poo 88]. Poole konštatoval, že na formalizáciu defaultového usudzovania netreba novú logiku, ale stačí zmena pohľadu. Na usudzovanie defaultového typu nazerá ako na veľmi jednoduchý prípad tvorenia teórií, kde defaulty sú preddefinované možné hypotézy.

Brewka tiež vychádza z rozlíšenia faktov (teória E, na ktorú sa aplikujú defaultové pravidlá) a hypotéz (dôsledkov defaultových pravidiel). Za charakteristický postoj k týmto dvom triedam viet považuje reakcie v prípade objavenia sa nekonzistentností v množine odvodených viet: ak sa objaví nekonzistentnosť, skôr sme ochotní vzdať sa defaultov ako faktov. V prípade konzistentnej množiny viet E sú extenzie defaultovej teórie (E, D) vlastne maximálne konzistentné množiny deduktívneho uzáveru E a dôsledkov defaultových pravidiel. Brewka túto ideu rozšíril – nejde mu o všetky možné maximálne konzistentné množiny presvedčení, ale iba o tie, ktoré sa z nejakého hľadiska považujú za preferované.

Za defaultovú teóriu považuje n-ticu teórií

$$T = (T_1, \ldots, T_n).$$

Každá z týchto teórií je množinou prvorádových formúl. Ich dôveryhodnosť klesá s rastúcim indexom.

$$S = S_1 \cup \cdots \cup S_n$$

je preferovaná podteória T, ak pre každé k platí, že $S_1 \cup \cdots \cup S_k$ je *maximálna konzistentná podmnožina* $T_1 \cup \cdots \cup T_k$. Keď si uvedomíme, že v definícii sa vlastne predpokladá konštrukcia od $k = 1$ až po $k = n$, vidíme, že preferovaná podteória obsahuje maximálnu konzistentnú podmnožinu T_1, teda zachová čo najviac z tých najdôveryhodnejších viet, a potom postupne z každej T_{i+1} vyberá maximálnu podmnožinu S_{i+1}, ktorou možno konzistentne doplniť $S_1 \cup$

$\cdots \cup S_i$. Prirodzené zovšeobecnenie možno dosiahnuť, ak sa usporiadanie teórií zmení na čiastočné.

Defaultové teórie tvaru $(T_1, \ldots, T_n)$ možno vyjadriť ako istý typ defaultových teórií s prioritami [Bre 89]. Tie sa definujú ako $(E, D_1, \ldots, D_n)$, kde D_i sú disjunktné množiny defaultových pravidiel a extenzia $\mathcal{E}$ sa konštruuje iteratívne: $\mathcal{E}_1$ je extenzia (E, D_1), $\mathcal{E}_2$ je extenzia $(\mathcal{E}_1, D_2)$, atď. až $\mathcal{E} = \mathcal{E}_n$ je extenzia $(\mathcal{E}_{n-1}, D_n)$.

Vzťahu revízií a defaultových teórií sa venovali aj Yuasa a Arikawa. Zdôrazňujú, že extenzia predstavuje iba predbežné poznanie a venujú sa problému modifikácie extenzií [YuA 88]. Revíziám defaultových teórií sa venujú v najnovších výskumoch aj [WiA 98, AGG 99].

Rounds a Zhang [RoZ 95, RoZ 97a, RoZ 97b, RoZ 97c, RoZ 97d] kombinujú Scottove myšlienky (teória domén, sémantika programovacích jazykov) a výsledky z oblasti nemonotónnej logiky. Vychádzajú z presvedčenia, že nie je adekvátne založiť defaultové usudzovanie na odvodzovacích pravidlách (ako to urobil Reiter a jeho nasledovníci). Formálny dôkaz nie je dobrá cesta na pochopenie a charakterizáciu defaultov. Dôraz kladú na sémantickú charakterizáciu. Ich cieľom je na báze dôkladnej teórie neúplnej informácie poskytnúť sémantiku defaultového usudzovania. Defaulty chápu ako algoritmy na konštrukciu (čiastočných) modelov (v kontraste k odvodzovacím pravidlám).[13]

Alternatívnosť extenzií vedie aj k tejto otázke: Predstavme si, že máme triedu alternatívnych teórií (deduktívne uzavretých množín viet). Tieto teórie môžu predstavovať alternatívne riešenia nejakého problému. Prirodzená je otázka, či sme schopní vytvoriť defaultovú teóriu, ktorej extenziami sú presne spomínané teórie. V takom prípade by daná defaultová teória dobre reprezentovala východzie alternatívne predstavy. Odpoveď na túto otázku je negatívna, pozri [Mar 97]. Niektoré pozitívne prípady našiel [Sat 00]. Príklady reprezentácií alternatívnych množín viet defaultovými teóriami možno nájsť v [CMM 95].

Defaultovými teóriami možno formalizovať usudzovanie (resp. rôzne triedy úsudkov) o typickom, obvyklom stave vecí v situácii, keď nám chýbajú kompletné informácie. Predstavu o súčasne udržateľných presvedčeniach, opretých o defaultové predpoklady dáva pojem extenzie defaultovej teórie (resp. rôzne jeho alternatívy – akou je napríklad obmedzovaná extenzia).

Videli sme, že niektoré defaultové teórie nemajú extenziu. Na druhej strane extenzia nemusí byť určená jednoznačne. Defaultové teórie teda umožňujú nahliadnuť do alternatívnej povahy hypotetického usudzovania. Zvláštny záujem si zaslúžia normálne defaultové teórie, pre ktoré je zaručená existencia extenzie. Tieto teórie sú aj semi-monotónne: pre danú množinu východiskových viet vedie rozširovanie množiny normálnych defaultových pravidiel

[13] Pod čiastočným modelom môžeme rozumieť taký model (T, F), kde $T \cup F$ nemusí vyčerpávať množinu všetkých základných atómov jazyka.

k rozširovaniu extenzie.

Pôvodná formalizácia defaultových teórií vedúca k pojmu extenzie naráža na mnohé problémy zásadného významu – ide o problémy s vypočítateľnosťou, vzájomnými interakciami pravidiel, lokálnou a nie globálnou kontrolou konzistentnosti a neschopnosťou robiť analýzu alternatív (možných prípadov). Niektoré z týchto problémov rieši zavedenie obmedzovaných extenzií. Existencia obmedzovaných extenzií je zaručená pre každú defaultovú teóriu a platí pre ne aj semi-monotónnosť.

Na záver sme podali sémantickú charakterizáciu defaultových teórií, extenzií a obmedzovaných extenzií.

4.2 Odvodzovanie v hierarchických sieťach

Človek si poznatky triedi a hierarchizuje. Vytvorenú hierarchiu využíva pri lepšej orientácii vo svojich znalostiach, pri usudzovaní. Usudzovanie o hierarchicky usporiadaných sústavách poznatkov i samotná organizácia hierarchií sú preto pre umelú inteligenciu zaujímavými a dôležitými problémami.

Čím je však štruktúra hierarchie komplikovanejšia, tým je aj usudzovanie problematickejšie. Niekedy sa v hierarchii vyskytujú výnimky a z nadradených úrovní hierarchie možno odvodiť konfliktné závery pre niektorú z nižších úrovní. V takých situáciách sa prirodzene využíva usudzovanie o typických, obvyklých prípadoch.

V predchádzajúcej časti sme sa oboznámili s defaultovými teóriami. Teraz ukážeme, ako ich možno aplikovať na hierarchie a odvodzovanie v nich. Bude nás zaujímať, či defaultové teórie poskytujú optiku, pomocou ktorej sa dá priblížiť k vyriešeniu problému výnimiek a konfliktov. Popri aparáte defaultových teórií spomenieme aj teórie, v ktorých sa dedenie definuje pomocou nejakých ciest v grafe a na záver sa budeme venovať možnosti charakterizovať dedenie pomocou štvorhodnotovej logiky QRI, ktorá neumožňuje odvodiť zo sporu čokoľvek.

Pochopenie a využívanie hierarchických vzťahov je dôležitým aspektom reprezentácie znalostí. Ak si pripomenieme McCarthyho rozlíšenie z [McC 77], hierarchické vzťahy sú zaujímavé aj z epistemologického, aj z heuristického hľadiska. Hierarchická stavba ľudského poznania je jednou z jeho charakteristických čŕt a nie je neprirodzené chcieť, aby podobnú štruktúru mali aj znalosti reprezentované pre výpočtové systémy. Navyše, využívanie hierarchickej organizácie poznatkov vedie k odvodzovaniu, ktoré je výpočtovo efektívnejšie.

4.2.1 Automatické dokazovanie a inferencia na typoch

Najprv si ukážeme, ako zavedenie hierarchickej disciplíny do obyčajného automatického dokazovača zvyšuje jeho výkonnosť. Ilustrujeme tým to, čo sme už vyslovili: hierarchická organizácia poznatkov podporuje lepšiu orientáciu v nich, usudzovanie. Potom prejdeme priamo k usudzovaniu v hierarchicky organizovaných sieťach. Špeciálne nás bude zaujímať problém dedenia vlastností z vyšších vrstiev hierarchie na nižšie, a predovšetkým ošetrovanie prípadných konfliktov.

Rozlišovanie typov a hierarchických vzťahov medzi nimi zjednoduší činnosť klasického dokazovača.[14] Postupne si budeme rozoberať príklad, ktorý je

[14] Základnú informáciu o klasickom prístupe k dokazovaniu teorém možno nájsť v dodatku D.

ťažký pre štandardný dokazovač, ale podstatne jednoduchší po zavedení hierarchicky usporiadaných typov.

Príklad 4.75 Formulácia problému (hlavolamu) [Wal 85]:

Vlci, líšky, vtáci, húsenice a slimáci sú zvieratá a existuje aspoň jeden exemplár z každého uvedeného druhu. Podobne existuje obilie. Obilie je rastlina. Každé zviera sa živí buď rastlinami alebo takými zvieratami, ktoré sú oveľa menšie a rady jedia rastliny. Húsenice a slimáky sú oveľa menšie ako vtáci, tí sú oveľa menší ako líšky, tie zas ako vlci. Vlci neradi konzumujú líšky a obilie. Vtáci jedia radi húsenice, ale nie slimáky. Húsenice a slimáky rady jedia nejaké rastliny.

Z toho vyplýva: Existuje zviera, ktoré rado je také zviera, čo sa kŕmi obilím. ■

Príklad 4.76 Teraz si túto úlohu sformulujeme v štandardnom jazyku predikátového počtu. Rozumné bude dohodnúť sa, že v tejto chvíli je pre nás vedľajší problém adekvátneho vyjadrenia viet prirodzeného jazyka prostriedkami logiky. Jednoducho, nasledujúcu Schubertovu formalizáciu [Wal 85] budeme chápať ako štartovný bod ďalšieho výkladu.

$$
\begin{array}{rlrl}
(1) & \exists X w(X) & (2) & \exists X f(X) \\
(3) & \exists X b(X) & (4) & \exists X c(X) \\
(5) & \exists X s(X)\} & (6) & \exists X g(X) \\
(7) & \forall X(w(X) \rightarrow a(X)) & (8) & \forall X(f(X) \rightarrow a(X)) \\
(9) & \forall X(b(X) \rightarrow a(X)) & (10) & \forall X(c(X) \rightarrow a(X)) \\
(11) & \forall X(s(X) \rightarrow a(X)) & (12) & \forall X(g(X) \rightarrow p(X))
\end{array}
$$

Iba kvôli pohodliu zamýšľané významy predikátov (symboly zodpovedajú prvým písmenám anglických pomenovaní spomínaných živočíšnych a rastlinných druhov): w – vlk, f – líška, b – vták, c – húsenica, s – slimák, g – obilie, a – zviera, p – rastlina. $\exists X w(X)$ čítame ako *existuje* X*, ktoré je vlkom* – teda, existuje nejaký vlk.

$$
\begin{array}{l}
(13) \forall X_1, X_2, X_3, X_4 (a(X_1) \wedge p(X_2) \wedge a(X_3) \wedge p(X_4)) \rightarrow \\
\quad (e(X_1, X_2) \vee ((m(X_3, X_1) \wedge e(X_3, X_4)) \rightarrow e(X_1, X_3)))
\end{array}
$$

Zamýšľaný význam novozavedených predikátov: m – byť menší, e – jesť (sú to dvojargumentové predikáty – niečo je menšie ako niečo iné, niečo rado konzumuje čosi). Čítanie: ak X_1 a X_3 sú zvieratá, X_2 a X_4 sú rastliny, potom buď X_1 konzumuje rastliny alebo zvieratá, ak sú menšie a jedia rastliny.

(14) $\forall X_1, X_2((c(X_1) \wedge b(X_2)) \rightarrow m(X_1, X_2))$
(15) $\forall X_1, X_2((s(X_1) \wedge b(X_2)) \rightarrow m(X_1, X_2))$
(16) $\forall X_1, X_2((b(X_1) \wedge f(X_2)) \rightarrow m(X_1, X_2))$
(17) $\forall X_1, X_2((f(X_1) \wedge w(X_2)) \rightarrow m(X_1, X_2))$
(18) $\forall X_1, X_2((w(X_1) \wedge f(X_2)) \rightarrow \neg e(X_1, X_2))$
(19) $\forall X_1, X_2((w(X_1) \wedge g(X_2)) \rightarrow \neg e(X_1, X_2))$
(20) $\forall X_1, X_2((b(X_1) \wedge c(X_2)) \rightarrow e(X_1, X_2))$
(21) $\forall X_1, X_2((b(X_1) \wedge s(X_2)) \rightarrow \neg e(X_1, X_2))$
(22) $\forall X_1(c(X_1) \rightarrow \exists X_2(p(X_2) \wedge e(X_1, X_2)))$
(24) $\forall X_1(s(X_1) \rightarrow \exists X_2(p(X_2) \wedge e(X_1, X_2)))$

Čítanie: (14): Pre každé X_1 a X_2, ak X_1 je húsenica a X_2 je vták, potom X_1 je menší ako X_2. Podobne sa čítajú (15) – (17). (18): Pre každé X_1 a X_2, ak X_1 je vlk a X_2 je líška, potom X_1 nemá rado X_2 (ako potravu). Analogicky pre (19) – (21). 22 (a 24): Pre každé X_1, ak X_1 je húsenica, potom existuje X_2, ktoré je rastlina a X_1 ho je. Dôvod, prečo je formula (23) vynechané, sa ozrejmí ochvíľu.

Napokon, treba dokázať: Existuje také zviera X_1, ktoré rado zje nejaké zviera X_2, čo rado zje každé obilie.

$$(26)\exists X_1, X_2, \forall X_3(a(X_1) \wedge a(X_2) \wedge (g(X_3) \rightarrow (e(X_2, X_3) \wedge e(X_1, X_2))))$$

■

Úlohou (pre automatický dokazovač) je preveriť, či z daných predpokladov vyplýva veta (26). Nasleduje preklad Schubertovej reprezentácie problému do klauzálnej syntaxe. Formula, ktorú treba dokázať, sa neguje a po jej pripojení k predpokladom je cieľom demonštrovať nekonzistentnosť vytvorenej množiny viet. Pravidlá prevodu do klauzálneho tvaru možno nájsť v dodatku D, načrtneme si ich však aj v nasledujúcom príklade.

Príklad 4.77 Klauza je disjunkcia literálov. Budeme používať obvyklé značenie – klauzu budeme reprezentovať množinou literálov. Uvádzané klauzy sú v skolemovskej normálnej forme – existenčné kvantifikátory sú nahradené buď skolemovskými konštantami ($s_w, s_f, s_b, s_c, s_s, s_g$) alebo funkčnými symbolmi (h, i, j). Využijeme nasledujúce transformácie: $\exists X w(X)$ na $w(s_w)$, $p \rightarrow q$ na $\neg p \vee q$, $\neg(p \wedge q)$ na $\neg p \vee \neg q$, $\forall X \exists Y r(X,Y)$ na $\forall X r(X, h(X))$, ďalej distributívnosť disjunkcie a konjunkcie. Zo značenia vypustíme všeobecné kvantifikátory a každú premennú budeme považovať za implicitne viazanú všeobecným kvantifikátorom.

(1)	$\{w(s_w)\}$	(2)	$\{f(s_f)\}$
(3)	$\{b(s_b)\}$	(4)	$\{c(s_c)\}$
(5)	$\{s(s_s)\}$	(6)	$\{g(s_g)\}$
(7)	$\{\neg w(X_1), a(X_1)\}$	(8)	$\{\neg f(X_1), a(X_1)\}$
(9)	$\{\neg b(X_1), a(X_1)\}$	(10)	$\{\neg c(X_1), a(X_1)\}$
(11)	$\{\neg s(X_1), a(X_1)\}$	(12)	$\{\neg g(X_1), p(X_1)\}$

$$(13)\{\neg a(X_1), \neg p(X_2), \neg a(X_3), \neg p(X_4), \\ e(X_1, X_2), \neg m(X_3, X_1), \neg e(X_3, X_4), e(X_1, X_3)\}$$

(14) $\{\neg c(X_1), \neg b(X_2), m(X_1, X_2)\}$
(15) $\{\neg s(X_1), \neg b(X_2), m(X_1, X_2)\}$
(16) $\{\neg b(X_1), \neg f(X_2), m(X_1, X_2)\}$
(17) $\{\neg f(X_1), \neg w(X_2), m(X_1, X_2)\}$
(18) $\{\neg f(X_1), \neg w(X_2), \neg e(X_1, X_2)\}$
(19) $\{\neg w(X_1), \neg g(X_2), \neg e(X_1, X_2)\}$
(20) $\{\neg b(X_1), \neg c(X_2), e(X_1, X_2)\}$
(21) $\{\neg b(X_1), \neg s(X_2), \neg e(X_1, X_2)\}$
(22) $\{\neg c(X_1), p(h(X_1))\}$
(23) $\{\neg c(X_1), e(X_1, h(X_1))\}$
(24) $\{\neg s(X_1), p(i(X_1))\}$
(25) $\{\neg s(X_1), e(X_1, i(X_1))\}$
(26) $\{\neg a(X_1), \neg a(X_2), g(j(X_1, X_2))\}$
(27) $\{\neg a(X_1), \neg a(X_2), \neg e(X_1, X_2), \neg e(X_2, j(X_1, X_2))\}$

■

Automatické dokazovače z konca sedemdesiatych rokov neboli schopné riešiť túto úlohu (hoci jej riešenie existuje). Vysvetlením je:

- veľký priestor na prehľadávanie (už v prvej generácii – vychádzajúc iba z kláuz (1) - (27) – možno získať 94 rezolvent),[15]

- veľká hĺbka, potrebná pre odvodenie sporu (ručný výpočet Schuberta, autora problému, mal hĺbku 20), s postupom do hĺbky prudko rastie počet vygenerovaných rezolvent.

Príklad 4.78 Waltherovo riešenie spočíva v zavedení viactypového (viacsortového) jazyka a hierarchie na typoch. Typy budeme označovať veľkými písmenami, premenné určitého typu veľkými písmenami s indexom:

(1)	*type* $s_w : W$	(2)	*type* $s_f : F$
(3)	*type* $s_b : B$	(4)	*type* $s_c : C$
(5)	*type* $s_s : S$	(6)	*type* $s_g : G$
(7)	*sort* $W < A$	(8)	*sort* $F < A$
(9)	*sort* $B < A$	(10)	*sort* $C < A$
(11)	*sort* $S < A$	(12)	*sort* $G < P$

$$(13)\{e(A_1, P_1), \neg m(A_2, A_1), \neg e(A_2, P_2), e(A_1, A_2)\}$$

[15] Základná infomácia o rezolvencii je tiež v dodatku D.

(14)	$\{m(C_1, B_1)\}$	(15)	$\{m(S_1, B_1)\}$
(16)	$\{m(B_1, F_1)\}$	(17)	$\{m(F_1, W_1)\}$
(18)	$\{\neg e(W_1, F_1)\}$	(19)	$\{\neg e(W_1, G_1)\}$
(20)	$\{e(B_1, C_1)\}$	(21)	$\{\neg e(B_1, S_1)\}$
(22)	*type* $h(C) : P$	(23)	$\{e(C_1, h(C_1))\}$
(24)	*type* $i(S) : P$	(25)	$\{e(S_1, i(S_1))\}$
(26)	*type* $j(A, A) : G$	(27)	$\{\neg e(A_1, A_2), \neg e(A_2, j(A_1, A_2))\}$

Azda ani netreba tento komentár: (1) – (6) sú deklarácie typov, vyjadrujú aj predpoklad neprázdnosti typu (množiny objektov). (7) – (12) definujú čiastočné usporiadanie na typoch. V (13) – (21), (23), (25), (27) sú vynechané – v porovnaní s príkladom 4.77 – literály definujúce typy a informácia o typoch sa vyjadruje pomocou typových premenných. Napokon, (22), (24), (26) sú deklaráciami typu funkcií h, i, j. ■

Už samotné preformulovanie problému do viacsortového jazyka ukazuje, že nové vyjadrenie (reprezentácia) je hutnejšie a priezračnejšie. Teraz si popíšeme inferenčnú procedúru, ktorá využíva typovú informáciu.

Jedinou zmenou – oproti štandardnej rezolvencii – je modifikovaná definícia unifikácie. Usporiadanie na typoch je základom pri sprísnení definície unifikácie: Ak je f typu $f(T_1, \ldots, T_k) : T$ a $u_1, \ldots, u_k$ sú typu $T_1, \ldots, T_k$, potom term $f(u_1, \ldots, u_k)$ je typu T. Ak je term 0-árny, typ termu je určený typom príslušnej konštany alebo premennej. Ak máme premennú x typu X, term t typu T, potom x možno unifikovať s t iba vtedy, keď $T \leq X$. (To znamená, že premennú možno unifikovať s nejakým termom iba vtedy, keď typ termu je porovnateľný s typom premennej a nie je všeobecnejší.) Na definícii rezolvencie sa už nič nemení. Vidno, že počet unifikovateľných literálov je menší, preto pri rezolvencii vznikne menší počet rezolvent a priestor na prehľadávanie je menší.

Príklad 4.79 Predpokladajme deklarácie a čiastočné usporiadanie z príkladu 4.78. Nech A_1, A_2 sú premenné typu A, B_1 typu B, C_1 typu C, P_1 typu P. Príklady prípustných unifikácií: A_1/B_1 (substitúcia B_1 za A_1 – keď niečo platí o zvieratách, môže to platiť po substitúcii aj o vtákoch) alebo $P_1/j(A_1, A_2)$. Neprípustné: B_1/A_1 alebo B_1/C_1 (o húseniciach nemusí platiť to, čo o vtákoch). ■

Riešenie nášho problému s využitím informácie o typoch (s hierarchickou štruktúrou):

Príklad 4.80 Za každým krokom rezolvovania je komentár odvolávajúci sa buď na klauzy z príkladu 4.78 alebo predchádzajúce klauzy nižšie uvedeného odvodenia. Poradové čísla kláuz z príkladu 4.78 sú od (1) po (27). Odvodenie začína poradovým číslom (28).
(28) $\{e(A_1, P_1), \neg m(A_2, A_1), \neg e(A_2, j(A_1, A_2))\}$; s využitím (13), (27) a substitúcie $P_2/j(A_1, A_2)$

počiatočný počet potenciálnych rezolvent	12	94
hĺbka prehľadávania	8	20
počet vygenerovaných kláuz	22	60
počet vygenerovaných literálov	32	221
počet dedukovaných kláuz v dôkaze	9	33

Tabuľka 4.1: Porovnanie riešenia vo viacsortovom a jednosortovom jazyku

(29) $\{e(W_1, P_1), \neg e(F_1, j(W_1, F_1))\}$; (17), (28), $A_1/W_1, A_2/F_1$
(30) $\{\neg e(F_1, j(W_1, F_1))\}$; (19), (29), P_1/G_1
(31) $\{e(F_1, P_1), \neg e(B_1, j(F_1, B_1))\}$; (16), (28), $A_2/B_1, A_1/F_1$
(32) $\{\neg e(B_1, j(F_1, B_1))\}$; (30), (31), $P_1/j(W_1, F_1)$
(33) $\{e(B_1, P_1), \neg m(S_1, B_1), \neg e(S_1, P_2)\}$; (13), (21), $A_1/B_1, A_2/S_1$
(34) $\{\neg m(S_1, B_1), \neg e(S_1, P_2)\}$; (32), (33), $P_1/j(F_1, B_1)$
(35) $\{\neg e(S_1, P_2)\}$; (15), (34)
(36) $\{\}$; (25), (35), $P_2/i(S_1)$ ■

Čitateľ už postrehol, že príčinou úspešného riešenia pri viacsortovej formalizácii a rezolvencii je podstatne redukovaný prehľadávaný priestor. Hierarchické usporiadanie na typoch, ktoré je priamo zabudované do reprezentácie, vedie k výraznému zredukovaniu potrebných výpočtov. Tabuľka 4.1 porovnáva niektoré dôležité hodnoty výpočtu vo viacsortovom prostredí s ručným výpočtom pôvodnej formulácie.

4.2.2 Striktné hierarchické siete

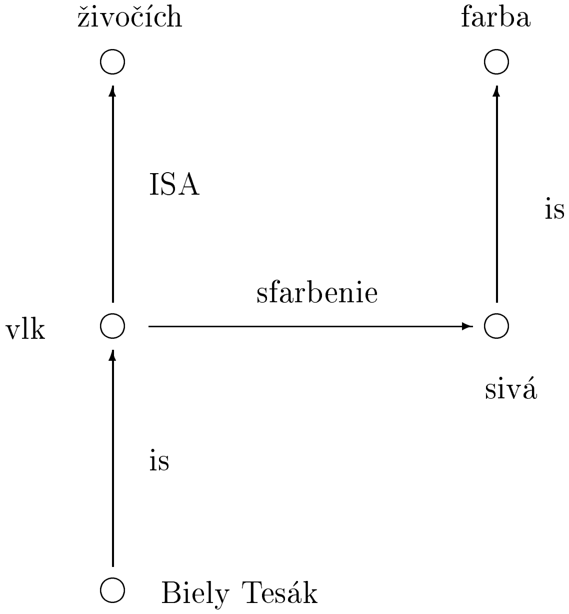
Videli sme, že reprezentáciu problému možno zjednodušiť a výpočet zefektívniť, ak zavedieme prirodzené hierarchické vzťahy. Teraz prejdeme k jadru problematiky odvodzovania v hierarchických sieťach. Začneme jednoduchým príkladom.

Príklad 4.81 (Príklad hierarchickej siete) Z obrázku 4.1 možno prečítať, že Biely Tesák je vlk, že všetci vlci sú živočíchy, že sfarbenie vlkov je sivé, že sivá patrí do triedy farieb.

Vo všeobecnosti, v každej hierarchickej sieti môžeme mať zaznamenané, že nejaký objekt, indivíduum patrí do nejakej triedy. Vyjadríme to hranou *is*. Hranou *ISA* vyjadríme, že nejaká trieda je podtriedou inej triedy.

Hierarchické siete umožňujú dediť informáciu: to, čo platí pre všeobecnejší prípad, by malo platiť aj v špeciálnejšom prípade. V našom príklade máme implicitnú informáciu o tom, že Biely Tesák je živočích, že Biely Tesák je sivý, že Biely Tesák má nejakú farbu. Príklad nám aj naznačuje, že s dedením budú isté problémy.

Hierarchické siete si budeme predstavovať (i definovať) ako nejaké grafy. Aj ďalej vrcholy budú reprezentovať indivíduá alebo triedy. Rôzne typy hrán



Obrázok 4.1: Jednoduchá hierarchická sieť

budú predstavovať členstvo v triede, vzťah podtriedy alebo nejaký iný vzťah (ako napríklad sfarbenie). Označenia (návestia) na vrcholoch a hranách predstavujú mená zodpovedajúcich indivíduí, tried a vzťahov. ■

Najpr zavedieme formálnu definíciu striktnej hierarchickej siete. V tomto prípade našou ambíciou je, aby sa všeobecnejšia informácia dedila automaticky a bez výnimiek. Ukážeme si, že informáciu z hierarchickej siete možno ekvivalentne vyjadriť prostriedkami zodpovedajúcimi definícii 1.4 z kapitoly 1. To znamená, reprezentácia grafom je rovnocenná reprezentácii vetami nejakého jazyka a využívajúcej nejaký operátor odvodenia a indexy. Toto zistenie je dôležité – jeden z mýtov umelej inteligencie hovorí o tom, že v jazyku logiky nemožno reprezentovať to, čo v hierarchických sieťach. Čitateľ, ktorého nezaujímajú detaily o ekvivalencii týchto dvoch formalizmov, môže skočiť priamo k časti 4.2.3.

Prejdime teda k definícii striktnej hierarchickej siete. Zavedieme v nej predovšetkým niektoré význačné hrany (is a ISA) a nejakú dohodu o označovaní hrán a vrcholov. Hrana is zodpovedá členstvu v triede. Hrana ISA zodpovedá vzťahu podtriedy. Hrany typu is a ISA môžu byť buď pozitívne alebo negatívne, označené ako $+is, -is, +ISA, -ISA$. Pozitívna hrana vyjadruje členstvo v triede resp. vzťah podtriedy, negatívna ich popiera.

Definícia 4.82 (Striktná hierarchická sieť) *Striktná hierarchická sieť* je trojica (V, H, N), kde V je nejaká množina vrcholov, $H \subseteq V \times V$ je množina orientovaných hrán, spájajúcich dvojice vrcholov a N je množina návestí. Navyše, splnené sú tieto podmienky:

1. existujú prinajmenšom dve vzájomne disjunktné a neprázdne podmnožiny V, označme ich ako I a J,[16]

2. existujú prinajmenšom dve vzájomne disjunktné a neprázdne podmnožiny množiny H, označme ich ako is a ISA; platí:

   - ak $(i, j) \in is$, potom $i \in I$ a $j \in J$ (hovoríme, že (i, j) je typu is),
   - ak $(u, v) \in ISA$, potom $u, v \in J$ (hovoríme, že (u, v) je typu ISA),

3. každý vrchol a každá hrana je označená jediným návestím z N, presnejšie, existuje injekcia $\phi : V \longrightarrow N$ a zobrazenie $\psi : H \longrightarrow N$ (platí pritom, že $\phi(V)$ a $\psi(H)$ sú disjunktné množiny návestí); pritom pre každú hranu z množiny is platí, že $\psi(h) = +is$ alebo $\psi(h) = -is$ a pre každú hranu h z množiny ISA platí, že $\psi(h) = +ISA$ alebo $\psi(h) = -ISA$,

4. pre žiadny vrchol u neexistuje cesta z u do u, obsahujúca aspoň jednu hranu typu ISA.

■

Striktná hierarchická sieť je teda ohodnotený orientovaný graf, ktorý spĺňa nejaké dodatočné podmienky. Teraz si zavedieme jazyk, v ktorom možno reprezentovať formulami všetko to, čo v striktnej hierarchickej sieti.

Definícia 4.83 (*HL*) Majme prvorádový jazyk, obsahujúci iba unárne a binárne predikátové symboly. Vety tohto jazyka môžu byť

- literály,
- implikácie tvaru $\forall X(L \leftarrow A)$, kde A je atóm, L literál a predikát v A môže byť iba unárny.

Jazyk takéhoto typu označíme ako *HL*. ■

Označenie *HL* pochádza od *hierarchical language.* Charakterizovali sme celú triedu jazykov. Konkrétne jazyky tejto triedy sú dané výberom predikátových symbolov a konštánt. Možno treba poznamenať, že ostávame pri opačných implikáciách, na ktoré sme si zvykli už v kapitole 2.

Explicitne reprezentované vety (množiny viet E) budú využívať iba fragment jazyka typu *HL*:

[16] Intuitívne, do I budú patriť indivíduá, do J množiny indivíduí, ktoré majú určenú vlastnosť).

- ak literál l patrí do E, potom je základný,
- ak implikácia c patrí do E, potom nie je základná,
- ak implikácia $\forall X(L \leftarrow A)$ s binárnym predikátom v L patrí do E, potom L je tvaru $q(X,b)$, kde b je konštanta.

Dôvod je zrejmý: základné literály zodpovedajú faktom, ne-základné implikácie všeobecným vzťahom medzi triedami.

V centre našej pozornosti sú od začiatku poznatky implicitné v nejakej reprezentácii poznatkov. Preto potrebujeme urobiť krok smerom k identifikácii implicitných poznatkov. Urobíme ho pre sieťovú reprezentáciu i pre reprezentáciu v *HL*.

Pokiaľ ide o sieťovú reprezentáciu, implicitná informácia sa obvykle definuje procedurálne: postupom umožňujúcim odvodiť zo striktnej hierarchickej siete obsah, ktorý je v nej implicitný. Čitateľovi je iste zrejmé, ako vyzerá procedúra, ktorá zisťuje, čo všetko dedí daný objekt alebo trieda z vyšších vrstiev hierarchie.

Definícia 4.84 (Dedenie vlastností.) Nech je daná striktná hierarchická sieť $\mathcal{H}$. Predpokladajme ďalej $u \in I, \phi(u) = a, v \in J, \phi(v) = q$. Budeme hovoriť, že objekt a má (resp. nemá) vlastnosť q

- buď vtedy, keď existuje hrana $h = (u,v)$ taká, že $\psi(h) = +is$ (resp. $\psi(h) = -is$),
- alebo v $\mathcal{H}$ existuje cesta $h_1,\dots,h_k$, kde $h_1 = (u,x_1), \psi(h_1) = +is$, ďalej pre každú z hrán h_i, $i = 2,\dots,k-1$ je $h_i = (x_{i-1},x_i)$ a $\psi(h_i) = +ISA$, napokon $h_k = (x_{k-1},v)$, pričom $\psi(h_k) = +ISA$ (resp. $\psi(h_k) = -ISA$).

■

Poznámka 4.85 Podobným spôsobom možno definovať, kedy je v sieti nejaký objekt a v binárnom vzťahu r k akémusi objektu b a kedy platí, že nejaká trieda je podtriedou inej triedy. Je zbytočné zaťažovať čitateľa detailami. ■

Na to, aby sme definovali reprezentáciu znalostí nad *HL*, potrebujeme špecifikovať operátor odovodenia $Cn_{hier}(E)$ a zadefinujeme aj množinu indexov. Pre danú množinu viet E budeme definovať množinu jej dôsledkov, získaných dedením. Môžeme hovoriť, že dedením (alebo na základe dedičnosti) sme odvodili všetky vety, ktoré patria do $Cn_{hier}(E)$, ale nepatria do E.

Definícia 4.86 (Cn_{hier}) Nech E je množina viet jazyka J, ktorý patrí do triedy jazykov *HL*. $Cn_{hier}(E)$ je najmenšia množina viet jazyka J, ktorá spĺňa nasledujúce podmienky:

- $E \subseteq Cn_{hier}(E)$,

- ak $(\forall X)\ \phi \in Cn_{hier}(E)$, potom $\phi' \in Cn_{hier}(E)$, kde ϕ' je inštanciou ϕ,
- ak $\phi \rightarrow \psi \in Cn_{hier}(E)$ a $\phi \in Cn_{hier}(E)$, tak $\psi \in Cn_{hier}(E)$,
- ak $(\forall X)\ \phi \in Cn_{hier}(E)$, tak $(\forall Y)\ \phi' \in Cn_{hier}(E)$, kde Y je ľubovoľná premenná a ϕ' sa líši od ϕ iba výskytom premennej Y miesto X (na všetkých miestach jej výskytu vo ϕ),
- ak $\forall X\ (\phi \rightarrow \psi) \in Cn_{hier}(E)$ a $\forall X\ (\psi \rightarrow \nu) \in Cn_{hier}(E)$, tak $\forall X\ (\phi \rightarrow \nu) \in Cn_{hier}(E)$.

■

Ľahko možno ukázať, že k E, množine viet jazyka *HL*, možno priradiť nejakú striktnú hierarchickú sieť. O E budeme predpokladať, že je acyklická v tom zmysle, že z nej nemožno odvodiť $A \rightarrow A$ pre žiadne A, reprezentujúce nejakú triedu.

Tvrdenie 4.87 *Predpokladajme, že v HL je explicitne sformulovaná nejaká konečná množina viet, databáza E, pričom $A \rightarrow A \notin Cn_{hier}(E)$ pre žiadne A, kde A je atóm s unárnym predikátom p.*

Potom E opisuje nejakú striktnú hierarchickú sieť $\mathcal{H}$ v nasledujúcom zmysle slova:

- *ak $p(a) \in E$ (resp. $\neg p(a) \in E$), potom existujú v $\mathcal{H}$ vrcholy $u \in I$, $v \in J$ také, že $\phi(u) = a$ a $\phi(v) = p$ a v $\mathcal{H}$ existuje hrana $h = (u,v) \in is$, pričom $\psi(h) = +is$ (resp. $\psi(h) = -is$),*
- *ak $\forall X(L \leftarrow A) \in E$, pričom A je $p(X)$ a L je $q(X)$ alebo $\neg q(X)$, potom existujú vrcholy $u, v \in J$ také, že $\phi(u) = p$ a $\phi(v) = q$ a existuje hrana $h = (u,v)$ taká, že $\psi(h) = +ISA$ (v prípade, že L je $q(X)$) alebo $\psi(h) = -ISA$ (v prípade, že L je $\neg q(X)$),*
- *ak $\forall X(q(X,b) \leftarrow p(X)) \in E$, potom v $\mathcal{H}$ existujú vrcholy u, v také, že $\phi(u) = p$, $\phi(v) = b$ a existuje hrana $h = (u,v)$, kde $\psi(h) = q$.*

Dôkaz je triviálny.

Podobne platí:

Tvrdenie 4.88 *Ak je daná striktná hierarchická sieť $\mathcal{H}$, potom možno skonštruovať jazyk typu HL tak, že $\mathcal{H}$ opisuje E, nejakú množinu viet tohto jazyka.*

Tvrdenie 4.89 *Nech E je množina viet nejakého jazyka typu HL a $\mathcal{H}$ je striktná hierarchická sieť. Nech E opisuje $\mathcal{H}$.*

Potom objekt a má vlastnosť q práve vtedy, keď $q(a) \in Cn_{hier}(E)$.

Dôkaz je jednoduchý; indukciou cez dĺžku odovodenia.

Napokon, zadefinujeme dvojprvkovú množinu funkcií $Ind = \{f, g\}$. Úlohou indexov je, aby sme na množine formúl E mohli konštruovať priamu dostupnosť medzi formulami, ktorá by zachovávala dostupnosť po hranách z hierarchických sietí. Chceme tým ukázať, že dostupnosť po hranách (ktorá sa tradične prezentuje ako výhoda hierarchických sietí v porovnaní s množinami formúl) možno reprezentovať aj v prostredí formúl. Funkcia g pre danú konštantu a „ukazuje" na všetky predikátové symboly, ktorých argumentom (priamo v E) je a. Funkcia f odkazuje na implikácie, ktorých antecedentom[17] je daný atóm. Tieto implikácie môžeme použiť pri odvodzovaní dôsledkov daného atómu.

Definícia 4.90 (Indexy) Nech E je množina viet v jazyku J. Potom g je funkcia definovaná na konštantách jazyka J, jej hodnoty sú množiny predikátových symbolov z J. Funkcia f je definovaná na literáloch z J, jej hodnotami sú množiny implikácií z J. Pritom platí:

- pre ľubovoľnú konštantu a je $g(a) = P = \{p \mid (p(a) \in E) \vee (\neg p(a) \in E)\}$,
- pre ľubovoľný literál A je $f(A) = \{\phi \mid \exists \sigma\ (\sigma(ant(\phi)) = A)\}$, kde ϕ je implikácia z E, $ant(\phi)$ je jej antecedent a σ je nejaká substitúcia.

■

Príklad 4.91 Ak $E = \{q \leftarrow p; r \leftarrow p; s \leftarrow r\}$, potom $f(p) = \{q \leftarrow p; r \leftarrow p\}$, $f(r) = \{s \leftarrow r\}$ a $f(q) = f(s) = \emptyset$. ■

4.2.3 Hierarchické siete s výnimkami a s viacnásobnou dedičnosťou

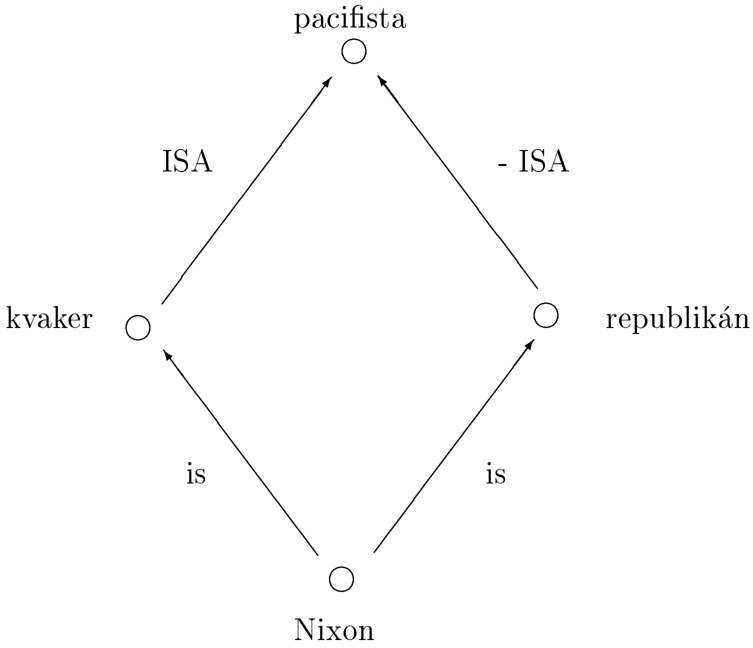
Zaviedli sme pojem striktnej hierarchickej siete. Ukázali sme, že existuje logický jazyk, v ktorom možno vyjadriť presne to, čo v striktnej hierarchickej sieti. Ukázali sme si, akú implicitnú informáciu možno odvodiť pri tomto prístupe. Naznačili sme, ako pri odvodzovaní môže pomáhať vhodné indexovanie.

Odvodzovanie v striktnej hierarchickej sieti nespôsobuje nijaké problémy, ak neobsahuje ani implicitne nijaký spor.

V tomto texte sa budeme opakovane stretávať s rôznymi klasickými príkladmi, ktoré testujú vyjadrovacie schopnosti reprezentačných formalizmov. Jeden z nich nasleduje (ilustruje, čo môže spôsobiť narušenie podmienok, že každý vrchol má nanajvýš jedného priameho predchodcu a že sieť je konzistentná).

Príklad 4.92 Na obrázku 4.2 máme jednoduchú sieť, ktorá obsahuje informáciu o tom, že Nixon je republikán a aj kvaker. (Vidíme teda, že Nixon

[17] Antecedentom implikácie $L \leftarrow A$ (alebo $A \rightarrow L$) je A.



Obrázok 4.2: Problematické dedenie v hierarchickej sieti

môže dediť z viacerých zdrojov – aj vlastnosti republikánov, aj vlastnosti kvakerov.) Ďalej, republikáni nie sú pacifisti, ale kvakeri sú pacifisti. V danej sieti je teda implicitná nekonzistentnosť.

Ak by sme skonštruovali množinu viet E, ktoré opisujú danú sieť, dostali by sme, že do $Cn_{hier}(E)$ patrí aj $\neg pacifista(nixon)$, aj $pacifista(nixon)$.[18]

Tento príklad okrem toho, že ilustruje, čo je dedičnosť, aj ukazuje, že dediť možno z viacerých zdrojov a aj konfliktné vlastnosti. ■

Po motivácii, ukazujúcej problémy späté s dedením v hierarchických sieťach, prejdeme k pokusu o ich priamočiare riešenie a ukážeme jeho intuitívnu neadekvátnosť (narušenie základných predstáv, spätých s dedením). Potom sa tejto neadekvátnosti pokúsime vyhnúť tým, že hierarchické siete budeme sémanticky špecifikovať v defaultových teóriách.

Komplikáciám, ilustrovaným uvedeným príkladom, sa medzi prvými vážne venoval Fahlman [Fah 79]. Jeho riešenie je výpočtovo rýchle, ideovo priamočiare, žiaľ, vedie k neadekvátnemu správaniu odvodzovacej mašinérie. Vôbec si totiž nepoložil problém sémantickej charakterizácie odvodzovania v hierarchickej sieti s výnimkami a viacnásobným dedením.

[18]Treba však upozorniť, že do $Cn_{hier}(E)$ nepatria všetky vety jazyka HL, napríklad $\neg republikán(nixon) \notin Cn_{hier}(E)$. Teda Cn_{hier} nie je triviálny operátor odvodenia.

Fahlmanov prístup bol založený na predstave masívneho paralelného výpočtového zariadenia. Táto predstava spočívala v tom, že každý vrchol hierarchickej siete Fahlman „realizoval“ procesorom a hrany medzi vrcholmi hierarchickej siete realizoval priamymi spojeniami medzi príslušnými procesormi. Základným problémom je zistiť, či nejaký vrchol je špeciálnym prípadom iného, všeobecnejšieho, vrcholu. Príklad 4.92 ukázal, že nie sú vylúčené konflikty. Niektoré cesty v sieti môžu potvrdzovať, že ide o špeciálny prípad, iné to môžu vyvracať. V transformácii do paralelnej výpočtovej siete: Máme štartovací procesor a cieľový procesor. Algoritmus šíri značky po všetkých možných linkách od procesora k procesoru. V každom kroku posun o jednu linku. Riešením problému je cesta zo štartu do cieľa, ktorá sa nájde ako prvá. Preto sa zodpovedajúci algoritmus nazýva algoritmom najkratšej cesty.

Fahlmanov prístup si preformulujeme v reprezentačnom formalizme typu HL. Explicitne využijeme index f. Budeme predpoladať acyklické a konečné E. Budeme potrebovať mocniny indexu f. Skôr než ich zadefinujeme, príklad:

Príklad 4.93 (Pokračovanie príkladu 4.91) $E = \{q \leftarrow p; r \leftarrow p; s \leftarrow r\}$.

Z p sa môžeme „dostať“ do r a odtiaľ do s. Túto prostú myšlienku budeme vyjadrovať pomocou mocnín indexu f. Pripomeňme, že $f(p) = \{q \leftarrow p; r \leftarrow p\}$, $f(r) = \{s \leftarrow r\}$ a $f(q) = f(s) = \emptyset$.

Prvá mocnina f bude funkcia, ktorá pre daný argument p dá konzekventy všetkých implikácií, čo patria do $f(p)$. T.j. $f^1(p) = \{q, r\}$. Druhá mocnina bude obsahovať konzekventy všetkých implikácií, čo patria do $f(q) \cup f(r)$, teda $f^2(p) = \{s\}$. ■

Ak ϕ je nejaká implikácia tvaru $A \to B$, potom jej konzekvent B si označíme ako $con(\phi)$. Ak Φ je nejaká množina implikácií, potom $con(\Phi)$ označuje množinu všetkých konzekventov z Φ.

Definícia 4.94 Nech f je index, A literál, $f(A) \neq \emptyset$.

- $f^1(A) = con(f(A))$,
- $f^{i+1}(A)$
  - $= \bigcup_{B \in con(f(f^i(A)))} f(B)$, pričom pre aspoň jedno B je $f(B) \neq \emptyset$,
  - je nedefinované, ak je $f(B) = \emptyset$ pre každé B.

■

Ak $f(A) \neq \emptyset$, tak existuje maximálne k, pre ktoré je definované $f^k(A)$, pretože E je konečná a acyklická. V príklade 4.92 sme videli, že do $Cn_{hier}(E)$ môže pre nejakú množinu viet E patriť veta i jej negácia.

Fahlmanov algoritmus (tabuľka 4.2)[19] má podobné nekonzistentnosti riešiť na základe najkratšej cesty, potrebnej k odvodeniu niektorého z dôsledkov.

Tento algoritmus charakterizujeme zadefinovaním operátora Cn_{fahl}. Mô-

[19] Algoritmus má rozhodnúť, či platí $p(a)$. Analogicky ho možno formulovať pre $\neg p(a)$.

Algoritmus 4.95 (Algoritmus najkratšej cesty)
vstup: *E, acyklická a konečná množina viet jazyka HL, množina $Ind = \{f, g\}$, predikát p a konštanta a*
výstup: *rozhodnutie, či platí $p(a)$*

1. *$P := g(a)$,*
2. *ak $p \in P$ a $\neg p(a) \notin E$, úspešný koniec,*
3. *ak $p \in P$ a ($(p(a) \notin E)$ alebo $(p(a) \in E$ a $\neg p(a) \in E)$), neúspešný koniec,*
4. *$k := 1$,*
5. *dovtedy, kým nebude splnená jedna z nasledujúcich podmienok:*

   neúspešný koniec *ak $f^k(q(a)) = \emptyset$ pre každé $q \in P$ alebo platí $\neg p(a) = con(\psi)$, pričom $\psi \in f^k(q(a))$ pre nejaké $q \in P$ (* platnosť $p(a)$ sa nezistí skôr ako platnosť $\neg p(a)$ *)*

   úspešný koniec *ak $p(a) = con(\phi)$ pre nejaké $\phi \in f^k(q(a))$ a pre nejaké $q \in P$;*

   rob $k := k + 1$.

Tabuľka 4.2: Algoritmus najkratšej cesty

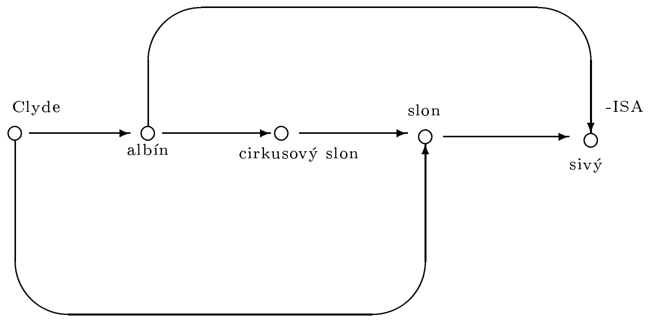
žeme zadefinovať: $p(a) \in Cn_{fahl}(E)$ práve vtedy, keď existuje $q \in P$ také, že $p(a) \in f^k(q(a))$ pre k, pri ktorom algoritmus úspešne zastavil.

Výpočet najkratšej cesty je rozhodne efektívny z výpočtového hľadiska, vedie však k protiintuitívnym výsledkom. Dokonca možno tvrdiť, že sémantika tohto výpočtu sa nedá charakterizovať. Uvedieme si príklady, ktoré ilustrujú základné problémy:

1. Redundantné hrany od vrcholov nízkej úrovne v hierarchii k vrcholom vysokej úrovne môžu nepriaznivo ovplyvniť odvodzovanie podľa najkratšej cesty.

2. Existujú principiálne viacznačné siete, v ktorých sú dva pevné vrcholy spojené konfliktnými cestami, ale prípadná menšia dĺžka jednej z nich nie je vážnym dôvodom pre jej preferovanie.

3. Na najkratšej ceste môže ležať vrchol, ktorý „nemá silu argumentu“ – môže byť neutralizovaný alebo vyvrátený inou cestou.



Obrázok 4.3: Redundantné hrany spôsobujú neadekvátnosť stratégie najkratšej cesty

Príklad 4.96 Obrázok 4.3 ukazuje, že náhodné hrany z nižších úrovní do oveľa vyšších úrovní siete nemožno brať s rovnakou vážnosťou ako hrany, ktoré postupne konštruujú nejakú hierarchiu.

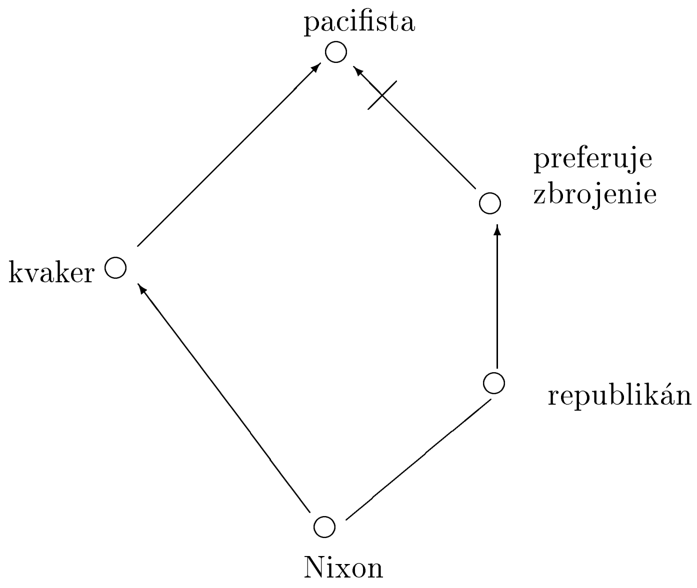
V našom prípade hrana od vrcholu *Clyde* k vrcholu *slon* pôsobí v hierarchii rušivo, pri odvodzovaní ju nemožno brať tak vážne ako ostatné hrany. Napriek tomu, pri algoritme najkratšej cesty neexistuje spôsob, ako sa s tým vyrovnať. ■

Príklad 4.97 Na obrázku 4.2 sme videli príklad dvoch alternatívnych ciest, ktoré sa vzájomne neutralizovali. Pridanie vrcholu na ľubovoľnú z ciest nezvyšuje našu dôveru v alternatívnu cestu, pozri napríklad obrázok 4.4. Dĺžka cesty nie je dôvodom, na ktorom by sa mohlo zakladať preferovanie. ■

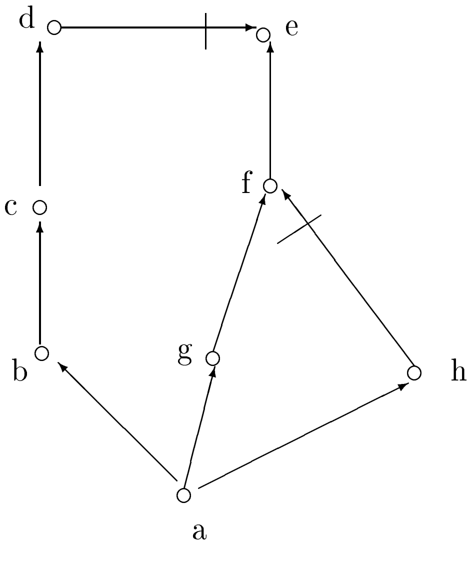
Príklad 4.98 Ďalší problematický prípad vidno na obrázku 4.5. Kratšia cesta (z a cez g a f do e je neutralizovaná: negatívna cesta z a cez h do f nedovoľuje rozhodnúť, či platí f, či f neplatí. Preto je dôveryhodnejšia dlhšia (negatívna) cesta z a cez b, c, d do e. ■

Stojíme teda pred úlohou nájsť nejaký spoľahlivý sémantický základ, umožňujúci zvládnuť problémy, na ktoré nestačí stratégia najkratšej cesty. Prítomnosť neadekvátností vo výpočtových modeloch usudzovania o hierarchických vzťahoch naznačuje, že je veľmi dôležité dôkladne špecifikovať, čo daná hierarchická sieť znamená (akú informáciu ukrýva).

Pokúsime sa detailnejšie pozrieť na to, čo prinesie revízia naivného prístupu k dedeniu v hierarchických sieťach. Najprv si však trochu „odľahčíme" nekonzistentnosti, ktoré sa objavujú v hierarchických sieťach – pripustíme, že nie sú neočakávané a patologické. Toto nadľahčenie je založené na predstave, že hierarchická sieť obsahuje fragmenty, reprezentujúce obvyklý, typický stav vecí. Pritom takýto obvyklý stav nemusí vždy platiť. Akceptovateľné sú



Obrázok 4.4: Kratšiu cestu možno ignorovať



Obrázok 4.5: Neutralizácia kratšej cesty

alternatívne predstavy o obvyklom stave. V defaultových teóriách sme to charakterizovali existenciou viacerých alternatívnych extenzií.

Bude nás zaujímať, ako špecifikovať odvodzovanie v hierarchických sieťach pomocou defaultových teórií. Defaultové teórie budú slúžiť ako báza sémantickej charakterizácie.

Treba pripomenúť, že v tomto texte sa stretáme s už tretím poňatím sémantickej špecifikácie: po špecifikácii vzorom (príkladom bola relačná algebra) a teórii modelov sa stretáme s prekladom nejakého jazyka (grafový popis hierarchických sietí) do iného, lepšie pochopeného jazyka (defaultových teórií).[20]

V reči defaultových teórií vzniká pri aplikovaní algoritmu najkratšej cesty takýto problém: Predpokladajme, že defaultová teória zodpovedajúca danej hierarchickej sieti má viac extenzií. Potom výpočet podľa najkratšej cesty môže poskytovať literály z rôznych extenzií a neposkytovať všetky literály jednej extenzie.[21]

Príklad 4.99 Našu doterajšiu predstavu o hierarchickej sieti rozšírime. Vráťme sa k príkladu 4.1. Je zrejmé, že v spomínanom príklade nie je rozumné predpokladať, že každý vlk je sivý.

Tu, v modifikovanej verzii nášho príkladu rozlišujeme striktné a slabé hrany. Slabá je hrana od vrcholu vlk k vrcholu sivá (označená je návestím D a aj návestím *sfarbenie*). Rozumieme jej tak, že vlk je spravidla (nie však nutne) sivý. Pozri obrázok 4.6. ■

Nasledujúci príklad zavedie ďalší typ hrán, tzv. výnimkové hrany. Navyše nám ukáže nesúlad algoritmu najkratšej cesty s konceptom extenzie.

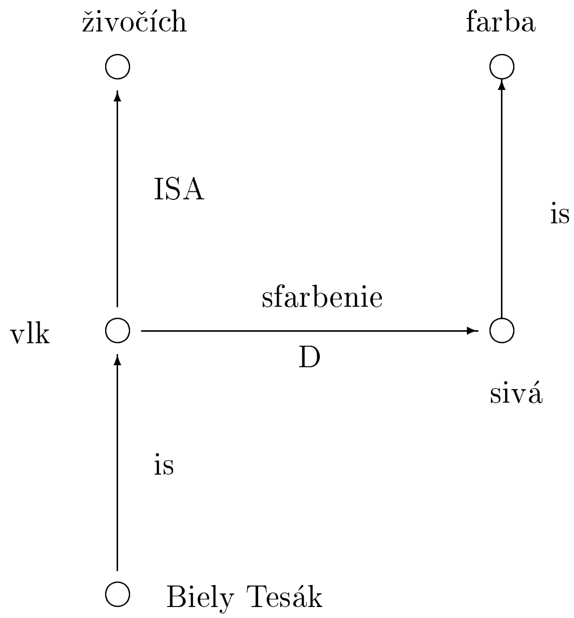
Príklad 4.100 Na obrázku 4.7 uvádzame ďalší druh hrán, výnimkové hrany. Sú tam tri hrany typu is: a je p, b je p, b je t. Ďalej sú na obrázku slabé hrany. Zámerne použime rôzne intuitívne vyjadrenia: p je spravidla q, q je typicky r, r je obvykle s, t obyčajne nebýva s.

Výnimková hrana vedie od vrcholu r k negatívne označenej hrane (t, s). Význam tejto výnimkovej hrany: t obyčajne nebýva s, výnimkou sú však tie prípady, ktoré majú vlastnosť r.

Takýto druh hrany zaviedol Etherington [Eth 88] nemá však jasný grafový zmysel (hrana od vrcholu k hrane). My si tu výnimkové hrany zavedieme

[20] Sú známe aj iné prístupy k prekladovej sémantike hierarchických sietí, využívajúce cirkumskripciu, autoepistemickú logiku alebo iné formalizmy. V tomto texte sme ako reprezentanta týchto formalizmov vybrali defaultové teórie, preto ostávame pri nich. Krátku informáciu o cirkumskripcii alebo autoepistemickej logike možno nájsť v dodatkoch J a I.

[21] Možno citovať aj vážny Etheringtonov [Eth 88] argument: „V prípade sietí s viac než jednou extenziou treba zabrániť vzájomnému interagovaniu rôznych extenzií pri odvodzovaní. Zdá sa, že tento cieľ je nemožné dosiahnuť jednoprechodovým paralelným algoritmom s čisto lokálnym riadením.“ Tento argument zosilňuje argument intuitívnej neadekvátnosti heuristiky najkratšej cesty.



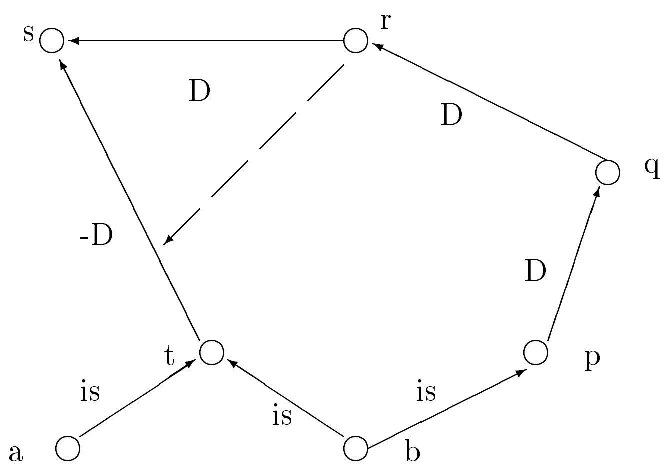
Obrázok 4.6: Revidovaný pohľad na Bieleho Tesáka

ako hyperhrany. Hyperhrana nie je dvojica vrcholov, ale nejaká množina vrcholov. V našom príklade by sme mali trojicu (t, s, r) s takýmto zamýšľaným významom: t obvykle nie sú s, s výnimkou tých, čo sú r. Musíme ešte vyriešiť komplikácie spôsobené rôznymi rolami prvkov hyperhrany a nejasnosťami, spôsobenými ich možnými pozitívnymi i negatívnymi výskytmi. Vyriešime to tak, že zavedieme akési značkovanie. Ohodnotenie hyperhrany bude vlastne funkcia, ktorá k danej hyperhrane priraďuje nejaké usporiadanie jej prvkov. Toto usporiadanie odlíši „typickú závislosť" a výnimkové stavy. Ďalej, funkcia k prvkom hyperhrany priradí nejaký príznak, určujúci, či daný vrchol budeme chápať pozitívne alebo negatívne.

Dohodneme sa, že prvé dva vrcholy uvedené v hyperhrane zodpovedajú nejakej slabej hrane, všetky nasledujúce reprezetujú prípadné výnimkové stavy. ■

Definícia 4.101 (Hypergraf) *Hyperhrana* je podmnožina množiny vrcholov. *Hypergraf* je dvojica množín (V, H), kde V je množina vrcholov a H je množina hyperhrán. ■

Definícia 4.102 *Hierarchická sieť* je ohodnotený orientovaný hypergraf, ktorý okrem podmienok kladených na striktnú hierarchickú sieť spĺňa nasledujúce podmienky:



Obrázok 4.7: Príklad výnimkových hrán

- existuje neprázdna podmnožina množiny H, označme si ju ako D,
- existuje zobrazenie $\rho : D \longrightarrow \{+, -\}$,
- existuje množina (nie nutne neprázdna) hyperhrán X a zobrazenie do návestí $\tau : X \longrightarrow N$, pričom
  - každá z hyperhrán $(u_1, \ldots, u_n) \in X$ má prinajmenšom tri prvky,
  - ak $hh \in X$ a $\tau(hh) = t$, potom t je meno funkcie, ktorá k $u_1, \ldots, u_n$ priradí dvojicu $\langle \chi(u_1, u_2), (\chi_3 u_3, \ldots, \chi_n u_n) \rangle$, kde $\chi, \chi_i \in \{0, 1\}$;
- $\rho(D) \cap \phi(V) \cap \psi(H) \cap \tau(Y)$ sú po dvojiciach disjunktné.

■

Na objasnenie: Predpokladajme, že $h = (u, v) \in D$. Ak $\rho(h) = +$, môžeme to chápať tak, že u je „v typickom prípade“ podtriedou v. Ak $\rho(h) = -$, tak u je „v obvyklom prípade“ podtriedou komplementu triedy v.

Ďalej: Majme dvojicu $< \chi_1(u_1, u_2), (\chi_3 u_3, \ldots, \chi_n u_n) >$, kde $\chi_i \in \{0, 1\}$. Zamýšľaný význam: Ak každé $\chi_i = 1$, tak prvok z triedy u_1 patrí do triedy u_2 s výnimkou tých prvkov, ktoré patria do $u_3 \cup \cdots \cup u_n$. Príznak $\chi_i = 0$ funguje ako negácia. Ak pred dvojicou (u_1, u_2) je $\chi_1 = 0$, potom u_1 nepatrí do triedy u_2. Ak χ_i má hodnotu 0 pred u_i, kde $i > 2$, potom výnimkovým stavom je to, že daný prvok nepatrí do triedy u_i.

Teraz prejdeme k rozšíreniu jazyka *HL* tak, aby sme mohli v tomto jazyku opisovať všeobecné hierarchické siete. Najprv jednoduchý príklad.

Príklad 4.103 Rozšírenú verziu príkladu o Bielom Tesákovi môžeme vyjadriť takto:

$$vlk(biely\_tesák),$$
$$\forall X\ (vlk(X) \rightarrow živočích(X)),$$
$$\forall X\ (sivá(X) \rightarrow farba(X)),$$
$$vlk(X) : sfarbenie(X, sivá)/sfarbenie(X, sivá).$$

Striktné hrany zodpovedajú klasickým implikáciám, nepripúšťajúcim výnimky. Slabé hrany zodpovedajú defaultovým pravidlám a vyjadrujú typické vzťahy. To, že sme použili otvorené defaultové pravidlo, nás neznepokojuje, naopak – budeme ich používať i ďalej a môžeme ich chápať ako reprezentantov všetkých ich inštancií.

Aby sme boli pripravení na porovnanie tohto jazyka a hierarchických sietí, treba ešte okomentovať zvláštnosti triedy hrán D. Všimnime si hranu $h = (vlk, sivá) \in D$ z obrázku 4.6, kde $\rho(h) = +$. Intuitívne, trieda vlkov je podtriedou sivo sfarbených (bytostí). Tá istá hrana, napríklad naša h, môže byť ohodnotená aj zobrazením ρ, aj zobrazením ψ. Teda aj prienik $ISA \cap D$ môže byť neprázdny. ■

Sme teda pripravení na rozšírenie jazyka *HL*.

Definícia 4.104 (Jazyk *HDL*) *Jazyk HDL* je jazyk *HL* obohatený o normálne a semi-normálne defaultové pravidlá tvaru

$$p(X) : q(X)/q(X),$$
$$p(X) : \neg q(X)/\neg q(X),$$
$$p(X) : q(X) \wedge r_1(X) \wedge \cdots \wedge r_k(X)/q(X),$$
$$p(X) : \neg q(X) \wedge r_1(X) \wedge \cdots \wedge r_k(X)/\neg q(X),$$

kde $k \geq 1$ a v predpokladoch, strážach a dôsledkoch sú literály jazyka *HL*. Predikátový symbol z literálu $q(X)$ s voľnou premennou X môže byť unárny alebo binárny. ■

Poznámka 4.105 Ľubovoľná formula jazyka *HL* je teda aj formulou jazyka *HDL*. Defaultové teórie sú dvojice (E, D). E obsahuje formuly jazyka *HL*. Vďaka rozšíreniu *HL* na *HDL* môžeme používať aj defaultové pravidlá dvoch druhov: normálne defaultové pravidlá s jediným literálom v predpoklade, jediným literálom v dôsledku (totožným so strážou) a semi-normálne pravidlá s nejakou konjunkciou literálov v stráži (tá obsahuje aj dôsledok).

Obmedzenie na jediný literál prirodzene zodpovedá vzťahom v hierarchii – ide o vzťahy podriadenosti medzi dvoma triedami. Výnimkové hrany zodpovedajú semi-normálnym defaultovým pravidlám. ■

Zhrňme: Hierarchické siete môžeme sémanticky špecifikovať v defaultových teóriách. Extenzie charakterizujú množinu presvedčení, v ktoré možno súčasne veriť na základe danej defaultovej teórie. Pri charakterizácii toho, čo sa dá odvodiť z hierarchickej siete, môžu byť užitočné extenzie zodpovedajúcej defaultovej teórie. Pomocou extenzií je vyjadrené *minimálne kritérium korektnosti* pre odvodzovanie v hierarchických sieťach:

Konvencia 4.106 (Etherington [Eth 88]) Nech defaultová teória T špecifikuje sémantiku hierarchickej siete $\mathcal{H}$. Potom odvodzovanie v hierarchickej sieti je korektné, ak odvodené dôsledky patria do tej istej extenzie T. ■

Vzťah hierarchických sietí a defaultových teórií charakterizujú nasledujúce dve tvrdenia. Konštatujú, že *každej hierarchickej sieti zodpovedá nejaká defaultová teória v jazyku HDL a naopak, množine viet v jazyku HDL zodpovedá nejaká hierarchická sieť.* Je celkom prirodzené, keď (doménovo-špecifické) defaultové pravidlá chápeme ako špeciálny druh viet.

Žiaľ, hoci zmysel tvrdenia o korešpondencii oboch formalizmov je priezračný a nekomplikovaný, opis pomerne zložitých štruktúr však vyžaduje ťažkopádnejšie formulácie. Čitateľ, ktorý je spokojný s intuíciou vyjadrenou textom v kurzíve o odstavec vyššie, môže preskočiť definíciu 4.107 a tvrdenie 4.108. Najprv pomocná definícia.

Definícia 4.107 Nech $l(X)$ a $r_i(X)$ pre $i = 1, \ldots, k$ sú literály a $p(X)$ je atóm.

Budeme hovoriť, že

$$\tau(hh) = \langle \chi(u_{k+1}, u_{k+2}), (\chi_1 u_1, \ldots, \chi_k u_k) \rangle$$

je *adekvátne definované* pre defaultové pravidlo

$$p(X) : l(X) \wedge r_1(X) \wedge \cdots \wedge r_k(X) / q(X)$$

a pre hyperhranu

$$hh = \{u_{k+1}, u_{k+2}, u_1, \ldots, u_k\},$$

ak platí: χ pred (u_{k+1}, u_{k+2}) je 1 práve vtedy, keď $l(X)$ je atóm a χ_i je 1 pred u_i, kde $i = 1, \ldots, k$, práve vtedy, keď r_i je atóm. ■

Tvrdenie 4.108 *Ľubovoľná množina viet E, vyjadrená v HDL, opisuje nejakú hierarchickú sieť $\mathcal{H}$ v nasledujúcom zmysle slova:*

- *pre všetky vety z E, ktoré patria do jazyka typu HL, sú splnené podmienky z tvrdenia 4.87,*
- *ak $p(X) : q(X)/q(X) \in E$ (alebo $p(X) : \neg q(X)/\neg q(X) \in E$), potom existujú vrcholy u, v z $\mathcal{H}$ také, že*

(1) *ak q je unárny predikátový symbol,* $\phi(u) = p$, $\phi(v) = q$, *tak v* $ISA \cap D$ *existuje hrana* $h = (u, v)$ *taká, že* $\rho(h) = +$ *(alebo* $\rho(h) = -$*),*

(2) *ak q je binárny predikátový symbol,* $\phi(u) = p$, $\psi(h) = q$, $\phi(v) = b$ *pre nejaké* $b \in I$, $h \in D$, *tak* $\rho(h) = +$ *(alebo* $\rho(h) = -$*);*

- *ak* $p(X) : l(X) \wedge r_1(X) \wedge \cdots \wedge r_k(X)/l(X) \in E$, *kde* $l(X)$ *a* $r_i(X)$ *pre* $i = 1, \ldots, k$ *sú literály a* $p(X)$ *je atóm, potom v* $\mathcal{H}$ *existujú vrcholy* $u_1, \ldots, u_k$ *také, že* $\phi(u_i) = r_i$ *(alebo* $\phi(u_i) = s_i$, *ak* $r_i = \neg s_i$*), ďalej existujú v* $\mathcal{H}$ *vrcholy* u_{k+1}, u_{k+2}, *spĺňajúce buď podmienku* (1) *alebo* (2) *a existuje hyperhrana* $hh = \{u_{k+1}, u_{k+2}, u_1, \ldots, u_k\} \in X$ *s adekvátne definovaným* $\tau(hh)$.

Naopak, každú hierarchickú sieť môžeme špecifikovať v nejakej defaultovej teórii.

Tvrdenie 4.109 *Ak je daná hierarchická sieť* $\mathcal{H}$, *potom možno skonštruovať jazyk typu HDL tak, že jediná množina viet tohto jazyka opisuje* $\mathcal{H}$.

Etheringtonovo kritérium (konvencia 4.106) nemusí pôsobiť dostatočne presvedčivo: Nie každá defaultová teória má extenziu. Našťastie, istý druh defaultových teórií, prirodzene zodpovedajúci hierarchickým sieťam, má zaručene extenziu. Videli sme, že stačí uvažovať iba semi-normálne defaultové teórie s jediným atómom ako predpokladom a jediným literálom ako konzekventom. Viac na špecifikovanie sémantiky hierarchických sietí nepotrebujeme.

Ťažkosti spôsobujú cyklické závislosti medzi výnimkovými stavmi.

Príklad 4.110 Nech d_1 je $: p \wedge \neg q/p$, d_2 je $: q \wedge \neg r/q$ a d_3 je $: r \wedge \neg p/r$. $D = \{d_1, d_2, d_3\}$. Uvažujme defaultovú teóriu $(\emptyset, D)$.

Všímajme si, ako vzájomne súvisia stráže a dôsledky rôznych defaultových pravidiel. Použitie jedného defaultového pravidla môže zabrániť použitiu ďalšieho z pravidiel. Pomocou d_1 možno odovodiť p, a tým zabrániť použitiu d_3. Keď však nemožno použiť d_3, je aplikovateľné d_2, tým sa však blokuje použitie d_1. Teda, použitie d_1 vo svojich dôsledkoch zabraňuje použitiu d_1. To isté platí pre každé z uvedených pravidiel. ■

Videli sme teda, že isté cyklické závislosti medzi strážami a dôsledkami viacerých semi-normálnych defaultových pravidiel majú nepríjemné následky.

Preto je dôležité venovať sa defaultovým teóriám, ktoré vylučujú takéto konflikty. Tieto teórie sa nazývajú usporiadanými [Eth 88]. Na literáloch jazyka defaultovej teórie zadefinujeme usporiadania $\lll$ a $\ll$. Defaultovú teóriu budeme považovať za usporiadanú, ak usporiadanie $\ll$ je nereflexívne, teda pre žiaden literál α jazyka danej defaultovej teórie neplatí $\alpha \ll \alpha$.

Ďalej si budeme všímať sieťové defaultové teórie:

Definícia 4.111 (Etherington) Defaultová teória (E, D) sa nazýva *sieťovou* teóriou práve vtedy, keď spĺňa nasledujúce podmienky:

(i) E obsahuje iba literály a disjunkcie literálov,

(ii) D obsahuje iba normálne a seminormálne pravidlá tvaru $\alpha : \beta \wedge \gamma_1 \cdots \wedge \gamma_k / \beta$, kde α je atóm, β, γ_i sú literály, $k \geq 0$.

■

Poznámka 4.112 Každá teória v jazyku *HDL* je sieťová. ■

Teraz prejdeme k definovaniu už spomínaného usporiadania. Základná myšlienka predkladanej definície je

- predpoklad defaultového pravidla nemá byť vyhodnocovaný neskôr ako jeho záver ($\alpha \lll \beta$),
- v prípade stráží, literály k nim komplementárne treba vyhodnotiť skôr ako dôsledok ($\overline{\gamma_i} \ll \beta$).

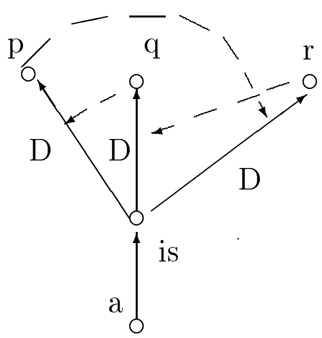
Ďalej treba myslieť na interakcie viacerých defaultových pravidiel: O usporiadaní $\lll$ je rozumné predpokladať, že je to reflexívna relácia. O oboch, že sú tranzitívne a že $\ll$ je špeciálnym prípadom (podmnožinou) $\lll$. Ak literál β_1 treba vyhodnotiť skôr ako literál β_2 a β_2 nemá byť vyhodnotené neskôr ako literál β_3, potom by β_1 mal byť vyhodnotený skôr ako β_3.

Definícia 4.113 Nech D je množina defaultových pravidiel tvaru $\alpha : \beta \wedge \gamma_1 \wedge \cdots \wedge \gamma_k / \beta$, kde $k \geq 0$, α je atóm, β, β_i a γ_i sú literály. Reláciu $\ll$ na literáloch definujeme takto:

- $\alpha \lll \beta$ pre každé defaultové pravidlo,
- $\overline{\gamma_i} \ll \beta$ pre každé i a pre každé defaultové pravidlo,
- $\gamma \lll \gamma$ pre každý literál γ,
- $\ll$ a $\lll$ sú tranzitívne,
- $\ll \subseteq \lll$,
- ak $\beta_1 \ll \beta_2$ a $\beta_2 \lll \beta_3$, potom $\beta_1 \ll \beta_3$,
- ak $\beta_1 \lll \beta_2$ a $\beta_2 \ll \beta_3$, potom $\beta_1 \ll \beta_3$.

Ak D je množina defaultových pravidiel nejakej (semi-normálnej) defaultovej teórie, v ktorej neplatí $\beta_i \ll \beta_i$ pre žiadny literál β_i, potom táto teória sa nazýva *usporiadaná*. ■

Príklad 4.114 Vráťme sa k príkladu 4.110. Na základe definície relácie $\ll$ môžeme povedať: $q \ll p, r \ll q, p \ll r$, teda $r \ll r, p \ll p, q \ll q$. ■



Obrázok 4.8: Patologicky zacyklené výnimkové hyperhrany

Tvrdenie 4.115 (Etherington [Eth 88]) *Každá usporiadaná konečná semi-normálna defaultová teória má aspoň jednu extenziu.*

Základná myšlienka dôkazu je vo využití istého rozvrstvenia, stratifikácie defaultových pravidiel na základe daného usporiadania.[22] Extenziu konštruujeme postupne, od najnižšej vrstvy po najvyššiu.

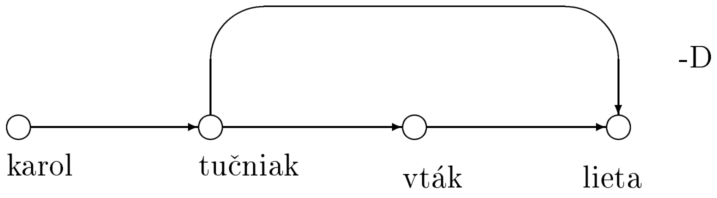
Zavedenie konceptu usporiadanej defaultovej teórie naznačuje, že naše doterajšie predstavy o hierarchických sieťach boli príliš liberálne. Potrebujeme nejako limitovať možnosti tvorenia výnimkových hyperhrán, aby neviedli k nejakým konfliktom (cyklickým závislostiam). Týmto smerom však nebudeme pokračovať. Samozrejme, medzi takto obmedzenými hierarchickými sieťami a usporiadanými defaultovými teóriami tiež existuje vzájomná korešpondencia, podobne ako v tvrdeniach 4.87 a 4.88.

Príklad 4.116 Príklad siete s patologicky zacyklenými výnimkami máme na obrázku 4.8 ■

Zdalo by sa teda, že sa nám podarilo dosiahnuť uspokojujúci stav. Máme dôveryhodnú konvenciu, na základe ktorej extenzie defaultových teórií sémanticky podopierajú odvodzovanie v hierarchických sieťach. Máme aj záruku, že usporiadané defaultové teórie, ktoré prirodzene zodpovedajú hierarchickým sieťam, zaručene majú extenziu. Samozrejme, situácia nie je až tak jednoduchá.

Príklad 4.117 Hierarchickú sieť z príkladu 4.92 vyjadríme v tvare defaultovej teórie.

[22] S ideou stratifikácie sa stretneme častejšie. Je to jedna z užitočných konštrukcií, používaných v teóriách hypotetického usudzovania. Pre defaultové teórie ideu stratifikácie detailnejšie rozpracoval Cholewinski [Cho 94, Cho 95].



Obrázok 4.9: Preferovanie špecifickejšej informácie

$E = \{republikán(nixon), kvaker(nixon)\}$,
$D = \{$

$$\frac{republikán(nixon) : \neg pacifista(nixon)}{\neg pacifista(nixon)},$$
$$\frac{kvaker(nixon) : pacifista(nixon)}{pacifista(nixon)}\}.$$

Uvedená defaultová teória má dve extenzie:

$$\begin{aligned} \mathcal{E}_1 &= Cn_{FOL}(\{republikán(nixon), kvaker(nixon), \neg pacifista(nixon)\}), \\ \mathcal{E}_2 &= Cn_{FOL}(\{republikán(nixon), kvaker(nixon), pacifista(nixon)\}). \end{aligned}$$

Teraz prichádza kľúčová otázka: čomu veriť – je nixon pacifista alebo nie je? ■

Narazili sme na problém: Defaultové teórie nedávajú nástroj selekcie medzi extenziami. Všetky možné extenzie sú pre ne „rovnako dobré“. Na druhej strane však cítime, že samotné hierarchické siete umožňujú rozlišovať medzi extenziami. Hierarchická štruktúra dovoľuje identifikovať a preferovať špecifickejšiu informáciu.

Príklad 4.118 Na obrázku 4.9 máme takúto situáciu: Existuje nejaký karol, ktorý je tučniakom. Každý tučniak je vták. Vtáci obvykle lietajú. Tučniaci obvykle nelietajú.

Keď sa od objektu *karol* pohybujeme smerom hore, môžeme sa dostať do konfliktu: môžeme odvodiť, že lieta i nelieta. V zásade je však rozumné považovať špecifickejšiu informáciu za záväznejšiu. S rastúcou mierou všeobecnosti rastie množstvo výnimiek a klesá relevantnosť informácie pre nižšie úrovne hierarchie. Predpoklad, že nejaký objekt je tučniak, je špecifickejší ako predpoklad, že je vták. Dôsledok, že karol nelieta, je založený na špecifickejšej informácii, preto ho prijmeme. ■

Bude teda užitočné, keď zaujmeme nejaký postoj, determinujúci výber medzi extenziami. Základnými možnosťami pri využití defaultových teórií na popis problému dedenia v hierarchických sieťach sú dva krajné postoje:

liberálny (dôverčivý), ktorý prijíma nejaký záver vtedy, keď patrí aspoň do jednej extenzie a radikálny (skeptický), ktorý prijme záver iba vtedy, keď patrí do každej extenzie:

Príklad 4.119 Vráťme sa k príkladu 4.117. Pri dôverčivom prístupe objekt *nixon* môže zdediť vlastnosť *pacifista* i jej opak.

Skeptik neprijme ani $pacifista(nixon)$, ani $\neg pacifista(nixon)$. Základný postoj skepticizmu je zdržať sa úsudku v prípade, keď je viacero možných konfliktných záverov. Zjednotenie viacerých extenzií normálnej defaultovej teórie je nekonzistentné. ■

Medzi krajnými prístupmi – dôverčivým a skeptickým – sú viaceré možnosti. Jednu z nich možno nazvať zdržanlivým skepticizmom. Touretzky a jeho kolegovia v [HTT 90, Tou 86] zliberalizovali radikálny skepticizmus. Podrobnejšie, tento postoj je charakteristický tým, že:

- Akceptuje – v základných rysoch – skeptický postoj: nejakú vetu považuje za platnú, ak platí v každej extenzii. Špeciálnym prípadom je, keď nejaká veta platí v jednej extenzii, v inej platí jej negácia, vtedy sa ani jedna z nich nepovažuje za platnú: „vzájomne konfliktné argumenty sa neutralizujú“ [HTT 90].
- V rozumných prípadoch možno zo skepticizmu zľaviť: dovolenými výnimkami z nekompromisného skepticizmu sú
  - ak sa konfliktné vety (povedzme, že ϕ a $\neg\phi$) explicitne vyskytujú v báze znalostí E, tak ich treba akceptovať; nie je dôvod vyberať medzi nimi, ani nie je nijaký podklad pre to, aby sme im neverili, keďže boli zaznamenané priamo do E,
  - ak je jeden člen konfliktnej dvojice podoprený špecifickejšou informáciou, potom ho preferujeme.

Príklad 4.120 Vráťme sa k príkladu 4.118. Zdržanlivý skeptik preferuje odvodenie, že *karol* nelieta. Karol je totiž tučniak, tučniaci obvykle nelietajú. Karol je aj vták (a vtáci lietajú), ale informácia o tučniakoch je špecifickejšia ako informácia o vtákoch, preto ju preferujeme. ■

Kritérium preferencie špecifickejšej informácie možno vyjadriť aj v reči defaultových teórií – opäť pomocou semi-normálnych teórií.

Príklad 4.121 Sieti z príkladu 4.118 zodpovedá defaultová teória:

$$\begin{aligned} E &= \{tučniak(a) \leftarrow; vták(X) \leftarrow tučniak(X)\}, \\ D &= \{vták(X) : lieta(X)/lieta(X); tučniak(X) : \neg lieta(X)/\neg lieta(X)\}. \end{aligned}$$

Extenziami tejto teórie sú

$$\begin{aligned} \mathcal{E}_1 &= Cn_{FOL}(\{vták(a), tučniak(a), lieta(a)\}), \\ \mathcal{E}_2 &= Cn_{FOL}(\{vták(a), tučniak(a), \neg lieta(a)\}). \end{aligned}$$

$\mathcal{E}_1$ však nepreferuje špecifickejšiu informáciu. Preferovanie špecifickejšej informácie možno vynútiť pomocou semi-normálneho pravidla. Keď použijeme pravidlo

$$vták(X) : lieta(X) \wedge \neg tučniak(X)/lieta(X)$$

miesto pravidla

$$vták(X) : lieta(X)/lieta(X),$$

$\mathcal{E}_1$ už nebude extenziou. ■

4.2.4 Priame teórie hierarchických sietí

Videli sme teda, že preferovanie špecifickejšej informácie možno reprezentovať pomocou semi-normálnych pravidiel. Táto reprezentácia je však veľmi ťažkopádna. Dôsledné využitie takejto techniky by znamenalo, že v každom potenciálne konfliktnom defaultovom pravidle by bolo treba medzi strážami reprezentovať všetky relevantné špecifickejšie vrcholy hierarchie. Zrejme by to zvyšovalo nutnosť často modifikovať defaultové pravidlá.

Etherington [Eth 88] prezentoval elegantný spôsob ako čeliť tejto výhrade: použije sa semi-normálne pravidlo typu $\alpha : \beta \wedge \gamma/\beta$, pričom γ je novozavedený literál, ktorého cieľom je iba signalizovať výskyt výnimky (alebo špecifickej informácie) a konkrétny obsah výnimiek (alebo špecifickejšej informácie) je otvorený, môže sa dynamicky modifikovať implikáciami tvaru $\xi \rightarrow \gamma$, kde ξ môže byť ľubovoľná konkrétna výnimka (alebo špecifickejšia informácia). Napriek tomuto návrhu, dôležité je venovať pozornosť aj iným technikám špecifikácie odvodzovania v hierarchických sieťach.

Ťažkopádnosti, súvisiace s „defaultovou“ sémantickou špecifikáciou odvodzovania v hierarchických sieťach viedli Touretzkého a ďalších [Tou 86, HTT 90, Ste 92, SLe 93] k tomu, že konštruovali špecializované teórie takéhoto odvodzovania. Tieto špecializované teórie možno nazvať aj *priamymi* teóriami hierarchických sietí. Teórie, ktoré hierarchické siete a odvodzovanie v nich charakterizujú prekladom do nejakého známeho formalizmu, možno nazvať aj nepriamymi. Priame teórie boli založené na stotožnení odvodenia s nájdením nejakej cesty v hierarchickej sieti. Jednoduchý základ takéhoto prístupu predstavuje aj naša definícia 4.84. Kľúčovým prvkom odvodenia je identifikácia takej cesty v hierarchickej sieti, ktorá by nebola prevážená alebo neutralizovaná inou cestou. Preferovanie špecifickejšej informácie i existenciu výnimiek možno v tomto prístupe zaviesť veľmi prirodzeným spôsobom. Prednosťou priamych teórií je ich silný intuitívny podklad.

Iba stručne (a zjednodušene, s menej jemným rozlišovaním a pre menej všeobecné ciele) zavedieme základné pojmy podľa veľmi intuitívneho výkladu z [SLe 93], ktorý zasadíme do rámca, zavedeného definíciou 4.102. Hyperhranám (explicitnej informácii o výnimkách) sa však v [SLe 93] pozornosť nevenuje. Nebudeme si ich všímať v tejto časti ani my.

Definícia 4.122 (Polarita hrany) Predpokladajme, že máme hierarchickú sieť $\mathcal{H}$. Ak H je množina jej hrán, $ISA \subseteq H$, $h \in ISA$, potom hovoríme, že polarita h je *pozitívna*, ak $\psi(h) = +ISA$ a *negatívna*, ak $\psi(h) = -ISA$.

Cesta σ je postupnosť hrán $(x_0, x_1), (x_1, x_2), \ldots, (x_{n-1}, x_n)$. Budeme si ju značiť aj ako $(x_0, x_1, \ldots, x_{n-1}, x_n)$, pričom polarita všetkých hrán – až na poslednú – je pozitívna a polarita poslednej hrany môže byť pozitívna alebo negatívna.

Polarita cesty je určená polaritou jej poslednej hrany.

Štart cesty $\sigma = (x_0, x_1, \ldots, x_{n-1}, x_n)$ je x_0, jej *cieľ* je x_n. Oba nazývame aj *koncovými* vrcholmi. Budeme používať aj značenie $begin(\sigma)$ a $end(\sigma)$.

Predpokladáme aj existenciu prázdnej cesty ϵ. Stotožníme ju s ľubovoľným vrcholom. ■

Definícia 4.123 Nech Φ je trieda ciest a $\sigma = (x_0, \ldots, x_n) \in \Phi$. vrchol x je *medzi* koncovými bodmi cesty σ, ak

- buď $x = x_i$, kde $0 < i < n$,
- alebo existuje cesta $\gamma \in \Phi$ taká, že $\gamma = (x_i, y_0, \ldots, y_j, \ldots, y_m, x_l)$, kde $0 \leq i < l \leq n$ a $0 \leq j \leq m$ a $x = y_j$.

■

Teraz ideme definovať dôležitý pojem. Predstavme si množinu ciest, ktoré reprezentujú nejakú informáciu o hierarchických vzťahoch. Informácia, obsiahnutá v niektorej z týchto ciest môže byť prevážená (neutralizovaná, vyvrátená, spochybnená). Stane sa tak vtedy, ak z vrcholu, ktorý leží medzi koncovými bodmi tejto cesty ide priama hrana ku koncovému vrcholu, ale jej polarita je opačná ako polarita danej cesty.

Príklad 4.124 V príklade 4.92 (a na obrázku 4.2) vrchol *kvaker* leží medzi vrcholmi *nixon* a *pacifista*. Ide od neho hrana opačnej polarity (v porovnaní s polaritou cesty $(nixon, republikán, pacifista)$ ku koncovému vrcholu. Samozrejme, platí to aj symetricky. Teda obe cesty sa vzájomne prevažujú, neutralizujú.

Na druhej strane, v príklade 4.118 (a na obrázku 4.9) hrana zápornej polarity od *tučniak* k *lieta* prevažuje cestu kladnej polarity od *karol* k *lieta*, ale cesta zápornej polarity od *karol* k *lieta* nie je prevážená. ■

Definícia 4.125 (Preváženie) Cesta $\sigma = (x_0, \ldots, x_n)$ je *prevážená* v triede ciest Φ, ak vo Φ existuje vrchol x, ktorý je medzi x_0 a x_n a vo Φ je aj hrana $\gamma = (x, x_n)$, pričom jej polarita je opačná ako polarita cesty σ. ■

Definícia 4.126 (Konkatenácia) Ak máme cesty $\sigma = (x_0, \ldots, x_n)$ a $\gamma = (y_0, \ldots, y_m)$, pričom $end(\sigma) = begin(\gamma)$ a inak sú cesty disjunktné, potom konkatenáciou $\sigma \circ \gamma$ je cesta

$$\varsigma = (begin(\sigma), \ldots, x_i, \ldots, end(\sigma), \ldots, y_i, \ldots, end(\gamma)).$$

■

Definícia 4.127 (Konflikt) V triede Φ je konflikt, ak existujú cesty σ a γ také, že $begin(\sigma) = begin(\gamma)$ a $end(\sigma) = end(\gamma)$ a ich polarity sú opačné. Hovoríme aj, že σ a γ sú v konflikte. ■

Teraz sme pripravení zadefinovať, kedy je nejaká cesta v grafe solídnym podkladom pre odvodenie informácie implicitnej v grafe. Východiskom bude vždy nejaký kontext Φ, nejaká množina ciest v danom grafe. Ideme definovať, kedy možno „zdediť" cestu σ v kontexte množiny ciest Φ, kedy možno dediť informáciu zakódovanú v σ, ak akceptujeme Φ.

Definícia 4.128 (Dedenie) Cestu σ možno *dediť* vo Φ práve vtedy, keď

- σ je konkatenáciou ciest z Φ,
- σ nie je v konflikte so žiadnou z ciest z Φ,
- σ nie je prevážená vo Φ.

■

Dedenie považujeme za spôsob získavania implicitnej informácie. Podmienka konkatenácie znamená, že dediť možno iba cesty, ktoré obsahujú minimálne dve hrany. Zodpovedá to presne intuícii o implicitnej informácii: Každá hrana zaznamenáva nejakú explicitnú informáciu. Spájaním hrán do ciest môžeme získať nejakú informáciu navyše.

Nepredpokladá sa, že vo Φ sú všetky cesty z nejakého grafu. Naopak, je tu predstava, že disponujeme podmnožinou ciest, z ktorej možno nekonfliktne dediť. Pomenujme si ju množinou podopretých ciest. Toto pomenovanie používame iba pracovne, intuitívne. Neskôr budeme používať presnejšie pojmy. Φ sa teda vyberá z množiny všetkých ciest. Každá množina podopretých ciest musí obsahovať všetky hrany (elementárne fakty, explicitne zaznamenané v hierarchickej sieti). Okrem nich obsahuje komplikovanejšie cesty tak, aby ich bolo možné dediť podľa definície 4.128 (nekonfliktne a bez preváženia).

Konštrukcia Φ je vlastne získavaním informácie, implicitnej v sieti. Možno si predstaviť rôzne stratégie, ako budovať Φ. Teraz budeme podrobnejšie charakterizovať dve základné stratégie, liberálnu (dôverčivú) a radikálnejšiu (skeptickú).

Definícia 4.129 (Dôverčivá extenzia) Ak je daná množina hrán Γ z hierarchickej siete $\mathcal{H}$, tak množina ciest Φ je jej dôverčivou extenziou práve vtedy, keď

- $\Gamma \subseteq \Phi$,
- $\sigma \in \Phi \setminus \Gamma$ práve vtedy, keď σ možno dediť vo Φ.

■

Teda, množina ciest Φ je dôverčivou extenziou vtedy, keď je „nadstavená“ nad nejakou množinou hrán a žiadna cesta, ktorá vznikne konkatenáciou, nie je v konflikte so žiadnou inou cestou vo Φ a ani nie je prevážená žiadnou inou cestou vo Φ. Možno treba poznamenať, že hovoríme o extenziách grafov, nie o extenziách defautových teórií. Čitateľ však iste vidí korešpondenciu oboch pojmov.

Príklad 4.130 V príklade 4.92 a na obrázku 4.2 máme graf (hierarchickú sieť), z ktorého možno skonštruovať dve dôverčivé extenzie. Prvou z nich je množina všetkých hrán a cesta s pozitívnou polaritou

$$(\mathit{Nixon}, \mathit{kvaker}, \mathit{pacifista}).$$

Druhá popri množine všetkých hrán obsahuje cestu s negatívnou polaritou

$$(\mathit{Nixon}, \mathit{republikán}, \mathit{pacifista}).$$

Teda, v jednej extenzii „možno dediť“, že Nixon je pacifista, v druhej, že nie je pacifista. Dôverčivé usudzovanie pripustí oba tieto dôsledky. ■

Prejdeme k skeptickému usudzovaniu. Konštruovať možno nanajvýš jednu extenziu.

Definícia 4.131 (Stupeň cesty) Dĺžka cesty je počet jej hrán. *Stupeň cesty* $(x_0, \ldots, x_n)$ je dĺžka najdlhšej cesty z x_0 do x_n. ■

Definícia 4.132 (Skeptická extenzia) Nech Γ je množina hrán. Φ je jej skeptická extenzia práve vtedy, keď $\Phi = \bigcup_{i=1}^{\infty} \Phi_i$, pričom

- $\Phi_1 = \Gamma$,
- Φ_{i+1} obsahuje všetky cesty z Φ_i a každú cestu σ stupňa $i+1$ s nasledujúcimi vlastnosťami:
  - σ je konkatenáciou dvoch ciest z Φ_i,
  - neexistuje γ (hrana z Φ_i alebo cesta, ktorú možno dediť vo Φ_i) taká, že

- $begin(\sigma) = begin(\gamma)$ a $end(\sigma) = end(\gamma)$,
- σ a γ sú v konflikte.

■

Uvedená induktívna definícia narábala so stupňom (a nie dĺžkou) cesty. To znamená, že pri skeptickej stratégii je nutné brať do úvahy preváženie a neutralizáciu na všetkých možných cestách s daným štartom a cieľom. Skôr sa konštruujú cesty nižšieho stupňa. Z ciest vyššieho stupňa sa vyberajú iba tie, ktoré neprichádzajú do konfliktu s prv vybratými cestami.

Príklad 4.133 Skeptická extenzia hierarchie z obrázku 4.2 obsahuje iba hrany. Každá cesta, pozostávajúca z dvoch hrán, je v konflikte s alternatívnou cestou. Dôverčivý postoj považuje za podklad pre dedenie v tomto príklade aj alternatívne cesty opačnej polarity. Skeptický postoj však nedovolí dediť, pretože alternatívne cesty vedú ku konfliktu. ■

Dôverčivá extenzia zodpovedá stratégii akceptujúcej akúkoľvek „rozumnú" cestu. Skeptická extenzia vychádza zo stratégie, že cestu možno akceptovať iba vtedy, ak neexistuje alternatívna cesta, ktorá je v konflikte s danou cestou.

Popri skeptickej extenzii, definovanej vyššie, možno rozlišovať ešte dva druhy extenzií, príbuzných skeptickej. V prvom prípade prijímame iba tie cesty, ktoré patria do každej dôverčivej extenzie. V druhom prípade preferujeme špecifickú informáciu.

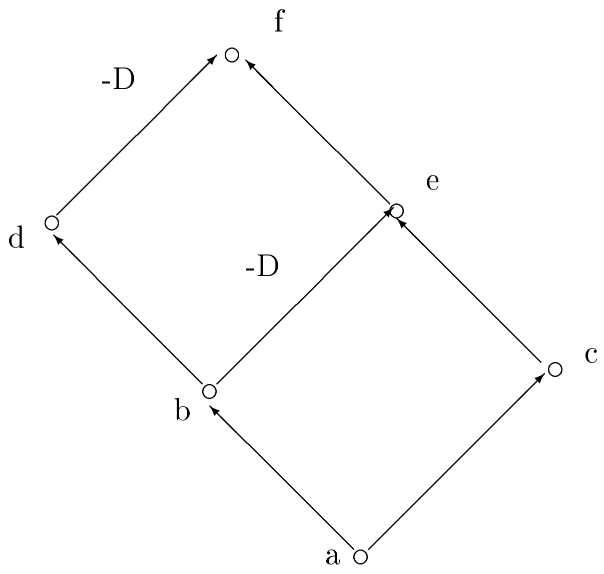
Definícia 4.134 (Ideálne skeptická extenzia) Nech M je množina všetkých dôverčivých extenzií hierarchickej siete $\mathcal{H}$, σ nech je ľubovoľná cesta z $\mathcal{H}$. Potom $\{\sigma : \forall e \in M \; \sigma \in e\}$ je *ideálne skeptická extenzia* siete $\mathcal{H}$. ■

Príklad 4.135 Obrázok 4.10 demonštruje rozdiel medzi skeptickou a ideálne skeptickou extenziou. Obe cesty od vrcholu a k vrcholu e sa neutralizujú (vzájomne prevažujú). Preto do skeptickej extenzie patrí cesta so zápornou polaritou (a, b, d, f). Táto cesta však nepatrí do ideálne skeptickej extenzie, pretože nepatrí do každej dôverčivej extenzie. ■

V priamej teórii hierarchických sietí možno definovať aj zdržanlivý skepticizmus (preferovanie špecifickejšej informácie). Základná myšlienka je taká, že zavedieme reláciu preferencie na dôverčivých extenziách tak, aby preferovanejšia extenzia uprednostňovala špecifickejšiu informáciu. Preferovanou (zdržanlivo skeptickou) extenziou bude minimum v takomto usporiadaní.

My však zvolíme alternatívnu cestu pri formalizácii zdržanlivého skepticizmu. Zdržanlivý skepticizmus budeme charakterizovať v nasledujúcej časti 4.2.5.

Tu si nakoniec zadefinujeme informáciu implicitnú v nejakej hierarchickej sieti. Predstavujeme si vlastne hrany, ktoré sú odvoditeľné z nejakej extenzie.



Obrázok 4.10: Rozdiel medzi skeptickou a ideálne skeptickou extenziou

Definícia 4.136 Nech $\sigma = (x_0, \ldots, x_n)$ je cesta v hierarchickej sieti $\mathcal{H}$. Budeme hovoriť, že hrana $h = (x_0, x_n)$ je ***dôsledok, podopretý cestou*** σ, pritom polaritu tejto hrany stotožníme s polaritou cesty σ. Značenie: $\sigma \triangleright h$.

Množina dôsledkov, ***podopretých extenziou*** Φ je množina hrán $\{h : (\exists \sigma \in \Phi)\ \sigma \triangleright h\}$. ■

Samozrejme, množina dôsledkov závisí od extenzie. Rôzne dôsledky prijímame na základe skeptického alebo dôverčivého prístupu. Pochopiteľne, odlišnosti sú aj medzi rôznymi dôverčivými extenziami.

4.2.5 Zdržanlivý skepticizmus

V tejto časti uvidíme, ako zdržanlivý skepticizmus, vrátane preferovania špecifickejšej informácie možno charakterizovať v tradičnejšom, teoreticky-modelovom prostredí. Konštrukciu postavíme nad formalizmom QRI z kapitoly 3.[23]

V kapitole 3 sme zaviedli pojem minimálneho vyplývania v QRI: Z množiny kláuz E *minimálne vyplýva* klauza ϕ (značenie: $E \models_{min} \phi$) práve vtedy, keď ϕ je pravdivé v každom najmenej konfliktnom modeli E.

Klauzy budeme písať v implikačnej notácii. Budeme používať výrokovologický jazyk (ohraničený na podmnožinu kláuz): Formuly sú literály alebo výrazy tvaru $L \leftarrow A$, kde L je literál a A je atóm (A môže byť prázdny, vtedy $L \leftarrow$ je nejaký fakt).

[23]Podobný prístup možno nájsť v [Gro 97] a [ABM 98].

Teda, používame značne obmedzený jazyk, ale zodpovedajúci hierarchickým sieťam. Implikácie tvaru $L \leftarrow A$ zodpovedajú hranám od A k L. Polarita hrany pritom závisí na tom, či L je pozitívny alebo negatívny literál. Podobne literály $p(a)$ resp. $\neg p(a)$ zodpovedajú hranám od a k p. S polaritou je to obdobne. Je zrejmé, že vrchol a nemá žiadneho predchodcu. Ľahko možno sformulovať – v štýle tvrdení 4.87, 4.88, 4.108 a 4.109 – tvrdenie o korešpondencii množín viet takéhoto jazyka a hierarchických sietí chápaných ako nejaké grafy.

Minimálne vyplývanie zavedené v QRI má niektoré vlastnosti, ktoré sú užitočné pri zdržanlivo skeptickej inferencii o hierarchických vzťahoch:

Ak medzi predpokladmi sú nekonzistentné základné literály, oba z predpokladov minimálne vyplývajú.

Príklad 4.137 Nech $E = \{p, \neg p\}$. Potom očividne $E \models_{min} p$ a $E \models_{min} \neg p$, E je totižto splnené iba v tých modeloch, ktoré obsahujú p aj $\neg p$. ■

Konfliktné vety sa neutralizujú:

Príklad 4.138 Dohodnime sa, že základný atóm prvorádového jazyka je výrokovou premennou. Nech $E = \{$

$$\begin{array}{rcl} kvaker(nixon) & \leftarrow & , \\ republik\acute{a}n(nixon) & \leftarrow & , \\ pacifista(nixon) & \leftarrow & kvaker(nixon), \\ \neg pacifista(nixon) & \leftarrow & republik\acute{a}n(nixon)\}. \end{array}$$

Tento príklad sa v QRI rieši rovnakým spôsobom ako v [HTT 90]: neodvodíme nijaký záver o Nixonovom pacifizme. Z dvoch možných vzájomne nekonzistentných záverov $pacifista(nixon)$, $\neg pacifista(nixon)$ nevyberieme ani jeden.

Tri najmenej konfliktné modely E sú (použijeme iba prvé písmená):

$$\begin{array}{rcl} \mathcal{Q}_1 & = & (k(n), r(n), p(n), \neg p(n)), \\ \mathcal{Q}_2 & = & (k(n), r(n), \neg k(n), \neg p(n)), \\ \mathcal{Q}_3 & = & (k(n), r(n), p(n), \neg r(n)). \end{array}$$

Je zrejmé, že ani $pacifista(nixon)$, ani $\neg pacifista(nixon)$ nevyplývajú minimálne z E. Na druhej strane: $kvaker(nixon)$ aj $republik\acute{a}n(nixon)$ minimálne vyplývajú z E. ■

Pripomenieme si ešte, že minimálne vyplývanie je nemonotónne:

Príklad 4.139 Nech $E = \{$

$$\begin{array}{rcl} vt\acute{a}k(a) & \leftarrow & , \\ lieta(a) & \leftarrow & vt\acute{a}k(a), \\ \neg lieta(a) & \leftarrow & tu\check{c}niak(a)\}. \end{array}$$

Potom $E \models_{min} lieta(a)$.

$lieta(a)$ je pravdivé v každom najmenej konfliktnom modeli E, tým je $(vták(a), lieta(a), \neg tučniak(a))$.

Ale $E' = E \cup \{tučniak(a) \leftarrow\} \not\models_{min} lieta(a)$. Najmenej konfliktný model $E \cup \{tučniak(a)\}$ je napríklad aj $(vták(a), \neg vták(a), tučniak(a), \neg lieta(a))$. Atóm $lieta(a)$ nie je v tomto modeli pravdivý. ■

Zatiaľ však v QRI nemáme prostriedky, ako rozlíšiť špecifickejšiu informáciu a v plnom rozsahu implementovať zdržanlivý skepticizmus [HTT 90].

Príklad 4.140 Adaptujeme príklad 4.139 tak, aby presne zodpovedal obrázku 4.9. $E = \{$

$$\begin{array}{rcl} tučniak(a) & \leftarrow & , \\ vták(a) & \leftarrow & tučniak(a), \\ lieta(a) & \leftarrow & vták(a), \\ \neg lieta(a) & \leftarrow & tučniak(a)\}. \end{array}$$

Najmenej konfliktné modely (modely s jediným konfliktným párom) rozšírenej teórie sú:

$$\begin{array}{rcl} M_1 & = & (vták(a), \neg tučniak(a), tučniak(a), \neg lieta(a)), \\ M_2 & = & (vták(a), tučniak(a), \neg lieta(a), lieta(a)), \\ M_3 & = & (tučniak(a), \neg tučniak(a), vták(a), lieta(a)), \\ M_4 & = & (vták(a), \neg vták(a), tučniak(a), \neg lieta(a)). \end{array}$$

Očividne $E \not\models_{min} \neg lieta(a)$: M_3 je modelom E, ale nie je modelom $\neg lieta(a)$. Keďže však chceme preferovať špecifickejšiu informáciu, $\models_{min}$ nám nevyhovuje. Platí aj $E' \not\models_{min} vták(a)$, čo je neintuitívne. (V oboch extenziách zodpovedajúcej defaultovej teórie platí $vták(a)$, teda aj podľa defaultovej sémantiky by $vták(a)$ mal byť skeptickým dôsledkom E.) ■

Teraz predstavíme spôsob, ako zaviesť do QRI inferenciu, ktorá rešpektuje prioritu špecifickejšej informácie.

Definícia 4.141 Model M implikácie $\phi \leftarrow \psi$ je podopretý práve vtedy, keď aj ψ je splnené v M.

Model množiny viet X jazyka QRI je podopretý, ak je podopretým modelom každej implikácie z X. ■

Spomedzi podopretých modelov si budeme všímať minimálne vzhľadom na inklúziu, t.j. také podopreté modely, ktorých žiadna podmnožina nie je podopretým modelom.

Ako východisko pre sémantickú charakterizáciu preferovania špecifickejšej informácie v QRI nám bude slúžiť nejaká minimálna množina minimálne konfliktných a podopretých modelov.

Teraz prejdeme k niektorým pomocným konštrukciám. Ak chceme vyjadriť preferenciu špecifickejšej informácie, musíme vedieť vyjadriť, ktoré alternatívy pri odvodzovaní sú prevážené. Na to potrebujeme identifikovať konflikty.

Definícia 4.142 (Závislostný graf) Nech X je množina viet jazyka QRI. Definujeme *závislostný graf* $G = (V, H)$ množiny X.

Množinu vrcholov N závislostného grafu G tvoria všetky atómy, ktoré sa vyskytujú ako podformuly v X.

Ak $L \leftarrow A \in X$ a L je atóm, potom závislostný graf obsahuje kladne ohodnotenú (pomocou +) hranu (A, L). Ak L je negatívny literál $\neg B$, potom graf obsahuje negatívne ohodnotenú (pomocou -) hranu (A, B).

Cesta v závislostnom grafe je postupnosť susediacich hrán, posledná z nich môže byť ohodnotená kladne alebo záporne, všetky predchádzajúce kladne. ■

Konvencia 4.143 Budú nás zaujímať iba množiny viet, ktorých závislostný graf je acyklický. ■

Definícia 4.144 *Množina konfliktných vrcholov* závislostného grafu $\mathcal{G}$ je množina tých vrcholov, do ktorých vchádza aj kladne aj záporne ohodnotená hrana. ■

Teraz prejedme k jadru našej konštrukcie. Začneme príkladom.

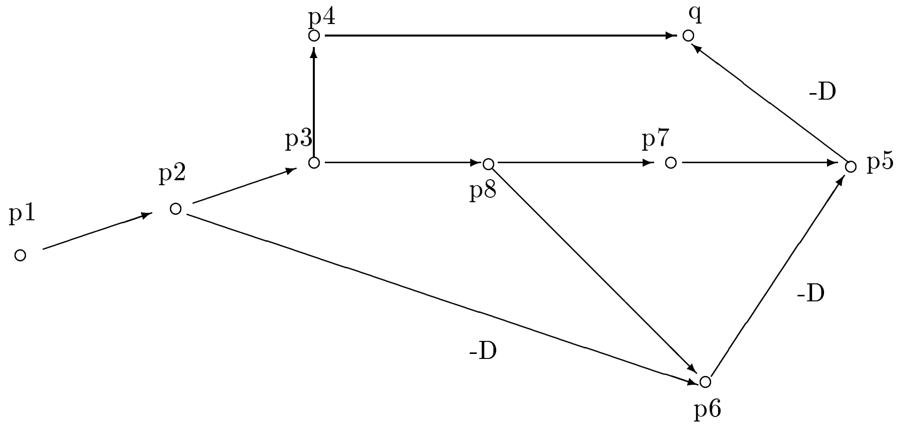
Príklad 4.145 Nech $E = \{$

$$\begin{array}{rcl} p(a) & \leftarrow & , \\ q(a) & \leftarrow & p(a), \\ \neg r(a) & \leftarrow & p(a), \\ r(a) & \leftarrow & q(a)\}. \end{array}$$

Možno treba pripomenúť, že $p(a)$ atď. považujeme za výrokové premenné. Čítať ich však budeme ako v predikátovom počte – a má vlastnosť p (atď.).

V našom príklade, vlastnosť p je špecifickejšia ako vlastnosť q (platí totiž $q(a) \leftarrow p(a)$). Stratégia, založená na priorite špecifickejšej informácie rozhoduje medzi dvoma možnými dôsledkami $r(a)$, $\neg r(a)$ takto: Druhý z nich je založený na špecifickejšej informácii a teda, je odvoditeľný z E. Ale v QRI neplatí ani $E \models_{min} r(a)$, ani $E \models_{min} \neg r(a)$.

Interpretácia $I = (p(a), q(a), \neg r(a), r(a))$ spĺňa všetky vety z E. My však nechceme, aby platilo $r(a) \leftarrow q(a)$, preto z I vypustíme $r(a)$. Teda, z hľadiska preferovania špecifickejšej informácie je lepšou interpretáciou $I' = I \setminus \{r(a)\}$ (musíme si zvyknúť, že nespĺňa implikáciu $r(a) \leftarrow q(a)$, a teda nespĺňa celú E – tú totiž potrebujeme revidovať). ■



Obrázok 4.11: Prevá ženie

Základným krokom našej konštrukcie je redukovanie interpretácií: vylučujeme z nich tie literály, ktoré zaručujú pravdivosť implikácií prevážených špecifickejšou informáciou. Ak chceme zaviesť korektnú definíciu, musíme však byť ostražitejší.

Príklad 4.146 Nech $E = \{p_1 \leftarrow, p_2 \leftarrow p_1, p_3 \leftarrow p_2, p_4 \leftarrow p_3, q \leftarrow p_4, \neg p_6 \leftarrow p_2, p_8 \leftarrow p_3, p_6 \leftarrow p_8, p_7 \leftarrow p_8, p_5 \leftarrow p_7, \neg p_5 \leftarrow p_6, \neg q \leftarrow p_5\}$.

Situáciu dobre ilustruje obrázok 4.11. Na prvý pohľad by sa mali vzájomne eliminovať $q \leftarrow p_4$ a $\neg q \leftarrow p_5$. p_5 je však neutralizované dvoma konfliktnými cestami, medzi ktorými nemožno vybrať. Teda by sa zdalo, že platí q. Situácia je však ešte komplikovanejšia. Špecifickejšia informácia prevažuje p_6. Preto p_5 nie je neutralizované. Teda: q a $\neg q$ sa skutočne eliminujú. ■

Formálnejšie si celú myšlienku môžeme vyjadriť takto: Pripomeňme, že predpokladáme podopreté modely. Ak $\phi_1 \to \phi_2, \ldots, \phi_{n-1} \to \phi_n, \phi_n \to \psi$ a $\phi_i \to \neg\psi$, kde $1 < i < n$, je splnené v J, potom $I = J \setminus \{\psi\}$ je lepšia interpretácia ako J, lebo $\neg\psi$ je podopretá špecifickejšou informáciou.

Zákadná intuícia je asi takáto: Vrstvy vyjadríme závislostným grafom. Ten nás vedie od nejakých východiskových tvrdení k stále viac všeobecným (odvodeným, závislým) tvrdeniam. Tranzitívnosťou implikácie sa bez akýchkoľvek problémov dostaneme až po prvé konfliktné vrcholy. V týchto konfliktných vrcholoch sa môžu rôzne alternatívne cesty usudzovania buď vzájomne neutralizovať, alebo niektorá z nich sa môže presadiť na základe špecifickejšej informácie. Potom usudzovanie môže prebiehať pokojne až po ďalšie najbližšie konfliktné vrcholy. Pri nich zopakujeme vyššie spomenuté prešetrovanie. Pokračujeme dovtedy, kým sa dá (alebo kým je to potrebné). Zámerom nasledujúcej konštrukcie je vyjadriť preferenciu špecifickejšej informácie. Po redukcii interpretácií tak, aby zodpovedali špecifickejšej informácii môžeme vzájomnú neutralizáciu vybaviť pomocou minimálne konfliktných modelov. Ak v niektorom minimálne konfliktnom modeli množiny E platí p a neplatí $\neg p$, v druhom, naopak, neplatí p a platí $\neg p$, potom neplatí ani $E \models_{min} p$ ani $E \models_{min} \neg p$. To znamená, že konfliktná informácia sa neutralizuje. Tým by sme mali zdržanlivý skpeticizmus vyjadrený. Samozrejme, ešte nám treba zadefinovať „lepšie“ interpretácie.

Pripomeňme, že vrcholmi závislostného grafu, prislúchajúceho k množine viet X jazyka QRI, sú výrokové premenné a orientovaná hrana od vrcholu u k vrcholu v vedie vtedy, keď u je antecedentom nejakej implikácie z X a v je konzekventom tej istej implikácie.

Stratifikovaný graf je taký, v ktorom konfliktné vrcholy zavádzajú nejaký rozklad množiny vrcholov:

Definícia 4.147 (Stratifikovaný závislostný graf) Nech je daná množina viet X. Uvažujme jej závislostný graf $\mathcal{G} = (V, H)$. Nech $\mathcal{K}$ je množina všetkých konfliktných vrcholov grafu $\mathcal{G}$.

Budeme hovoriť, že $\mathcal{G}$ je stratifikovaný, ak všetky vrcholy z V roztriedime do vrstiev tak, že $V = V_0 \cup \dots V_n$ pre nejaké n pričom V_i a V_j sú disjunktné pre každé $i \neq j$. Okrem toho

V_0 obsahuje všetky vrcholy na ľubovoľnej ceste od listu po prvý konfliktný vrchol (bez konfliktných vrcholov)

V_{i+1} obsahuje všetky výrokové premenné na ľubovoľnej ceste od nejakého konfliktného vrcholu k takého, že existuje $u \in V_i$ a $(u, k) \in H$, po prvý nasledujúci konfliktný vrchol k' (bez k').

Ak sú splnené uvedené podmienky, $V_0, \dots, V_n$ nazývame *stratifikáciou* $\mathcal{G}$. ■

Po vrstvách budeme budovať najlepšiu interpretáciu (vyjadríme ju ako množinu literálov):

Definícia 4.148 (Najlepšia interpretácia) Nech X je množina viet jazyka QRI, $\mathcal{G} = (V, H)$ je jej stratifikovaný závislostný graf a $V = V_0 \cup \dots V_n$. Potom $\mathcal{I}$, *najlepšiu* interpretáciu X, konštruujeme takto:

I_0 obsahuje všetky literály L také, že $(L \leftarrow) \in X$,

I_1 ak $A \in V_0$ a do A vedie v závislostnom grafe

- pozitívna hrana, tak $A \in I_1$,
- v opačnom prípade $\neg A \in I_1$.

I_{i+1} je najmenšia množina, ktorá obsahuje I_i a okrem toho

- ak $A \notin \mathcal{K}$, $A \in V_{i+1}$, $B \in I_{i+1}$ a v X existuje implikácia $A \leftarrow B$, potom $A \in I_{i+1}$,

- ak $A \notin \mathcal{K}$, $A \in V_{i+1}$, $B \in I_{i+1}$ a v X existuje implikácia $\neg A \leftarrow B$, potom $\neg A \in I_{i+1}$,
- ak $A \in \mathcal{K}$, $A \in V_{i+1}$, v X existujú implikácie $A_1 \leftarrow A_2, \ldots, A_{n-1} \leftarrow A_n$, ktorých antecedenty aj konzekventy sú splnené v I_{i+1} a okrem toho sú v X aj implikácie $A \leftarrow A_1, \neg A \leftarrow A_j$, kde $1 < j < n$, potom $\neg A \in I_{i+1}$,
- ak $A \in \mathcal{K}$, $A \in V_{i+1}$, v X existujú implikácie $A_1 \leftarrow A_2, \ldots, A_{n-1} \leftarrow A_n$, ktorých antecedenty aj konzekventy sú splnené v I_{i+1} a okrem toho sú v X aj implikácie $\neg A \leftarrow A_1, A \leftarrow A_j$, kde $1 < j < n$, potom $A \in I_{i+1}$.

Najlepšou interpretáciou je $\mathcal{I} = I_n = \bigcup_{i=0}^{i=n} I_i$, kde n je počet vrstiev v $\mathcal{G}$. ■

Treba poznamenať, že interpretácia konštruovaná podľa definície 4.148 nemusí byť modelom pôvodnej množiny viet X. Tá množina viet z X, ktoré platia v I, zodpovedá preferovaniu špecifickejšej informácie. I môžeme považovať za výsledok lokálnej revízie, potrebnej pre realizáciu inferencie preferujúcej špecifickejšiu informáciu.

Definícia 4.149 Predpokladajme ľubovoľnú množinu viet X, ľubovoľné

$$(L \leftarrow A) \in X, (\overline{L} \leftarrow B) \in X,$$

kde L a $\overline{L}$ sú komplementárne literály, pričom A, B a $(B \leftarrow A) \in X$ sú splnené v I.

Budeme hovoriť, že interpretácia I *preferuje špecifickejšiu informáciu*, ak $L \in I$ a $\overline{L} \notin I$.

Z konštrukcie $\mathcal{I}$ vidno, že platí:

Tvrdenie 4.150 *Ak X je množina viet jazyka QRI a $\mathcal{I}$ je jej najlepšia interpretácia, tak $\mathcal{I}$ preferuje špecifickejšiu informáciu.*

4.2.6 Komentáre

Niekedy je potrebné reprezentovať spolu so znalosťami o hierarchickej štruktúre aj znalosti, ktoré nemožno vyjadriť jednoduchým zaradením do nejakej hierarchie. Priekopnícku prácu v tomto smere vykonala Leora Morgenstern [Mor 96a, Mor 96b, MoS 97]. Vytvorila expertný systém pre zdravotnícku poisťovňu. Jeho cieľom je podporovať činnosť pracovníkov poisťovne, ktorí odpovedajú na otázky o pokrytí nejakej zdravotníckej služby nejakou poistkou. Najprv treba poznamenať, že pomocou techník umelej inteligencie vytvorila systém oveľa silnejší a flexibilnejší než sú tradične používané systémy. Využila pritom teoreticky dobre fundované koncepty a algoritmy z oblasti hierarchických sietí. Navyše však, reálna a rozsiahla aplikácia ju inšpirovala aj k teoretickým inováciám.

Znalosti, ktoré nemožno priamočiaro zaradiť do nejakej hierarchie, reprezentovala tak, že k jednotlivým vrcholom hierarchickej siete priradila množiny formúl.[24] Pri dedení množín formúl v hierarchickej sieti môže dôjsť k tomu, že množina formúl, ktorú získame tak, že dedíme z rôznych vrcholov, je nekonzistentná. Ideu z [Mor 96a, Mor 96b, MoS 97] zakomponujeme do našej defnície hierarchickej siete.

Definícia 4.151 (Rozšírená hierarchická sieť) Predpokladajme, že $\mathcal{H}$ je hierarchická sieť, N jej množina vrcholov a H množina jej hyperhrán. $\mathcal{H}$ nazveme rozšírenou hierarchickou sieťou vtedy, keď navyše obsahuje:

- F a B, množiny formúl nejakého prvorádového jazyka,
- $L \subseteq N \times 2^F$, reláciu medzi vrcholmi siete a množinou podmnožín F.

Navyše, je definovaný rozklad $L = +strict \cup -strict \cup +weak \cup -weak$. ■

Relácia L priraďuje k vrcholom množiny formúl (podmnožiny F). B je „background knowledge", všeobecné poznanie, ktoré má platiť nezávisle od štruktúry hierarchickej siete (B by sa dalo priradiť ku každému uzlu). Striktné väzby hovoria o tom, že príslušné množiny viet určite (ne)platia. Slabé väzby oznamujú, že príslušné množiny viet v typickom prípade (ne)platia.

V porovnaní s problémami, ktoré sa bežne riešia v kontexte hierarchických sietí, Morgenstern (a reálna, netriviálna aplikačná doména) prichádza s niektorými novými problémami.

V rozšírenej hierarchickej sieti sú dva zdroje nemonotónnosti. Jedným je stavba hierarchickej siete – niektoré hrany v sieti pripúšťajú výnimky. Druhým je to, že relácia L umožňuje – do istej miery – vyjadriť dôveryhodnosť pravidiel, priradených k danému vrcholu.

Už sme si uvedomili, že dôležitou úlohou pri dedení je identifikovať maximálne konzistentné podmnožiny tej množiny formúl, ktorá sa získala dedením. Pri výbere spomedzi viacerých maximálnych konzistentných podmnožín prichádza do úvahy preferencia špecifickejšej informácie i preferovanie niektorých hrán (čo je nový prístup, zavedený Leorou Morgenstern).

Usudzovanie v hierarchických sieťach nemusí brať do úvahy iba prioritu špecifickejšej informácie. V rôznych úlohách a kontextoch si možno predstaviť väčšiu dôležitosť iných hľadísk. Na nemožnosť adekvátneho globálneho syntetizovania rôznych možných preferencií poukázal [DoP 91].

Morgenstern v [MoS 97] dôvodí, že výskum v oblasti nemonotónnej inferencie by mal byť senzitívnejší na problémy a úlohy, iniciované a/alebo zodpovedajúce softwarovému priemyslu.

V tejto časti sme sa zaoberali niekoľkými metódami, ako zefektívniť inferenciu na hierarchických sieťach.

[24] Reprezantačný formalizmus, ktorý zavádza, sa nazýva FAN, Formula Augmented inheritance Network.

Najprv sme spomínali rozšírenie automatického dokazovača pomocou inferencie na typoch. Ukázali sme, že použitie viacsortového jazyka a inferencie na typoch umožňuje napísať získanú informáciu oveľa hutnejšie, čo výrazne zredukuje priestor, ktorý je potrebné pri hľadaní riešení prehľadať, a tým výpočet výrazne zefektívni.

Jadro tejto časti bolo venované hierarchickým sieťam, v ktorých sa vyskytuje viacnásobná dedičnosť a výnimky. Najprv sme pozornosť sústredili na naivný výpočet, na algoritmus najkratšej cesty. Po analýze jeho nedostatkov sme sa pokúsili o sémantickú špecifikáciu hierarchických sietí prekladom do defaultových teórií. Ukázalo sa, že výnimky a viacnásobná dedičnosť prinášajú do výpočtov úlohu riešiť nekonzistentnosti a nejednoznačnosti. V tejto súvislosti sme pristúpili k istej forme preferencie, menovite k preferovaniu špecifickejšej informácie.

V ďalšej časti sme pojednali o zdržanlivom skepticizme, ktorý poskytuje návod, ako sa vyrovnať s prítomnosťou konfliktov v hierarchickej sieti. Na záver sme sa stručne zoznámili s hierarchickými sieťami rozšírenými o formuly.

4.3 Výpočtové aspekty defaultov a hierarchií

Už sme zdôrazňovali, že výpočet extenzií defaultových teórií je vo všeobecnom prípade nerozhodnuteľný. V tejto časti budeme venovať detailnejšiu pozornosť výpočtom v defaultových teóriách a hierarchických sieťach. Opíšeme niektoré experimentálne implementácie. Uvedieme niektoré algoritmy a ich zložitostné charakteristiky. Ukážeme špeciálne prípady všeobecných reprezentačných formalizmov, pre ktoré je výpočet problémov v defaultových teóriách a hierarchických sieťach výpočtovo zvládnuteľný.

V centre pozornosti tohto textu sú sémantické špecifikácie hypotetického usudzovania. Našou ambíciou však je, aby tieto špecifikácie mohli byť podkladom výpočtového modelovania. Preto výpočtové aspekty generovania hypotéz nemôžu byť úplne mimo nášho záujmu. Táto časť, venovaná výpočtovým aspektom defaultových teórií a hierarchických sietí sa však trochu líši od ostatných častí. Text tu nadobúda skôr povahu príručky, informácie o základných faktoch, dôkazy chýbajú. Pre čitateľa s hlbším záujmom o problematiku slúžia odkazy na literatúru.

Túto časť možno čítať súbežne s dodatkom H, kde sú uvedené základné intuície a fakty, súvisiace s problematikou vypočítateľnosti a výpočtovej zložitosti. Čitateľ tam môže nájsť aj zmienku o tom, že problémy umelej inteligencie bývajú často z výpočtového hľadiska veľmi ťažké. V tejto súvislosti možno poznamenať, že problémy, súvisiace s extenziami defaultových teórií sú ťažšie ako „najľahšie“ nerozhodnuteľné problémy – nie sú ani polorozhodnuteľné.[25]

4.3.1 Pesimistický obraz

Tri základné výpočtové problémy, späté s extenziami, sú:

Q1 Patrí daná formula do každej extenzie danej defaultovej teórie?

Q2 Patrí daná formula do niektorej extenzie danej defaultovej teórie?

Q3 Nájdenie extenzie.

Pre prvorádové defaultové teórie sú tieto základné výpočty ťažšie ako nerozhodnuteľné. Ak chceme zistiť, či nejaké defaultové pravidlo d je aplikovateľné na extenziu $\mathcal{E}$, preverujeme, či $\mathcal{E} \models pre(d)$ a či $\mathcal{E} \not\models \neg\phi$ (pre $\phi \in just(d)$). Vieme, že vzťah $\models$ nie je rekurzívny, ale je rekurzívne spočítateľný.[26] Vieme aj, že vzťah $\not\models$ nemôže byť rekurzívne spočítateľný – inak by vzťah $\models$ bol rekurzívny (podľa tvrdenia H.10). To znamená, že pre prvorádové defaultové

[25] ⟨ polorozhodnuteľnosť, aritmetická hierarchia : dodatok H ⟩

[26] Vzťah $\models$ je nejaká množina dvojíc. Teda hovoríme o rozpoznateľnej, ale nie rekurzívnej množine.

teórie výpočtové problémy súvisiace s extenziami nie sú ani polorozhodnuteľné.

Predstavme si teraz defaultovú teóriu $T = (E, D)$, kde E je množina formúl výrokovej logiky a D je množina defaultových pravidiel, ktorých predpoklady, stráže a dôsledky sú výrokovologické formuly. V takomto prípade sú síce vzťahy $\models$ (a teda aj $\not\models$) rozhodnuteľné, ale ich zložitostné charakteristiky sú veľmi zlé. Spomínané základné výpočty $Q1 - Q3$ sú ťažšie ako NP-úplné problémy (ak $P \neq NP$).[27]

Problém $Q2$. Na jeho riešenie treba nedeterministický polynomický algoritmus, ktorý volá orákulum.[28] Volané orákulum je tiež nedeterministickým polynomickým algoritmom.

Podrobnejšie: V časti 4.1 sme videli, že neexistuje všeobecná metóda konštrukcie extenzií defaultových teórií. Čitateľ vie, že najprv treba odhadnúť nejakú množinu viet (kandidáta na extenziu), a potom testovať, či spĺňa požiadavky na extenziu. Teda, máme situáciu, kedy je vhodné využívať konštrukt nedeterministického algoritmu, pracujúceho v dvoch fázach, odhadovania a preverovania. Nech nedeterministický algoritmus $\mathcal{A}$ generuje (odhaduje, vyberá) extenzie. Na preverenie, či predložený odhad je skutočne extenziou, volá orákulum. Orákulum rieši problém splniteľnosti, ktorý je NP-úplný. Preto $Q2$ patrí do triedy Σ_2^P. Otázku, či tento problém je ľahší ako najťažšie problémy z triedy Σ_2^P zodpovedal negatívne Gottlob [Got 92]: problém $Q2$ je Σ_2^P-úplný.

Problém $Q1$ možno transformovať na komplementárny problém: Nech je daná formula ϕ a výrokovologická defaultová teória T. Existuje extenzia $\mathcal{E}$, do ktorej ϕ nepatrí? Teda problém je z triedy $coNP^{NP}$, t.j. Π_2^P. Navyše, je Π_2^P-úplný (tiež [Got 92]).

Aj *problém* $Q3$ je Σ_2^P-úplný. V špeciálnom prípade normálnych defaultových teórií je NP-úplný (oba výsledky [PaS 92]).

Po týchto základných negatívnych informáciách je evidentné, že veľkú pozornosť treba venovať výskumu špeciálnych prípadov defaultových teórií, ktoré majú slušné výpočtové vlastnosti.

4.3.2 Implementácie

Začneme staršou implementáciou. Poole opísal v [Poo 88] nielen originálny prístup k defaultovému (a vôbec, nemonotónnemu) usudzovaniu, ale aj jeho implementáciu. Už vieme, že Poole miesto konštrukcie novej logiky schopnej usudzovať nemonotónne navrhol zmeniť optiku. K defaultovému usudzovaniu navrhol pristupovať ako k tvoreniu teórií, ako k výberu hypotézy z danej množiny jednoduchých hypotéz, umožňujúcich vysvetliť nejaké pozorovania. Usudzovanie takéhoto typu možno chápať ako konštrukciu konzis-

[27] ⟨NP-úplný problém, polynomická hierarchia : dodatok H ⟩

[28] ⟨ orákulum : dodatok H ⟩. Posledný raz pripomínam, že čitateľ sa na pojmy a fakty z oblasti vypočítateľnosti a výpočtovej zložitosti tiež môže pýtať orákula – je ním dodatok H.

tentnej teórie zo známych faktov a vybratých hypotéz tak, aby bolo možné dedukovať (vysvetliť) dané pozorovania.

Poole navrhol jazyk *Theorist*, ktorý obsahoval deklarácie **fact**, **default** a **constraint**. Argumentami sú nejaké formuly.[29] Posledná z deklarácií je určená na situácie, v ktorých má defaultová teória viac extenzií a tvorca systému chce niektoré z nich vylúčiť ako anomálie. Otázky sa formulujú deklaráciou **explain**. Jej argumentom je tiež formula.

Scenár je konzistentná množina $H \cup F$, kde H je podmnožinou danej množiny hypotéz (deklarovaných ako defaulty) a F je množina deklarovaných faktov. Extenziu tvoria všetky dôsledky maximálneho (vzhľadom na inklúziu) a konzistentného scenára. Deklarácia **constraint** umožňuje požadovať, aby scenár bol konzistentný s formulou, ktorá je argumentom constraintu.

Odpoveďou na otázku **explain** ϕ je **no**, ak ϕ nevyplýva z maximálneho scenára alebo **yes** spolu so zoznamom hypotéz, ktoré treba prijať do scenára, ak z takéhoto rozšíreného scenára vyplýva ϕ.

Systém **Theorist** bol implementovaný v Prologu a umožňoval experimentovať s defaultovým usudzovaním, ale aj s diagnostickým usudzovaním alebo učením (poňatým ako tvorenie teórií), pozri aj [PGA 87].

DeReS Implementácie systémov, automatizujúcich nemonotónne usudzovanie nie sú príliš početné. Tu sa zoznámime s jedným z najambicióznejších projektov posledných rokov.

Systém **DeReS** (Default Reasoning System) [CMT 95, CMM 95] slúži na experimentovanie s defaultovými teóriami (a logickými programami, my sa tu však obmedzíme na defaultové teórie). Jeho cieľom je vytvoriť štandardnú základňu (benchmark) pre porovnávanie experimentov v rôznych centrách.[30] Podľa [CMT 95] sú systematické experimenty s implementáciami nemonotónneho odvodzovania nevyhnutné pre lepšie pochopenie jeho výpočtových vlastností.

Systém obsahuje dva základné komponenty. Prvý z nich, *TheoryBase*, využíva *GraphBase*, ktorý vyvinul D.Knuth [Knu 93]. GraphBase generuje grafy a tak slúži pre kompatibilné a porovnateľné výpočtové experimentovanie s grafmi. Problémy na konkrétnych grafoch, vygenerovaných GraphBase-om, kóduje TheoryBase do tvaru defaultových teórií. Po ich zakódovaní možno použiť DeReS a počítať extenzie takto vytvorených defaultových teórií.[31]

V zásade, každý problém, ktorý sa dá riešiť štýlom *generuj a testuj*, možno kódovať pomocou defaultových teórií. Úloha, v ktorej treba na šachovnici rozostaviť osem dám tak, aby sa neohrozovali, je klasickým príkladom:

[29] V prípade defaultov to môže byť komplikovanejšie, ale pre nás je to v tejto chvíli zbytočná podrobnosť.

[30] Systém je verejne dostupný. Možno ho získať pomocou *ftp* na *al.cs.engr.uky.edu*. Ide o súbory `TheoryBase.tar.gz` a `DeReS.tar.gz`, ktoré sa nachádzajú v `/cs/software/logic`.

[31] V prípade, že graf je zakódovaný do logického programu, počítajú sa stabilné modely, pozri kapitolu 6.

Príklad 4.152 (Osem dám [RoZ 95]) Nech $E = \{$

$$\begin{array}{rcl} put(i,j) & \rightarrow & 1 \leq i \leq 8 \wedge 1 \leq j \leq 8, \\ put(i,j) \wedge put(k,l) & \rightarrow & i \neq k \wedge j \neq l \wedge |i-k| \neq |j-l|)\}. \end{array}$$

$D = \{: put(i,j)/put(i,j)|1 \leq i,j \leq 8\}$.

Extenziami tejto defaultovej teórie sú nekonfliktné konfigurácie ôsmich dám. ■

Teraz ilustrujeme, ako možno problémy definované na grafoch kódovať v defaultových teóriách.

Príklad 4.153 (Nezávislé množiny [CMM 95]) Nech je daný graf $G = (V,E)$ a číslo k, nie väčšie ako kardinalita V ($k \leq card(V)$).

Nezávislá množina grafu G veľkosti k je množina uzlov V' ($V' \subseteq V$, $card(V') \geq k$), ak žiadne dva uzly vo V' nie sú spojené hranou $h \in E$.

Majme $v \in V$. Nech množina jeho susedných uzlov je $\{v_1, \ldots, v_k\}$. Všimnime si teraz defaultové pravidlo

$$d_v = \frac{: \neg in(v_1), \ldots, \neg in(v_k)}{in(v)},$$

kde $in(v)$ znamená, že uzol v je v nezávislej množine.

Predstavme si teraz extenziu defaultovej teórie $T = (\emptyset, D)$, kde $D = \{d_v : v \in V\}$. Bude to nejaká množina atómov tvaru $in(v)$ (a ich dôsledkov). Predpoklad súčasnej platnosti $in(v_i)$ a $in(v)$ vedie k sporu, ak v_i a v sú susedné uzly: je vylúčené, aby $in(v)$ patrilo do extenzie $\mathcal{E}$ teórie T, ak do tej extenzie patrí $in(v_i)$ pre aspoň jedno i. Pravidlo d_v teda hovorí: ak možno prijať, že všetci susedia v sú mimo (nejakej jednej) nezávislej množiny, potom v môže byť v (tejto) nezávislej množine.

Pre konkrétny graf množina takýchto pravidiel – pre každý uzol jedno pravidlo – kóduje problém nezávislej množiny na tomto grafe. ■

Prístup z príkladu 4.153 skutočne umožňuje nachádzať nezávislé množiny pomocou extenzií defaultových teórií:

Tvrdenie 4.154 ([CMM 95]) *Majme graf $G = (V,E)$ a defaultovú teóriu $T = (\emptyset, D)$, kde $D = \{d_v : v \in V\}$ a d_v je pravidlo z príkladu 4.153.*

Množina $U \subseteq V$ je maximálna nezávislá množina v G práve vtedy, keď $Cn_{FOL}(\{in(v) : v \in U\})$ je extenzia defaultovej teórie T.

Zhrňme teda: GraphBase generuje grafy. TheoryBase kóduje v defaultovej teórii pre vygenerovaný graf nejaký grafový problém. S defaultovými teóriami, vygenerovanými pomocou TheoryBase, pracuje vlastný systém DeRes. DeRes rieši problémy $Q1 - Q3$ alebo ďalšie, čo s nimi súvisia:

- testuje, či daná teória má extenziu,

- počíta jej extenzie (jednu, všetky),
- zisťuje, či nejaká formula patrí do niektorej extenzie (do všetkých).

Vieme, že definícia extenzie odkazuje na vzťah logického vyplývania. Vieme aj, že preverovanie tohto vzťahu je citlivým miestom výpočtov extenzií. Preto treba hľadať efektívnejšie vypočítateľné špeciálne prípady. Ide napríklad o to, aby sa všeobecný výrokovologický dokazovač nejak špecializoval (na jazyky s nejak obmedzovanou syntaxou), alebo aby množina defaultových pravidiel mala nejaké špeciálne vlastnosti. Vplyv výrokovologického dokazovača a stratifikácie defaultových pravidiel (podľa [Cho 94, Cho 95]) na efektívnu implementáciu odvodzovania v defaultových teóriách analyzovali [CMT 95].

Výsledky experimentov ukazujú, že stratifikácia zlepšuje výkonnosť – umožňuje nájsť extenziu defaultových teórií so stovkami až tisíckami defaultových pravidiel. Čas výpočtu bol výrazne najlepší pre obmedzený výrokovologický dokazovač, aplikovateľný iba na defaultové teórie bez disjunkcie (predpoklady, stráže a dôsledky sú konjunkciami literálov, pozri definíciu 4.180). Opäť je tu príklad reprezentačného formalizmu s limitovanou vyjadrovacou silou, zato však so slušnou výpočtovou efektívnosťou.

GADEL Extenzie defaultových teórií hľadá aj experimentálny systém Genetic Algorithms for Default Logic (GADEL) [NSS 00]. Ako hovorí jeho názov, pri výpočte extenzie sa využíva genetické programovanie. Ide o výpočet štýlu *generuj a testuj*. Genetický algoritmus generuje kandidátov na extenzie. Experimentálne výsledky sú slušné (úspešne sa porovnávali so systémom DeReS). Navyše, korektnosť metódy je opretá o teoretický výsledok práce [NSS 00].

Exten Ako ďalší príklad implementácie defaultového usudzovania možno uviesť systém *Exten*. Systém bol implementovaný v rámci projektu CIN, zameraného na výskum a vývoj nástrojov pre inteligentné informačné systémy [WiA 98, AnW 97]. Systém počíta nielen klasické extenzie a problémy s nimi súvisiace, ale aj obmedzované extenzie. Významným príspevkom je schopnosť revidovať extenzie po pridaní novej informácie. V defaultovej logike sa tento problém neštudoval, predpokladalo sa, že extenzia sa po modifikácii teórie vypočíta znova (samozrejme, s výnimkou tých prípadov, kde možno využiť semi-monotónnosť). Exten sa venuje práve problémom nekonzistentnosti novej informácie a prv vypočítanej extenzie, problémom revízie extenzií. V rámci tohto projektu sa dosiahli pôvodné teoretické výsledky, ktoré sa využili v experimentálnej implementácii. Stratifikácia defaultových pravidiel sa využíva aj pri revíziách.

BenInq Ako posledný príklad implementácie uvedieme expertný systém *BenInq* [MoS 97], pozri aj časť 4.2.6. Tento systém využíva reprezentáciu znalostí pomocou hierarchickej siete.

Ide o expertný systém, vyvinutý pre zdravotnú poisťovňu a jeho cieľom je poskytovať podporu pri informovaní klientov o tom, čo z danej zdravotníckej služby im možno hradiť. Umožňuje aj pracovníkovi poisťovne (policy maker) modifikovať bázu znalostí – okrem iného aj zaraďovať do nej nové produkty poisťovne. (Systém pozná vnútornú a vonkajšiu formu reprezentácie znalostí, vonkajšia je prístupná pracovníkom poisťovne.)

Zdravotná poisťovňa, ktorá expertný systém využíva, poskytuje tisícky produktov. Každý z týchto produktov ponúka mnoho služieb a v jeho rámci platí veľa pravidiel.

Systém uvádzame preto, lebo ide o rozsiahlu a novátorskú implementáciu pre poisťovnícky priemysel a aj preto, že významne stimuluje teoretický výskum. Stimuluje ho jednak niektorými konkrétnymi problémami, na ktoré sa narazilo pri návrhu a implementácii priemyselného produktu, ale aj preto, lebo explicitne formuluje niektoré výzvy pre teoretický výskum.

Implementácia je novátorská i tým, že výrazne prekonáva štandardnú softwarovú podporu pre zodpovedanie otázok klientov poisťovne. Tradične sa používali softwarové produkty dvoch druhov. Jeden z nich využíval textové obrazovky, na ktorých sa nachádzajú základné informácie o danom produkte. Druhý pracoval s tabuľkami, ktoré k danému kódu poistenia a danému kódu zdravotníckej služby obsahovali informácie o tom, či je služba hradená z danej poistky a v akom rozsahu. Oba prístupy majú vážne nedostatky. Okrem nutnej znalosti kódov v druhom z uvedených prístupov sú to predovšetkým:

- Nedostatočná flexibilita schopnosti zodpovedať otázky (systém je pripravený na obmedzený rozsah otázok). Treba dodať, že nie je možné meniť mieru detailnosti ani pri formulácii otázky, ani pri jej zodpovedaní.
- Nie je možné sledovať súvislosti medzi rôznymi textovými obrazovkami (resp. informáciami na nich uvedenými).
- Systém nie je schopný usudzovať so znalosťami, ktoré má k dispozícii.
- Modifikácie sú veľmi náročné – manuálne treba uskutočniť modifikáciu tej istej informácie na viacerých miestach.

Aplikačná doména umožňuje využiť hierarchický pohľad na znalosti: aj zdravotnícke služby, aj poistky sú organizované hierarchicky. Obe hierarchie sú prepojené: systém obsahuje informáciu o tom, ktorá poistka umožňuje hradiť aké služby. Navyše, platia mnohé pravidlá, regulujúce poisťovací priemysel a jednotlivé prípady. Vzhľadom na hierarchickú stavbu, možno pravidlá dediť: ak niečo platí pre chirurgické služby, malo by to platiť aj pre plastickú chirurgiu. Lenže, máme klasickú hierarchickú sieť s viacnásobným dedením, výnimkami a konfliktami.

Ide o typickú oblasť nemonotónneho usudzovania. Podnety z tejto veľkej aplikácie smerujú aj k teoretickým inováciám [Mor 96b]. Popri hierarchicky organizovaných znalostiach existujú mnohé znalosti (pravidlá), ktorých reprezentácia v rámci hierarchie by bola možná iba vďaka veľkému znásilneniu prirodzeného vyjadrenia. Navyše, spôsobila by výrazný nárast siete. Preto sa využíva rozšírená hierarchická sieť [Mor 96a] (pozri aj časť 4.2.6): ku každému uzlu siete je priradená množina formúl. Samozrejme, dedením formúl môžu vzniknúť nekonzistentnosti (napríklad, niektorá trieda služieb môže vyžadovať 20%-ný spolupodiel klienta, jej špeciálny prípad však iba 10%-ný). *BenInq* rieši tieto nekonzistentnosti iteratívnou konštrukciou maximálnej konzistentnej podmnožiny formúl. Takýchto množín môže byť viac. Pri výbere niektorej z nich sa využívajú informácie zo siete (preferovanie špecifickej informácie, preferovanie niektorej hrany pred inou hranou). Zvýšenú výpočtovú náročnosť, spôsobenú ošetrovaním nekonzistentností, systém kompenzuje tým, že roztriedi formuly do disjunktných tried tak, aby nekonzistentnosti mohli vzniknúť iba v rámci jedinej triedy.

Záverom k niektorým výzvam teoretickému výskumu. *BenInq* sa líši od naivných expertných systémov tým, že využíva teoreticky dobre fundovaný konceptuálny aparát a efektívne algoritmy, ktoré boli vyvinuté pre túto reprezentáciu (pozri [Ste 92]). Na druhej strane, táto aplikácia umožnila poukázať na niektoré slabiny súčasného stavu výskumu v oblasti nemonotónneho usudzovania. Morgenstern vo svojej pozvanej prednáške na IJCAI'97 [Mor 97] upozornila na priepasť medzi výskumom v oblasti nemonotónnej inferencie na strane jednej a priemyselnými aplikáciami na druhej strane. Táto priepasť je tým zarážajúcejšia, že takéto aplikácie poskytujú prirodzenú doménu pre nemonotónne usudzovanie – treba pri nich odvodzovať z neúplnej informácie, vyskytujú sa tam výnimky a aj nekonzistentnosti. Navrhuje preto, aby výskumníci v oblasti nemonotónnej inferencie vyhľadávali priemyselné aplikácie. Zdôrazňuje strategický význam takéhoto pootočenia pozornosti pre celú výskumnú oblasť. Môže priniesť aj nové problémy, užitočné a zaujímavé pre teoretický výskum.[32]

4.3.3 Elementárny algoritmus

Vieme už, že vo všeobecnom prípade nemáme algoritmus na konštrukciu extenzie. Extenziu odhadujeme a potom tento odhad preverujeme, napríklad podľa definície. Definícia však využíva pojmy logického vyplývania a konzistentnosti, ktoré sú vo všeobecnom prípade nerozhodnuteľné. Algoritmus 4.155 (tabuľka 4.3) dáva nejakú schému konštrukcie. Samozrejme, iba vo veľmi voľnom zmysle slova. Po prvé, je nedeterministický. Po druhé, obsahu-

[32] Morgenstern vo svojej pozvanej prednáške na LPNMR'99 [Mor 99] informuje aj o novších pokusoch v oblasti riešenia problémov reálneho sveta pomocou techník nemonotónneho usudzovania. O zreteľnom pokroku v oblasti implementácií nemonotónneho usudzovania sa možno dočítať v časti 8.5. Stále však ešte chýba väčší počet priemyselných aplikácií.

Algoritmus 4.155 (Etherington [Eth 88])
vstup: *konečná defaultová teória* (E, D) *– obsahuje iba konečný počet premenných, konštánt, žiadne funkčné symboly*
výstup: *extenzia teórie*
Dôležité premenné: h, H *– kandidáti na extenziu,* GD *– generujúce defaultové pravidlá*

$H_0 := E; j := 0$
repeat
  $h_0 := E; GD_0 := \emptyset; i := 0;$
  repeat
    $D_i := \{\alpha : \beta/\gamma \in D \mid h_i \models \alpha,\ h_i \not\models \neg\beta,\ H_j \not\models \neg\beta\};$
    if $D_i \setminus GD_i \neq \emptyset$ **then**
      vyber δ *z* $D_i \setminus GD_i$*;* $GD_{i+1} := GD_i \cup \{\delta\}$*;* $h_{i+1} := h_i \cup \{cons(\delta)\}$
    $i := i + 1$
  until $(D_i \setminus GD_i) = \emptyset$*;*
  $j := j + 1$*;*
  $H_j := h_i$*;*
until $H_j = H_{j-1}$

Tabuľka 4.3: Výpočet extenzie

je odvolávky na vzťah vyplývania (môžeme to chápať ako otázky nejakému orákulu).

Algoritmus 4.155 predpokladá na vstupe konečnú defaultovú teóriu, t.j. teóriu s konečným počtom konštánt, predikátových symbolov a defaultových pravidiel. Pracuje v dvoch cykloch. Vnútorný cyklus nedeterministicky vyberá defaultové pravidlá, aplikovateľné v danej chvíli. Výsledok aplikácie všetkých aplikovateľných pravidiel je po vypadnutí z vnútorného cyklu uchovaný v h_i. Odovzdá sa do „nového“ H_j – kandidáta na extenziu. H_j predstavuje vlastne aproximáciu extenzie.

Vonkajší cyklus „koriguje odhady“ vnútorného cyklu. Odovzdáva s „novým odhadom“ H_j riadenie vnútornému cyklu a ten pracuje opäť so všetkými pravidlami, aplikovateľnými na tento nový odhad (samozrejme, aj na „lokálne“ h_i). „Nové“ H_j nemusí byť nadmnožinou „starého“ H_j. Aproximácie môžu aj oscilovať. Extenziu dostaneme vtedy, keď narazíme na pevný bod (t.j., keď sa postupné aproximácie ustália). Nedeterministický výber umožňuje výpočet rôznych extenzií v rôznych behoch.

Z časti 4.3.1 vieme, že nad jazykom predikátovej logiky je problém, ktorý by mal riešiť uvedený algoritmus, ťažší ako polorozhodnuteľný. Ak sa pohybujeme v prostredí výrokovej logiky, problém je Σ_2^P-úplný. Ďalej nás budú zaujímať rôzne limitované verzie výrokovej logiky a budeme sa venovať otázke, či existujú pre ne varianty tohto algoritmu, ktoré by počítali efektívne.

Tvrdenie 4.156 *Nech výpočet podľa algoritmu 4.155 konverguje v* H_n.
Potom $Cn_{FOL}(H_n) = \mathcal{E}$ *vtedy a len vtedy, keď* $\mathcal{E}$ *je extenziou defaultovej teórie* (E, D).

Nie je však zaručená konvergencia tohto algoritmu pre ľubovoľné konečné defaultové teórie (i keď majú extenzie). Tvrdenie 4.156 je totiž vyslovené iba pre konvergujúce výpočty.

V niektorých prípadoch však výpočet zaručene konverguje:

Tvrdenie 4.157 *Pre konečné usporiadané a sieťové defaultové teórie algoritmus 4.155 vždy konverguje.*

Tvrdenie 4.158 *Pre konečné normálne teórie algoritmus 4.155 okamžite konverguje:* $Cn_{FOL}(H_1)$ *je vždy extenziou.*

Dôkaz zo semi-monotónnosti. ■

Môžeme prejsť k hľadaniu špeciálnych prípadov, kedy je výpočet problémov súvisiacich s výrokovologickými defaultovými teóriami efektívny. Budeme sa venovať defaultovým teóriám v nejako limitovanom výrokovologickom jazyku. Najprv zjednodušený problém – nebudeme hovoriť priamo o výpočte extenzií, ale o výpočte preferovaných modelov (ktorý súvisí s výpočtom extenzií, pozri časť 4.1.3).

4.3.4 Zložitosť nájdenia preferovaných modelov

V tejto a v nasledujúcich častiach začnem stručným zhrnutím (písaným kurzívou) prezentovaných výsledkov. Čitateľ, ktorý sa nezaujíma o detaily, môže zvyšok obskočiť.

Zhrnutie:
V tejto časti sa možno zoznámiť s nepriamymi zložitostnými charakteristikami defaultového usudzovania, ktoré získali Selman a Kautz. S nepriamymi preto, že ich hľadali v zjednodušenom prostredí. Defaultové usudzovanie reprezentovali v [SKa 90] reláciou preferencie na modeloch. Z časti 4.1.3 vieme, že extenziu možno iba za istých okolností stotožniť s najpreferovanejšou množinou modelov. Napriek zjednodušeniam však získané výsledky naznačia, že aj špecializácie problémov súvisiacich s výpočtom extenzií sú z výpočtového hľadiska ťažké.

Selman a Kautz pracovali s defaultami tvaru $\alpha \rightsquigarrow x$, *kde* α *je množina literálov a* x *literál. Môžeme to považovať za isté kódovanie normálnych defaultových pravidiel.*

Defaulty umožňujú nejaké usporiadanie na modeloch: Aplikácia defaultu d *na nejaký model* m *vedie k modelu* m', *v ktorom platí* x. *Z hľadiska* d

je m' lepší ako m. Takéto usporiadanie má pre konečné množiny defaultov minimum. V časti 4.1.3 sme zistili, že minimálny model – resp. minimálna trieda modelov – súvisí za istých podmienok s extenziou defaultovej teórie. Sústredenie pozornosti na minimálne modely má však aj samo o sebe zmysel: Teória, ktorá reprezentuje neúplné poznanie, má rôzne modely. Spomedzi nich sú najzaujímavejšie tie, ktoré sú najpreferovanejšie (vzhľadom na nejaké rozumné kritérium). Miesto odvodzovania potom možno problémy riešiť vyhľadávaním v týchto modeloch.[33]

Selman a Kautz problém nájdenia minima študovali pre päť prípadov. Vo všeobecnom prípade, pre defaulty tvaru $\alpha \rightsquigarrow x$, je to problém NP-ťažký. Ak sa rozhodneme preferovať špecifickejšiu informáciu, je tiež NP-ťažký. Keď jazyk zúžime na hornovské defaulty (α obsahuje iba pozitívne literály), existuje polynomický algoritmus, ktorý ho rieši. Keď však v jazyku hornovských defaultov preferujeme špecifickejšiu informáciu, problém sa opäť stáva NP-ťažkým. Ak by sme množiny defaultov syntakticky obmedzili tak, že by boli acyklické, problém by sa opäť stal riešiteľný algoritmicky v polynomickom čase.

Teraz prechádzame k detailom.

Konvencia 4.159 Budeme predpokladať výrokovologický jazyk s konečnou množinou výrokových premenných $P = \{p_1, \ldots, p_n\}$.

Literály tohto jazyka budeme značiť ako x, x_i. Literál komplementárny k literálu x budeme značiť ako $\overline{x}$. Množiny literálov ako α, β. ■

Definícia 4.160

- Defaultové pravidlo je výraz tvaru $\alpha \rightsquigarrow x$.
- Default[34] d je hornovský práve vtedy, keď α obsahuje iba pozitívne literály.
- Default $d = \alpha \rightsquigarrow x$ je aplikovateľný na modeli m práve vtedy, keď $m \models \alpha$.

■

Definícia 4.161 (Podmienka špecifickosti) Nech $d \in D$, kde D je množina defaultov. Pravidlo $d = \alpha \rightsquigarrow x$ je *blokované* na modeli m a množine D vtedy a len vtedy, keď $\exists d' \in D \; ((d' = (\alpha \cup \beta) \rightsquigarrow \overline{x}) \;\wedge\; (m \models \alpha \cup \beta))$ ■

Pravidlo je blokované iným pravidlom, ak ich dôsledky sú konfliktné a blokujúce pravidlo sa opiera o špecifickejšiu informáciu: V porovnaní s množinou podmienok α je množina $\alpha \cup \beta$ menej všeobecná.

[33] Pozri aj kapitolu 8, kde sa hovorí o intenzívnej reprezentácii a o technike testovania modelov.

[34] Termín „default" tu budeme používať ako skratku za „defaultové pravidlo".

Definícia 4.162 (Preferovanie špecifickejšej informácie) Pravidlo $d = \alpha \leadsto x$, $d \in D$ je š-aplikovateľné práve vtedy, keď

- $m \models \alpha$,
- d nie je blokované v m a D.

■

Definícia 4.163 (Acyklické defaulty) Nech je daná množina defaultov D. Definujeme graf $G(D) = (U, H)$ takto:

- ak sa výroková premenná p_i vyskytuje v D, potom v U existuje uzol ohodnotený ako p_i,
- ak $\alpha \leadsto x \in D$, $(p_i \in \alpha) \vee (\neg p_i \in \alpha)$, $(p_j = x) \vee (\neg p_j = x)$, tak $(p_i, p_j) \in H$.

Množina defaultov D je acyklická práve vtedy, keď $G(D)$ je acyklický. ■

Definícia 4.164 (Usporiadanie na modeloch) Ak je na nejaký model m aplikovateľný default d, potom je model m' d-lepší ako m ($m' \leq_d m$) práve vtedy, keď sa od m líši iba tým, že splňuje $cons(d)$. Budeme to značiť ako $m' = m \mid cons(d)$.

Ďalej, $m \leq_D m'$, ak pre nejaké $d \in D$ je $m \leq_d m'$ alebo keď existuje model m'' taký, že $m \leq_D m''$ a $m'' \leq_D m'$. ■

Ak platí $m \leq_D m'$, budeme hovoriť, že m je preferovanejší (lepší) ako m'.

Niekedy budeme hovoriť o modeli množiny defaultov. Myslíme tým model defaultovej teórie s prázdnym E. Ďalej sa zväčša predpokladá prázdna množina viet E.

Definícia 4.165 Minimálny (najlepší) model defaultovej teórie $T = (E, D)$ je taký model m, pre ktorý

- $m \models E$,
- neexistuje model m' taký, že $m' \models E$ a $m' \leq_D m$.

■

Teraz budeme definovať päť typov defaultov (trochu, ale iba kozmeticky, zmeníme výklad z [SKa 90]):

Definícia 4.166 Triedu množín defaultových pravidiel

- budeme značiť ako $\mathcal{D}$,
- ktorých aplikovanie je obmedzené na š-aplikovateľnosť, označíme ako $\mathcal{D}^+$,

- hornovských budeme značiť ako $\mathcal{DH}$,
- hornovských, ktorých aplikovanie je obmedzené na š-aplikovateľnosť, označíme ako $\mathcal{DH}^+$,
- acyklických hornovských, ktorých aplikovanie je obmedzené na š-aplikovateľnosť, označíme ako $\mathcal{DH}_a^+$.

■

Algoritmus 4.167
vstup: *model M_0 a množina defaultov DH z tvrdenia 4.169*
výstup: *najlepší model DH*

$POS(M_0, DH)$ *(* volanie procedúry POS špecifikovanej nižšie s parametrami M_0 a DH *)*
procedure $POS(m, D)$
if $(\exists d = \alpha \rightsquigarrow p \in D)\ m \models (\alpha \cup \{\neg p\})$
  then return $POS(m|p, DH)$ *(* rekurzívne volanie procedúry POS *)*
  else return m

Tabuľka 4.4: Algoritmus pre $\mathcal{DH}_s$

Pre každú z týchto tried študovali [SKa 90] výpočtovú zložitosť problému hľadania najpreferovanejšieho modelu. Máme teda päť verzií tohto problému, pre každú triedu jednu. Verzie problému označíme tak, že k označeniu zodpovedajúcej triedy (napríklad $\mathcal{DH}$) pripojíme index s (napríklad, $\mathcal{DH}_s$ označuje problém nájdenia najpreferovanejšieho modelu pre teórie typu $\mathcal{DH}$).

Tvrdenie 4.168 *Problém $\mathcal{D}_s$ (problém nájdenia najlepšieho modelu množiny defaultov z triedy $\mathcal{D}$) je NP-ťažký.*

Dôkaz vychádza zo zobrazenia $3CNF$ formúl (formúl v konjunktívnej normálnej forme, obsahujúcich presne tri literály v klauze) do nejakej množiny defaultových pravidiel. Klauze $c = \{x_i, x_j, x_k\}$ sa priradia pravidlá $f_D(c) = \{$

$$\begin{array}{lcl} \{\neg x_i, \neg x_j\} & \rightsquigarrow & x_k, \\ \{\neg x_j, \neg x_k\} & \rightsquigarrow & x_i, \\ \{\neg x_k, \neg x_i\} & \rightsquigarrow & x_j\} \end{array}$$

a množine kláuz C sa priradí $\bigcup_{c \in C} f_D(c)$. Využije sa potom, že platí: m je minimálny model $f_D(C)$ práve vtedy, keď $m \models C$. ■

Algoritmus 4.170 (Min-Model-DH)
vstup: *množina hornovských defaultových pravidiel DH, konzistentná množina literálov E, množina výrokových premenných P, obsiahnutých v DH a E.*
výstup: *najlepší model m_{min} množín E a DH (minimálny vzhľadom na $\leq_D$)*

begin
 $m_0 := E \cup \{\neg p | p \in P \wedge p \notin E\}; \delta := \emptyset$ *(* E je množina literálov *)*
 loop
  $D := DH \setminus \{d \in DH | d = \alpha \leadsto p \wedge p \in \delta\}$
  $m_{pos} := POS(m_0, D)$
  $\beta := \{p | \neg p \in E \wedge m_{pos} \models p\}$ *(* treba vyriešiť nesúlad E a m_{pos} *)*
  $\gamma := \beta \setminus NEG(\beta, m_{pos}, D)$
  if $\gamma = \emptyset$
   then $m_{min} := m_{pos} | \overline{\beta}$; **return**
   else $\delta := \delta \cup \gamma$
 end loop
end

procedure $NEG(\beta, m, D)$
if $(\exists d \in D)\ ((d = \alpha \leadsto \neg p)\ \wedge\ (p \in \beta)\ \wedge\ (m \models \alpha)\ \wedge\ (\alpha \cap (\beta \setminus \{p\}) = \emptyset)$
 then return $\{p\} \cup NEG(\beta \setminus \{p\}, m, D)$
 else return $\emptyset$

Tabuľka 4.5: Algoritmus Min-Model-DH

Vzhľadom na lineárnu zložitosť výrokovologických hornovských teórií[35] sa Selman s Kautzom venovali analýze hornovských defaultových teórií. Problém $\mathcal{DH}$ je skutočne zvládnuteľný. Najprv si všimnime prípad defaultovej teórie, kde $E = \emptyset$.

Tvrdenie 4.169 *Nech $DH \in \mathcal{DH}$, P je množina výrokových premenných, ktoré sa vyskytujú v DH. M_0 je model taký, že $M_0 \models \{\neg p | p \in P\}$.*

Potom algoritmus 4.167 rieši problém $\mathcal{DH}_s$ v čase $O(nk)$, kde n je počet výskytov literálov v DH a k je počet výrokových premenných v P.

Aj v prípade defaultovej teórie s neprázdnou množinou E (obsahujúcou iba literály) existuje polynomický algoritmus 4.170, pozri tabuľku 4.5. Algoritmus používa túto notačnú konvenciu: ak β je množina literálov, potom $\overline{\beta} = \{l : \overline{l} \in \beta\}$. Teda $\overline{\beta}$ obsahuje komplementy všetkých literálov z β.

[35]Pozri algoritmus 2.47.

Algoritmus 4.170 štartuje z m_0, „najnegatívnejšieho“ modelu, spĺňajúceho E. Na odstraňovanie nadbytočných negatívnych literálov slúži procedúra POS, ktorá bola definovaná v tabuľke 4.4. m_{pos} určite spĺňa pozitívne literály z E (na základe špecifikácie POS). V prípade, že E obsahuje nejaké negatívne literály, môže sa stať, že m_{pos} nespĺňa E. Preto sa do β zozbierajú atómy z konfliktných literálov. Teraz treba vyškrtať nadbytočné atómy. To robí procedúra *NEG*. Jej vstupmi sú: β – množina atómov, m – model, D – množina defaultov. Procedúra *NEG* vracia všetky tie atómy, pre ktoré existujú pravidlá z D s komplementárnym dôsledkom, nazvali sme ich falzifikujúce defaulty (napríklad pre atóm p to je $cons(d) = \neg p$). Podrobnejšie, postupne vyberá pravidlá $d \in D$, ak dôsledok $cons(d) = \neg p$ je komplementárny k nejakému atómu p z β, ak predpoklady α tohto pravidla sú splnené v m a α nie je – až na p – znehodnotené konfliktnou množinou β.

Ak *NEG* našla pre každý atóm z β falzifikujúci default, teda $\gamma = \emptyset$, potom m_{min} získame z m_{pos}: zmeníme ho tak, aby spĺňal komplementy všetkých atómov z β. V opačnom prípade sa do množiny δ zozbierajú všetky atómy, ku ktorým neexistuje falzifikujúci default a v ďalšom behu cyklu sa (pomocou modifikovaného δ) osekáva množina defaultových pravidiel a konštruuje sa nové m_{pos}.

Algoritmus 4.170 je polynomický:

Tvrdenie 4.171 *Nech $DH \in \mathcal{DH}$, E je množina literálov a P nech je množina výrokových premenných obsiahnutých v DH a v E.*

Algoritmus 4.170 nájde minimálny model defaultovej teórie (E, D) v čase $O(nk^2)$, kde n je počet výskytov literálov v DH a k je počet výrokových premenných v P.

Podmienka špecifickosti robí výpočet v hornovských teóriách *NP*-ťažkým (pripomeňme, že keď k označeniu triedy defaultov T pripojíme index s, potom T_s nám označuje problém nájdenia najpreferovanejšieho modelu pre triedu defaultov typu T):

Tvrdenie 4.172 *Problém $\mathcal{DH}^+_s$ je NP-ťažký.*

Vidno teda, že relatívne malá zmena v reprezentačnom formalizme ($\mathcal{DH}$ vs. $\mathcal{DH}^+$) môže viesť od zvládnuteľného reprezentačného formalizmu k nezvládnuteľnému.

Dôsledok 4.173 *Problém $\mathcal{D}^+_s$ je NP-ťažký.*

Zmenu vo výpočtovej zložitosti spôsobuje aj pridanie neprázdnej množiny hornovských kláuz E k defaultovej teórii typu $\mathcal{DH}$ (v tvrdení 4.171 sa predpokladala E tvorená iba literálmi):

procedure ORDER(P, D)
vstup: acyklická množina defaultov D a množina výrokových premenných P z D
výstup: zoznam výrokových premenných $(p_{i_1}, \ldots, p_{i_k})$ z P takých, že pre každú dvojicu premenných p_{i_l} a p_{i_m} platí: ak $l < m$, potom neexistuje cesta z p_{i_m} do p_{i_l} v grafe $G(D)$.
function ELEM
vstup: nejaký zoznam
výstup: množina prvkov zoznamu
function HEAD
vstup: nejaký zoznam
výstup: prvý člen zoznamu
function TAIL
vstup: zoznam
výstup: chvost zoznamu (vstupný zoznam, až na prvý člen)

Tabuľka 4.6: Procedúra a funkcie volané algoritmom 4.176

Tvrdenie 4.174 *Nech je daná množina hornovských defaultov DH a množina hornovských klauz E, problém nájdenia minimálneho modelu teórie $T = (E, DH)$ je NP-ťažký.*

Teraz prejdeme k acyklickým hornovským defaultovým teóriám. Uvedieme ďalší polynomický algoritmus 4.176, tentokrát pre problém $\mathcal{DH}^+_{a\ s}$. Tento algoritmus možno nájsť v tabuľke 4.7. V tabuľke 4.6 je stručne charakterizované vstupno-výstupné správanie procedúry a funkcií, ktoré sa v algoritme 4.176 využívajú. Procedúra $ORDER$ využíva predpoklad acyklickosti.

Môžeme prejsť k vlastnému algoritmu 4.176. Postupne vyberáme výrokové premenné a defaultové pravidlá, ktoré ich zavádzajú. Algoritmus identifikuje blokujúce sa pravidlá. Ostatné pravidlá postupne dopĺňajú model m_{part}. Nakoniec sa m_{part} zúplní a dostávame najlepší model.

Tvrdenie 4.175 *Nech $DH^+_a \in \mathcal{DH}^+_a$. Nech P je množina výrokových premenných obsiahnutých v DH^+_a.*

Algoritmus Min-Model-DH^+_a nájde minimálny model DH^+_a v čase $O(kn^2)$, kde n je počet výskytov literálov v DH^+_a a k je počet výrokových premenných v P.

Algoritmus 4.176 (Min-Model-DH_a^+)
vstup: *DH_a^+ – nožina defaultových pravidiel typu $\mathcal{DH}_a^+$, P – množina výrokových premenných z DH_a^+*
výstup: *minimálny model m_{min} množiny DH_a^+*

begin
  $p\_remain := ORDER(P, DH_a^+)$
  $m_{part} := \emptyset$
    (m_{part} je 3-interpretácia *)*
  loop
    if $ELEM(p\_remain) = \emptyset$ **then return**
    $p := HEAD(p\_remain); p\_remain := TAIL(p\_remain)$
    $set\_t :=$ **false**; $set_f :=$ **false**
    for $\forall d = \alpha \rightsquigarrow x \in DH_a^+$ *with* $x = p$ *or* $x = \neg p$ **do**
      blocked := **false**
      if m_{part} *splňuje* α **then**
        $m_{neg} := \{x | x =$ (**if** $p \in M_{part} \cup \alpha$ **then** p **else** $\neg p), \ p \in P\}$
        for $\forall d' = (\beta \cup \alpha) \rightsquigarrow \neg x \in DH_a^+$ **do**
          if m_{neg} *splňuje* β **then** *blocked* := **true**
        end for
        if not *blocked* **then**
          if $x = p$ **then** $set\_t :=$ **true else** $set\_f :=$ **true**
    end for
    if not set_t **then** $m_{part} := m_{part} \cup \{\neg p\}$
    else if not set_f **then** $m_{part} := m_{part} \cup \{p\}$
  end loop
  $m_{min} := \{x | x =($ **if** $p \in m_{part}$ **then** $x = p$ **else** $\neg p\}), p \in P\}$
end

Tabuľka 4.7: Algoritmus Min-Model-DH_a^+

[SKa 90] analyzovali veľmi jednoduché jazyky a v nich jediný problém – problém nájdenia nejakého minimálneho modelu; ich zistenia nie sú veľmi optimistické: $\mathcal{D}_s$, $\mathcal{DH}_s^+$, $\mathcal{D}_s^+$ sú *NP*-ťažké; iba $\mathcal{DH}_s$ a $\mathcal{DH}_{a\ s}^+$ sú vypočítateľné v polynomickom čase.

Ostáva ešte pripomenúť, aký dosah majú uvedené výsledky pre zložitostnú charakterizáciu výpočtov súvisiacich s extenziami defaultových teórií. Z časti 4.1 už vieme, že nie je priamočiara korešpondencia medzi najlepšími modelmi a extenziami defaultových teórií. Okrem toho, extenziu necharakterizuje jediný model, ale celá trieda modelov. Vo všeobecnosti má extenzia viac modelov. Vynára sa teda celkom prirodzená otázka, nakoľko uvedené výsledky charakterizujú zložitosť výpočtu extenzií defaultových teórií. V tomto smere dosiahli Selman s Kautzom tento výsledok:

Definícia 4.177 Pár blokujúcich sa defaultov je dvojica defaultov d, d' takých, že $cons(d) = \overline{cons(d')}$. ■

Tvrdenie 4.178 *Nech $DH^+ \in \mathcal{DH}$. Nech DH^+ neobsahuje dvojicu blokujúcich sa defaultov.*

Ak DH^+ vytvára reláciu preferencie, ktorá je čiastočným usporiadaním, potom m je najlepší model DH^+ práve vtedy, keď množina všetkých formúl splnených v m je extenziou nejakej defaultovej teórie priradenej k DH^+.

Samozrejme, vzniká otázka o priamej zložitostnej charakterizácii štandardných defaultových teórií, respektíve o identifikovaní takých jazykových obmedzení, ktoré umožňujú zvládnuteľný výpočet. Aj tomuto problému sa venujú Kautz so Selmanom. V práci [KSe 91] získali tiež prevažne pesimistické výsledky o zložitosti výpočtu základných problémov, súvisiacich s extenziami defaultových teórií, hoci sa zaoberali výpočtami vo veľmi jednoduchých prípadoch defaultových teórií.

4.3.5 Príčiny zložitosti

Zhrnutie:
Selman s Kautzom sa v [KSe 91] rozhodli preverovať rozšírený predpoklad, podľa ktorého príčinou zlých zložitostných charakteristík defaultových teórií sú náklady na testovanie konzistentnosti. Podarilo sa im však ukázať, že príčiny sú komplikovanejšie. Skúmali totiž limitované jazyky, v ktorých testy konzistentnosti sú triviálne, a napriek tomu zložitostné charakteristiky sú pesimistické.

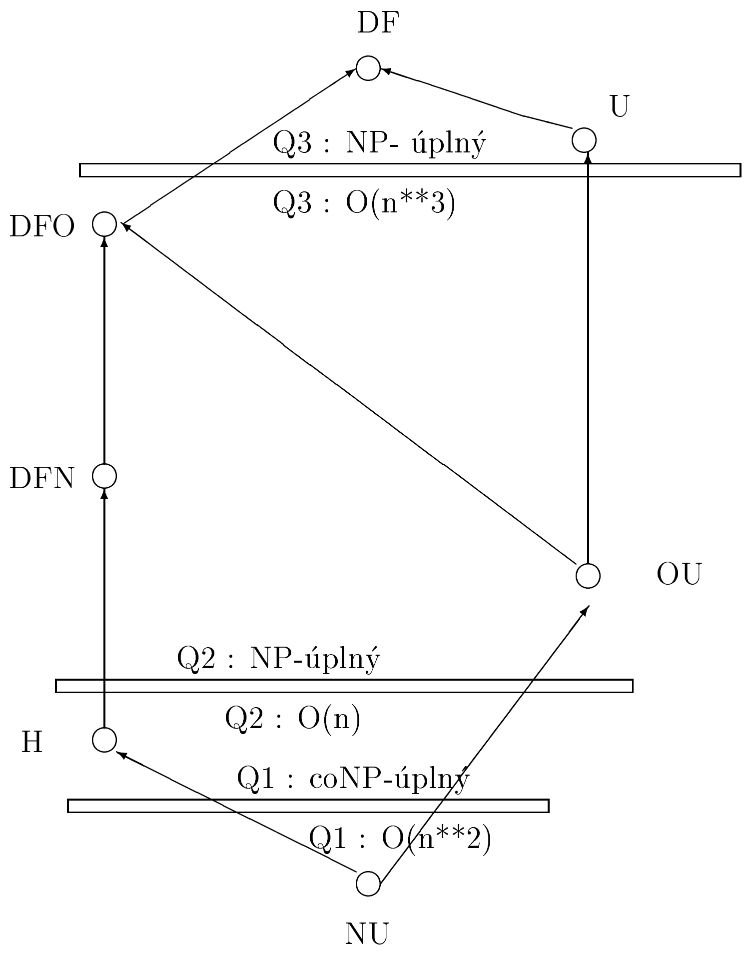
Popísali čiastočne usporiadanú množinu typov defaultových pravidiel (pozri obrázok 4.12). Formuly v týchto pravidlách nevyužívali plnú vyjadrovaciu silu výrokovologického jazyka. Predpokladalo sa ďalej, že E je množinou literálov. Pre defaultovú teóriu každého druhu z tejto hierarchie študovali problémy $Q1, Q2, Q3$ (pozri časť 4.3.1).

Pre najkomplikovanejší jazyk tejto hierarchie sú všetky tri problémy nezvládnuteľné. Pre najjednoduchší sú známe polynomické algoritmy na riešenie každého z problémov. Pre všetky ostatné typy jazykov aspoň jeden problém je nezvládnuteľný. Obrázok 4.12 dáva dobrú vizualizáciu, skratky typov defaultových pravidiel sú zavedené v definíciách 4.180 a 4.182.

Analýza týchto výsledkov ukázala príčiny veľkej výpočtovej zložitosti:

- *potrebu kontrolovať konflikty medzi defaultovými pravidlami (s prípadnou nutnosťou vrátiť výpočet do niektorého z predchádzajúcich stavov),*
- *exponenciálny počet extenzií.*

Prejdime k detailom. Definícia najzložitejšieho z opísaných typov jazykov:



Obrázok 4.12: Hierarchia jazykov a problémov pre defaultové teórie bez disjunkcie. Problém $Q1$: patrí daná formula do každej extenzie danej defaultovej teórie? Problém $Q2$: patrí daná formula do niektorej extenzie danej defaultovej teórie? Problém $Q3$: nájdenie extenzie.

Algoritmus 4.179
vstup: $T = (E, D)$, *defaultová teória bez disjunkcií*
výstup: *extenzia* $\mathcal{E}$ *(ak existuje)*

begin
nedeterministická voľba
    postupnosti $\langle d_1, \ldots, d_k \rangle$ *pravidiel z* D
    extenzie $\mathcal{E}$
$E_0 := E$
for $i = 1$ **to** k **do**
        if $pre(d_{i-1}) \subseteq E_{i-1} \wedge \neg \exists c \in just(d_i)\ \overline{c} \in \mathcal{E}$
        then $E_i := E_{i-1} \cup \{cons(d_i)\}$
if $E_n = \mathcal{E}$ **then** *úspech* **else** *neúspech*
(po úspešnom ukončení činnosti algoritmu platí:* $\neg \exists d \in D\ (pre(d) \subseteq E_n \wedge \neg \exists c \in just(d)\ (\overline{c} \in E_n) \wedge cons(d) \not\subseteq E_n)$ **)*
end

Tabuľka 4.8: Algoritmus konštruujúci extenziu DF-defaultovej teórie

Definícia 4.180 (DF) Defaultová teória je *DF* (bez disjunkcie, disjunction-free), ak obsahuje iba defaultové pravidlá tvaru $a_1 \wedge \cdots \wedge a_k : b_1 \wedge \cdots \wedge b_m \wedge c_1 \wedge \cdots \wedge c_n / b_1 \wedge \cdots \wedge b_m$, kde a_i, b_j, c_l sú literály. ■

Namiesto konjunkcie môžeme hovoriť o množine literálov. Dôsledkom rozhodnutia obmedziť sa v defaultových pravidlách na množiny resp. konjunkcie literálov je, že stačí testovať členstvo literálu v množine. Zámer vyhnúť sa disjunkciám je pochopiteľný už z intuitívneho hľadiska: disjunkcie sú signálom neúplnosti informácií a zdrojom výpočtových ťažkostí. Odvodzovanie z disjunkcií iniciuje prehľadávanie alternatívnych vetiev možných dôsledkov. Prednosti defaultových teórií bez disjunkcie dokladá aj nasledujúca lema.

Lema 4.181 ([SKa 90]) *Nech* $T = (E, D)$ *je defaultová teória bez disjunkcií. Potom* $\mathcal{E}$ *je jej extenziou práve vtedy, keď existuje konečná postupnosť pravidiel* $d_1, \ldots, d_n$ *a konečná postupnosť množín viet* $E_0, E_1, \ldots, E_n$ *takých, že* $E_0 = E$, $E_n = \mathcal{E}$ *a pre každé* $i > 0$ *platí*

$$E_i = E_{i-1} \cup \{cons(d_i)\},$$
$$pre(d_i) \subseteq E_{i-1},$$
$$\neg \exists c \in just(d_i)\ \overline{c} \in E_n,$$
$$\neg \exists d \in D\ (pre(d) \subseteq E_n \wedge \neg \exists c \in just(d)\ (\overline{c} \in E_n) \wedge cons(d) \not\subseteq E_n).$$

■

Uvedená lema umožňuje po konečnom počte krokov preveriť, či nejaká postupnosť defaultových pravidiel a množín viet dáva extenziu (pozri algoritmus 4.179, tabuľka 4.8). Konzistentnosť ani vyplývanie nemusíme preverovať. Stačí príslušnosť k množine a inklúzia. Samozrejme, ide o nedeterministický polynomický algoritmus: Prejdeme teraz k výpočtovej zložitosti problémov $Q1 - Q3$ pre defaultové teórie bez disjunkcie.

Definície ostatných typov jazykov spomínanej hierarchie:

Definícia 4.182 Budeme predpokladať, že máme DF defaultovú teóriu. Ak na jej tvar naložíme ďalšie obmedzujúce podmienky, dostávame:

U Defaultová teória je *unárna*, ak v každom jej defaultovom pravidle predpoklady pozostávajú z jediného atómu, dôsledky pozostávajú z jediného literálu, stráže môžu obsahovať konjunkciu atómu a literálu ($p : q/q$, $p : \neg q/\neg q$, $p : q \wedge \neg r/q$).

DFO Defaultová teória je *bez disjunkcie a usporiadaná* (disjunction-free ordered), ak je bez disjunkcie a neobsahuje literál l taký, že $l \ll l$. Usporiadanie $\ll$ na literáloch je definované v definícii 4.113.

OU Defaultová teória je *usporiadaná a unárna*, ak je unárna a súčasne usporiadaná.

DFN Defaultová teória je *bez disjunkcie a normálna*, ak jej konzekvent sa zhoduje s jej strážou (pravidlá sú tvaru $a_1 \wedge \cdots \wedge a_k : b_1 \wedge \cdots \wedge b_m/b_1 \wedge \cdots \wedge b_m$, kde a_i, b_j sú literály).

H Defaultová teória je *hornovská*, ak jej predpoklady vytvára konjunkcia atómov, stráž (a s ňou zhodný dôsledok) obsahuje jediný literál (pravidlá sú tvaru $p_1, \ldots p_n : l/l$, kde p_i sú atómy a l je literál).

NU Defaultová teória je *normálna a unárna*, ak je normálna a jej predpokad tvorí jediný atóm, jej stráže (a dôsledok) vytvára jediný (ten istý) literál (pravidlá sú tvaru $p : q/q$ a $p : \neg q/\neg q$).

■

Čiastočné usporiadanie na týchto typoch defaultových teórií je ilustrované na obrázku 4.12.

Pre problém Q1 existuje kvadratický algoritmus v prípade NU teórie. Pre všetky ostatné prípady (pre všetky teórie so zložitejšími defaultovými pravidlami) je problém *coNP*-úplný.

Pre problém Q2 existuje lineárny algoritmus pre H teórie (a samozrejme pre ich špeciálny prípad, NU). Pre všetky ostatné typy teórií je problém *NP*-úplný.

Pre Q3 existuje kubický algoritmus pre DFO teórie (a ich špeciálne prípady). Pre DF a U je aj tento problém *NP*-ťažký.

Začneme od problému $Q3$, od problému nájdenia nejakej extenzie (všetky tvrdenia a algoritmy pochádzajú z [SKa 90]).

Tvrdenie 4.183 *$Q3$ je NP-ťažký pre unárne teórie.*

Tvrdenie 4.184 *Existuje $\mathcal{O}(n^3)$ algoritmus pre $Q3$ v usporiadaných teóriách bez disjunkcie.*

Teda, hranica nezvládnuteľnosti pre problém $Q3$ leží v hierarchii na obrázku 4.12 pomerne vysoko. Vlastný algoritmus pre $Q3$ v usporiadaných teóriách bez disjunkcie (DFO) je v tabuľke 4.9. Skôr než prejdeme ku komentovaniu, potrebujeme malú prípravu.

Predpokladajme usporiadanú teóriu bez disjunkcií (E, D). Vďaka tomu, že sme v usporiadanej teórii, môžeme zadefinovať aj usporiadanie na defaultových pravidlách.

Definícia 4.185 $d_1 \prec d_2$ práve vtedy, keď $\exists b \in cons(d_1)\ \exists c \in just(d_2)\ b \ll \overline{c}$, kde $\overline{c}$ je komplement literálu c. ■

Ešte si zadefinujeme predikát *applicable*, ktorý využijeme na testovanie podmienky v algoritme 4.186, pozri tabuľku 4.9. Bude hovoriť o tom, kedy je defaultové pravidlo d aplikovateľné na množine viet E pri extenzii $\mathcal{E}$.

Definícia 4.187 Nech d je defaultové pravidlo, E a $\mathcal{E}$ sú množiny viet. Formula $applicable(d, E, \mathcal{E})$ je pravdivá práve vtedy, keď

$$
\begin{gathered}
pre(d) \subseteq E, \\
cons(d) \not\subseteq E, \\
\neg\exists c \in just(d)\ \overline{c} \in \mathcal{E}.
\end{gathered}
$$

■

V algoritme 4.186 využívame konštrukciu z lemy 4.181 (a algoritmu 4.179), nedeterminizmu sme sa zbavili vďaka usporiadaniu. Teraz prejdeme k problému $Q2$. Možno by sa niekomu zdalo, že preverenie jediného literálu je ľahšie ako výpočet celej extenzie.

Tvrdenie 4.188 *$Q2$ je pre usporiadané unárne defaultové teórie NP-úplný.*

Tvrdenie 4.189 *$Q2$ je pre normálne defaultové teórie bez disjunkcie NP-úplný.*

Algoritmus 4.186 (Q3)
vstup: *usporiadaná teória bez disjunkcií* (E, D)
výstup: *jej extenzia*

begin
usporiadať D *podľa* $\prec$, i*-te defaultové pravidlo v* D *je* d_i
$\mathcal{E} := E$
$i := 1$
while $i \leq |D|$ **do**
  if $applicable(d_i, \mathcal{E}, \mathcal{E})$
    then
      begin
      $\mathcal{E} := \mathcal{E} \cup \{cons(d_i)\}$
      $i := 1$
      end
    else $i := i + 1$
return $\mathcal{E}$ **end**

Tabuľka 4.9: Algoritmus pre riešenie $Q3$ v usporiadaných teóriách bez disjunkcie

Teda, preverenie, či jediný literál patrí do nejakej extenzie, nie je ľahšie ako výpočet celej extenzie. Defaultové usudzovanie totiž nie je lokálne. Predpokladá sa pri ňom rozhľad po celej extenzii. Dokonca nestačí spoznať jedinú extenziu: aby sme určili, či daný literál patrí do nejakej extenzie, v najhoršom prípade treba prehľadať všetky extenzie.

Problém $Q2$ možno efektívne počítať pre hornovské defaultové teórie. Všimnime si, že hornovské defaultové pravidlá s negatívnym dôsledkom nevedú k aplikácii žiadneho z ďaších hornovských pravidiel: tie totiž obsahujú iba pozitívne predpoklady. Teda, defaultový dôkaz, že atóm p patrí do nejakej extenzie využije iba hornovské defaulty s pozitívnym dôsledkom. Takéto pravidlá si môžeme reprezentovať implikáciami $\alpha \to p$.

V prípade, že potrebujeme ukázať, že negatívny literál l patrí do nejakej extenzie, situácia je jednoduchá: treba použiť iba jedno pravidlo s negatívnym dôsledkom l (ostatné použité pravidlá budú s pozitívnymi dôsledkami).

Tvrdenie 4.190 (Q2) *Predpokladajme, že* (E, D) *je hornovská defaultová teória,* p *je atóm,* E *je množina literálov a* l *je literál.*

Konštruujme (štandardnú) hornovskú teóriu H *takto:*

- $E \subseteq H$,
- *ak* $\alpha : p/p \in D$, $\neg p \neq l$ *a* $\neg p \notin E$, *potom* $\alpha \to p \in H$,

- *ak l je negatívny literál a $\alpha : l/l \in D$, potom $\alpha \to l \in H$.*

Potom platí: x je v nejakej extenzii teórie (E, D) práve vtedy, keď je dokázateľné z H.

Teda, problém zistiť, či literál patrí do nejakej extenzie, je transformovateľný na problém dokázateľnosti vo výrokovologickej hornovskej teórii. Dokázateľnosť je ekvivalentná s nesplniteľnosťou. Problém splniteľnosti je pre hornovské teórie rozhodnuteľný v lineárnom čase, pozri [DGa 84] a aj kapitolu 2, algoritmus 2.47.

Tvrdenie 4.191 *Existuje $\mathcal{O}(n)$ algoritmus pre $Q2$ na hornovských defaultových teóriách.*

Teraz prejdeme k problému $Q1$. Ide o členstvo nejakého literálu l v každej extenzii. Tento problém môžeme transformovať na problém, či existuje extenzia $\mathcal{E}$, do ktorej l nepatrí. Na základe takejto redukcie nasledujúce tvrdenia hovoria o *coNP*-úplnosti.

Tvrdenie 4.193 *$Q1$ je coNP-úplný pre usporiadané unárne teórie.*

Tvrdenie 4.194 *$Q1$ je coNP-úplný pre hornovské teórie.*

Druhé z tvrdení platí napriek tomu, že existuje lineárny algoritmus pre $Q2$ na H. V prípade problému $Q1$ však nemôžeme ignorovať pravidlá s negatívnymi konzekvenciami a strážami. Zaujímajú nás totiž aj tie extenzie, kde je blokovaná príslušnosť literálu l k danej extenzii.

Existuje polynomický algoritmus pre normálne unárne teórie, ktorý rieši problém $Q1$. Podľa [SLe 93] je to prvý korektný a úplný algoritmus tohto druhu (môžeme si problém formulovať aj ako problém ideálne skeptického usudzovania).

Tvrdenie 4.195 *Algoritmus 4.192 nájde všetky extenzie normálnej unárnej teórie v kvadratickom čase. (Teda, problém $Q1$ je $O(n^2)$ pre NU.)*

Najprv definície, využívané vo formulácii algoritmu 4.192, tabuľka 4.10.

Definícia 4.196 Nech p je atóm, L množina literálov, D množina normálnych unárnych defaultových pravidiel. Formula tvaru *fixed-pos(p,L,D)* platí práve vtedy, keď

$$: p/p \in D,$$
$$: \neg p/\neg p \notin D,$$
$$\forall (q : \neg p/\ \neg p \in D)\ \neg q \in L.$$

■

Algoritmus 4.192 (Q1)
vstup: *literál x; normálna unárna defaultová teória* $T = (E, D)$
výstup: áno, *ak každá extenzia T obsahuje x*

$D' := \{d \in D \mid \overline{cons(d)} \notin E\} \cup \{: y/y \mid y \in E\}$
if x *je pozitívny literál*
  then $NUAEP(x, D')$
  else
    begin
    nech p je nová výroková premenná
    $D'' := D' \cup \{pre(d) : p/p \mid d \in D' \wedge cons(d) = \{x\}\}$
      $\cup \{pre(d) : \neg p/\neg p \mid d \in D' \wedge cons(d) = \{\overline{x}\}\}$
    $NUAEP(p, D'')$
    end

procedure $NUAEP(p_k, D)$
vstup: pozitívny literál p_k, D *– množina normálnych unárnych defaultov, obsahujúca premenné* $p_1, \ldots, p_n$
výstup: **áno**, *ak každá extenzia teórie* $(\emptyset, D)$ *obsahuje* p_k

$L := \{p_1, p_2, \ldots, \neg p_k, \ldots, p_n\}$
while *pos-consistent(L,D)* **do**
  if $\exists p, \neg q \in L$ ($\neg grounded(p, L, D) \vee$ *neg-inconsistent*$(p, \neg q, L, D)$)
    then $L := (L \setminus \{p\}) \cup \{\neg p\}$
    else return *„nie"*
return *„áno"*
end

Tabuľka 4.10: Algoritmus na riešenie problému $Q1$ pre NU defaultové teórie

Formula *fixed-pos(p,L,D)* je teda pravdivá, ak vstupný atóm p možno odvodiť pomocou niektorého bezpredpokladového normálneho unárneho pravidla z D, ak nemožno takto odvodiť jeho popretie a ak ostatné pravidlá, ktoré by toto popretie mohli odvodiť, sú blokované množinou viet L.

Definícia 4.197 Nech L je množina literálov a D množina normálnych unárnych defaultových pravidiel.

Potom *pos-consistent(L,D)* platí práve vtedy, keď pre každý atóm p splnená podmienka fixed-pos(p, L, D) stačí na to, že $p \in L$. ■

Definícia 4.198 Nech p je atóm, L množina literálov, D množina unárnych normálnych defaultových pravidiel.

Potom *grounded(p, L, D)* platí práve vtedy, keď existuje postupnosť $\langle q_0, q_1, \ldots, q_k \rangle$, kde $q_k = p$, taká, že

$$q_j \in L, 0 \leq j \leq k,$$
$$: q_0/q_0 \in D,$$
$$q_{j-1} : q_j/q_j \in D, 1 \leq j \leq k.$$

■

Atóm p je teda grounded (uzemnený) v L a D práve vtedy, keď ho možno získať sériou defaultových pravidiel, pričom každý literál z týchto pravidiel patrí do L.

Definícia 4.199 Nech p a $\neg q$ sú literály, L množina viet a D množina defaultových pravidiel.

Potom *neg-inconsistent*(p, $\neg q$, L, D) práve vtedy, keď

$$p, \neg q \in L,$$
$$p : q/q \in D,$$
$$: \neg q/\neg q \notin D,$$
$$\forall (r : \neg q/\neg q \in D)\ \neg r \in L.$$

■

Ak dané p umožňuje defaultovo odvodiť q, ktoré je komplementárne k danému $\neg q$, potom L a D znemožňujú odvodiť $\neg q$.

Závery: Príčinami zložitosti výpočtov v defaultových teóriách sú okrem logického pozadia (testy konzistentnosti a vyplývania) predovšetkým:

- detekovanie nekoherentných cyklov v pravidlách (preto sú zložitostné charakteristiky lepšie pre usporiadané defaultové teórie)
- a exponenciálne množstvo rôznych extenzií, ktoré vedie k nedeterministickému výpočtu.

Medzi pozitívne závery patrí zistenie dôležitosti usporiadaných a hornovských defaultových teórií. Negatívne zložitostné výsledky sú cenné aj tým, že dobre charakterizujú, ako je usudzovanie tohto typu ťažké. Dôsledkom získaných zistení je, že v praxi môže byť veľmi užitočné používanie takých nástrojov (jazykov), ktoré sú nejako limitované, ale dovoľujú v rozumnom zmysle slova aproximovať bohatšie jazyky.

4.3.6 Výpočty v hierarchických sieťach

Zhrnutie:

I keď sa možno zdá, že výpočty v hierarchických sieťach sú jednoduché, aj

	skeptický prístup	dôverčivý prístup
zhora	NP-ťažký	NP-ťažký
zdola	P	P

Tabuľka 4.11: Zložitosť rôznych foriem dedičnosti

v tejto oblasti sa naráža na NP-úplné a ťažšie problémy. Uvedieme niekoľko základných výsledkov a aj niekoľko polynomických algoritmov. Celkový prehľad dáva tabuľka 4.11.

Výsledky Kautza a Selmana z [KSe 91] do istej miery charakterizujú aj dedenie v hierarchických sieťach. Unárne usporiadané defaultové teórie sú postačujúce na jeho špecifikáciu. Ak si zrekapitulujeme: Pre usporiadané unárne teórie je problém $Q3$ zvládnuteľný (existuje kubický algoritmus pre usporiadané defaultové teórie bez disjunkcie a pre všetky ich špeciálne prípady). Ale problém $Q2$ je už *NP*-úplný pre tieto teórie a problém $Q1$ je pre ne *coNP*-úplný.

Samozrejme, riešenie problému $Q3$ nie je presne to, čo potrebujeme, keď sa pýtame na dedičnosť v hierarchickej sieti – nestačí nám nejaká náhodne nájdená extenzia. Potrebujeme takú extenziu, v ktorej sa nachádza daný literál (vyjadrujúci to, že nejaký objekt a má vlastnosť p). Ak uvažujeme dôverčivo, musíme riešiť problém $Q2$, ak skepticky $Q1$. Okrem toho, potrebujeme ešte zvládnuť preferovanie špecifickejšej informácie.

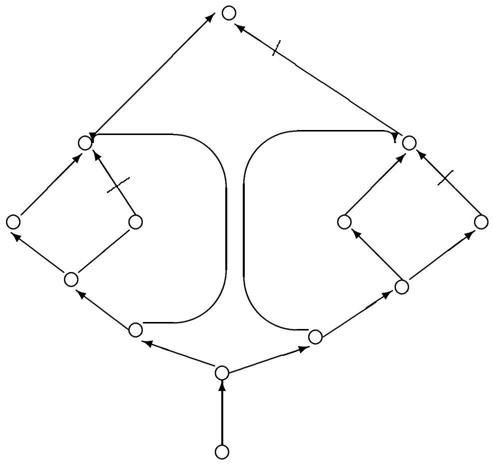
Prvý polynomický, korektný a úplný algoritmus pre usudzovanie v hierarchických sieťach navrhla Stein [Ste 92]. Jeho zložitosť je $\mathcal{O}(n^5)$. Je to algoritmus vychádzajúci z pozície zdržanlivého skepticizmu: rešpektuje špecifickejšiu informáciu. Výstupom je nejaká extenzia (v tom zmysle slova ide o dôverčivé usudzovanie). Vo všeobecnosti, môže byť viac dôverčivých extenzií tej istej hierarchickej siete, rešpektujúcich špecifickejšiu informáciu. Túto možnosť ilustruje obrázok 4.13. Aby sme boli v súlade s teoretickým rámcom z časti 4.2.4, prejdeme na výsledky Selmana a Levesqua. Spomínaný rámec pochádza z [SLe 93].

Selman a Levesque [SLe 93] podali ucelený pohľad na zložitosť výpočtov v priamych teóriách dedičnosti. Tento pohľad prehľadne vyjadruje tabuľka 4.11 a tvrdenie 4.201.

Definícia 4.200 Cesta $\sigma = (x_1, \ldots, x_{n-1}, x_n)$ je konkatenáciou nejakých ciest z Φ *zdola nahor*, ak hrana (x_{n-1}, x_n) je z Φ a cesta $(x_1, \ldots, x_{n-1})$ je tiež z Φ.

Cesta $\sigma = (x_1, x_2, \ldots, x_{n-1}, x_n)$ je konkatenáciou nejakých ciest z Φ *zhora nadol*, ak $(x_2, \ldots, x_n)$ je z Φ a $(x_1, \ldots, x_{n-1})$ je tiež z Φ. ■

Tvrdenie 4.201 *Výpočtová zložitosť problému dedičnosti pre rôzne voľby medzi konkatenáciou zhora nadol alebo zdola nahor, skeptickým a dôverčivým*



Obrázok 4.13: Sieť s dvoma dôverčivými extenziami rešpektujúcimi špecifickejšiu informáciu

prístupom je zosumarizovaná v tabuľke 4.11.

Tabuľka ukazuje, že výpočet zdola nahor (bez ohľadu na to, či postoj je skeptický alebo dôverčivý) je zvládnuteľný, kým výpočet zhora nadol nezvládnuteľný.

Zoznámime sa ešte s polynomickými algoritmami Selmana a Levesqua [SLe 93]. Najprv však zadefinujeme potrebné pojmy.

Predpokladáme, že máme danú hierarchickú sieť Γ. K nej postupne konštruujeme množinu dôsledkov C tak, aby boli podopreté nejakou extenziou siete Γ (aby reprezentovali dedičnosť v tejto sieti). Množinu dôsledkov C postupne rozširujeme dovtedy, kým nebude obsahovať všetky cesty, ktoré umožňujú korektné dedenie. Predstavme si, že uvažujeme dva uzly x a z. V sieti buď platí, že x sú z alebo nie sú z. Potrebujeme rozšíriť množinu C buď o $x \to z$ alebo o $x \not\to z$ (Pripomeňme, že v [SLe 93] sa na dôsledky nazerá ako na virtuálne hrany.) Samozrejme, toto rozšírenie je možné vtedy, keď nebude preválžené dôsledkom, opierajúcim sa o špecifickejšiu informáciu.

Definícia 4.202 Nech x, y, z sú uzly z Γ, $x \to y \in C$ a buď $y \to z \in \Gamma$ alebo $y \not\to z \in \Gamma$.

Dôsledok $x \to z$ (alebo $x \not\to z$) je preválžený (pri množine dôsledkov C, súčasne aj pri hierarchickej sieti Γ), ak neexistuje v Γ uzol u, z ktorého vedie cesta (v Γ) do y, do ktorého vedie cesta (v C) z x a Γ obsahuje aj hranu $u \not\to z$ (alebo $u \to z$). Značenie: $\mathit{preempted}(x, y, z, C, \Gamma)$. ■

Definícia 4.203 Nech C je množina dôsledkov, Γ hierarchická sieť, x, y, z sú uzly.

Potom $extendable(x, y, z, C, \Gamma)$ platí práve vtedy, keď $x \to z \notin C, x \not\to z \notin C, x \to y \in C$ a platí buď $y \to z \in \Gamma$ alebo $y \not\to z \in \Gamma$. ■

Teda množina dôsledkov C (siete Γ) je rozšíriteľná, ak obsahuje dôsledok $x \to y$ a zodpovedajúca hierarchická sieť obsahuje hranu z y do z, ale C neobsahuje nijaký dôsledok, ktorý sa týka x a z. Uzol y môžeme volať prostredníkom (sprostredkujúcim uzlom) rozšírenia.

Algoritmus 4.205 pre „dôverčivé dedenie“ zdola nahor je v tabuľke 4.13 a algoritmus 4.204 pre „skeptické dedenie“ zdola nahor v tabuľke 4.12.

Algoritmus 4.204 (Skeptický algoritmus zdola nahor)
vstup: *acyklická hierarchická sieť* Γ *s* n *uzlami*
výstup: *množina skeptických dôsledkov* C

begin
usporiadať uzly z Γ *tak, že uzol* x_i *je* i*-ty v poradí*
$C := \Gamma$
for $i = 1, \ldots, n-1$ **do**
    for $j = i+1, \ldots, n$ **do**
        if $\exists x \neg \exists y\ (extendable(x_i, x, x_j, C, \Gamma) \wedge \neg preempted(x_i, x, x_j, C, \Gamma) \wedge$
            $extendable(x_i, y, x_j, C, \Gamma) \wedge$ *hrana* (y, x_j) *je opačnej polarity ako*
            hrana $(x, x_j) \wedge \neg preempted(x_i, y, x_j, C, \Gamma)$
        then $C := C \cup \{x_i \to x_j$ *if* $x \to x_j \in \Gamma$
        else $x_i \not\to x_j$
    end for
end for
end

Tabuľka 4.12: Skeptický algoritmus zdola nahor

4.3.7 Komentáre

Vysoká výpočtová zložitosť defaultových teórií je predmetom mnohých ďalších výskumov. Veľmi stručne budeme informovať o niektorých z nich.

Cholewinski [Cho 94, Cho 95] ukázal, že existuje podtrieda defaultových teórií, v ktorej výpočet extenzií a rozhodovanie o tom, či nejaký literál patrí do extenzie možno urýchliť. Technika, ktorá sa používa, je stratifikácia.

Tu iba veľmi stručne zhrnieme základy prístupu. Defaultové pravidlá sa rozdelia do viacerých skupín (vrstiev). Extenzie sa počítajú pre každú vrstvu zvlášť a potom sa pospájajú. Výpočet je efektívnejší, potvrdzujú to aj experimenty opísané v [CMT 95].

Algoritmus 4.205 (Dôverčivý algoritmus zdola nahor)
vstup: *acyklická sieť* Γ *s* n *uzlami*
výstup: *množina dôverčivých dôsledkov* C

begin
usporiadať uzly z Γ *tak, že uzol* x_i *je* i*-ty v poradí*
$C := \{e : e$ *je hrana v* $\Gamma\}$ *(* všetky hrany priamo zaznamenané v* Γ *sú dôsledkami siete* Γ **)*
for $i = 1, \ldots, n-1$ **do**
  for $j = i+1, \ldots, n$ **do**
    if $\exists x \neg \exists x'$ $(extendable(x_i, x, x_j, C, \Gamma) \wedge x' \prec x \wedge extendable(x_i, x', x_j, C, \Gamma))$
      *(** $\prec$ *je čiastočné usporiadanie definované grafom* Γ **)*
      (podmienka vyjadruje, že* C *možno rozšíriť o nejaký dôsledok, týkajúci sa uzlov* x_i, x_j*, pričom za prostredníka toho rozšírenia sa berie minimálny sprostredkujúci uzol vzhľadom na* $\prec$ **)*
    then $C := C \cup \{x_i \rightarrow x_j$ *if* $x \rightarrow x_j \in \Gamma$ **else** $x_i \not\rightarrow x_j\}$
  end for
end for
end

Tabuľka 4.13: Dôverčivý algoritmus zdola nahor

Roundsovi a Zhangovi [RoZ 97d], sa podarilo ukázať, že zmena pohľadu na defaultové usudzovanie vedie k zlepšeniu zložitostných charakteristík pre normálne defaultové teórie. Tento výsledok je založený na originálnej konštrukcii defaultového usudzovania, vychádzajúcej zo Scottovej teórie domén.

Veľmi zaujímavý smer výskumu reprezentuje práca [CDS 95]. Autori si kladú otázku, či nepríjemné zložitostné vlastnosti defaultového usudzovania nutne treba hodnotiť negatívne.

Kľúčová technika, ktorú používajú, je kompilácia jedného problému do iného problému. Hlavný vtip je v tom, že nejaké off-line výpočty môžeme urobiť vtedy, keď nie sú nijaké zvláštne nároky na rýchlosť výpočtu. Potom on-line riešime predpripravený problém. O čo ide, si predstavíme na príklade.

Príklad 4.206 Majme problém $(T \models \alpha, T, \alpha)$. Prvá zložka tejto trojice obsahuje špecifikáciu problému, v tomto prípade ide o to, či výrokovologický literál α vyplýva z formuly T, ktorá je v konjunktívnej normálnej forme. Druhá zložka, fixná časť problému, je daná formula T. Posledná zložka problému je variabilná – variabilita spočíva v tom, že daný problém môže mať rôzne inštancie. V našej inštancii sa pýtame, či α vyplýva z T.

Predstavme si, že sa nám podarí vytvoriť tabuľku o dvoch stĺpcoch. V prvom stĺpci je nejaký literál, v druhom indikátor toho, či daný literál vyplýva z T alebo nie. Pôvodný problém sme transformovali na nový problém – vyhľadávanie v tabuľke.

Takáto tabuľka má polynomickú veľkosť vzhľadom na počet atómov v T a vyhľadávanie v tabuľke je zvládnuteľné. Preto je pôvodný problém efektívne kompilovateľný na nový problém.

Samozrejme, nepožaduje sa, aby vlastná kompilácia pôvodného problému na nový problém prebehla v polynomickom čase. Už sme uviedli predstavu, že kompilácia prebieha off-line (vtedy, keď nie sú nijaké zvláštne nároky na dobu výpočtu). ■

Všimnime si teraz problém skeptického usudzovania ($T \vdash_{SK} x, T, x$), kde T je defaultová teória, pričom defaultové pravidlá sú bez predpokladov, normálne a unárne, x je nejaká formula. Ak index SK nahradíme indexom CR, budeme hovoriť o probléme dôverčivého usudzovania. Platí:

- ak x je konjunkcia pozitívnych literálov alebo disjunkcia negatívnych literálov, problém dôverčivého usudzovania nie je kompilovateľný (teda neexistuje transformácia na iný problém, reprezentovaný polynomicky väčšou údajovou štruktúrou a vypočítateľný v polynomickom čase),
- ak x je disjunkcia pozitívnych literálov, problém dôverčivého usudzovania je kompilovateľný,
- ak x je disjunkcia pozitívnych literálov alebo disjunkcia negatívnych literálov, problém skeptického usudzovania nie je kompilovateľný,
- ak x je konjunkcia literálov, problém skeptického usudzovania je kompilovateľný.

Tieto výsledky možno interpretovať takto:

1. prípad nekompilovateľnosti – ide o extrémnu kompaktnosť reprezentácie v nemonotónnom formalizme: nemonotónna reprezentácia znalostí môže byť natoľko kompaktná, že ju treba zaplatiť výpočtovou nezvládnuteľnosťou; takúto reprezentáciu dokonca nemožno ani prekompilovať na polynomicky rozsiahlejšie údajové štruktúry, ktoré by dovolili výpočet v polynomickom čase,
2. prípad kompilovateľnosti – existujú nemonotónne reprezentácie, ktoré nie sú výpočtovo zvládnuteľné, dajú sa však prekompilovať na polynomicky rozsiahlejšie údajové štruktúry, ktoré umožňujú výpočet v polynomickom čase (samozrejme, nepredpokladá sa, že by kompilácia bola v polynomickom čase); v tomto prípade nemonotónna reprezentácia poskytuje špecifikáciu problému, ktorý možno po vhodnej kompilácii počítať efektívne.

Záverom: problém reprezentácie znalostí (ktorý sme si vymedzili v kapitole 1) a tvorba výpočtových modelov inteligencie naráža na obmedzenia, súvisiace s výpočtovou zložitosťou. Okrem už opakovanej zásady, že je rozumné vyvažovať vyjadrovaciu silu a výpočtovú efektívnosť reprezentačných formalizmov, sa ukazuje aj ďalší pohľad. Na základe spomínanej zásady, v prípadoch, keď stačí chudobnejší jazyk (napríklad hornovské defaultové teórie), treba ho zvoliť a zabezpečí nám výpočet v polynomickom čase. Na druhej strane, ak môžeme oddeliť off-line spracovanie od on-line výpočtu a bohatší jazyk nám umožní kompaktnejšiu reprezentáciu, je účelné off-line prekompilovať reprezentáciu do polynomicky rozsiahlejšieho priestoru, ak sa tým umožní zvládnuteľný výpočet.

Navyše, môže sa stať, že znalosti reprezentované v bohatom jazyku dávajú tak kompaktnú reprezentáciu, že nie je možné skompilovať ju do rozumného prostredia, v ktorom je možný rozumný výpočet. Vtedy treba hľadať spôsoby výpočtu, ktoré zľavia z prísnych nárokov na korektnosť a/alebo úplnosť.

V tejto časti sme si priblížili niektoré implementácie defaultového usudzovania a dedenia v hierarchických sieťach. Potom sme sa oboznámili so základnými algoritmami pre túto oblasť a zložitostnými charakterizáciami. Všimli sme si niektoré triedy limitovaných jazykov, umožňujúcich zvládnuteľný výpočet.

Kapitola 5

Generalizácie a vysvetlenia

V tejto kapitole máme na programe ďalšie dva typy tradične študovaného hypotetického usudzovania, indukciu a abdukciu.

Predmetom nášho záujmu bude schopnosť odvodzovať zovšeobecnenia z pozorovaní a schopnosť vysvetliť nejaké pozorovania na základe prijatých teórií.

Ľudská inteligencia umožňuje nielen spoľahlivo odhadovať v dobre známom prostredí, ale aj pomerne rýchlo sa adaptovať v neznámom prostredí. Základnými črtami tzv. zdravého sedliackeho rozumu je práve spoľahlivá orientácia v dobre známom prostredí i úspešná adaptácia na neznáme prostredie. Indukcia a abdukcia umožňujú práve adaptáciu na neznáme prostredie. Indukcia rozširuje naše poznanie so všeobecnejším dosahom. Abdukcia umožňuje začleniť naše pozorovania do rozsiahlejšieho systému poznatkov (týmto začlenením ich dokáže vysvetliť).

Pripomeňme si ešte, že naším hlavným cieľom v kapitolách 4 a 5 je ilustrovať a rozšíriť všeobecné zistenia o hypotetickom usudzovaní na niektorých dobre študovaných príkladoch. Príklady z predchádzajúcej kapitoly sa týkali skôr orientácie v dobre známom prostredí, príklady z tejto kapitoly skôr adaptácie na neznáme prostredie. Treba povedať, že oba aspekty sa prelínajú a že naším cieľom nie je ostré odlíšenie v tomto smere. V tejto kapitole ide iba o to, zoznámiť sa s prístupmi k štúdiu indukcie a abdukcie.

5.1 Indukcia

Postavíme si problém, akými cestami možno z nejakých pozorovaní odvodiť užitočné zovšeobecnenia. Začneme od zaujímavého príkladu. Na jeho základe budeme charakterizovať, o čo vlastne pri indukcii ide. Potom načrtneme spôsob, ako možno získať induktívne zovšeobecnenia výpočtom. Prejdeme opäť k databázovej oblasti – krátkou informáciou o dolovaní znalostí z ohromných objemov databázových viet. Ďalej sa budeme venovať induktívnemu logickému programovaniu – snaženiu indukciu špecifikovať a implementovať nástrojmi logického programovania. Záverečnou témou tejto časti bude indukcia, založená na nemonotónnej sémantike.

Príklad 5.1 ([GeN 87]) Predstavme si takúto situáciu: Hráč ťahá z kôpky karty. Odmenu dostal, keď vytiahol *4 listovú*, *7 listovú* a *2 pikovú*. Po vytiahnutí *5 srdcovej* a *J pikového* odmenu nedostal. Úlohou hráča je odvodiť, za aké karty sa dáva odmena.

Inými slovami, hráč indukuje. Na základe niekoľkých pozorovaní (pozitívnych alebo negatívnych) zovšeobecňuje, pokúša sa sformulovať všeobecné pravidlo. ■

O čo vlastne ide pri zovšeobecňovaní z pozorovaní? Skúsime to špecifikovať sémantickými prostriedkami.

V našom príklade sa vyskytujú odpozorované dáta (označme si ich Δ). Pre indukciu je príznačné, že medzi pozorovaniami sa spravidla rozlišujú pozitívne príklady Δ^+ (v našom príklade to boli karty, za ktoré hráč dostal odmenu) a negatívne príklady Δ^- (v našom príklade to boli karty, za ktoré hráč odmenu nedostal). Z pozorovaní chceme odvodiť nejaké zovšeobecnenie, nejakú hypotézu, resp. množinu hypotéz, označme si ju Φ. Môžeme sa pritom opierať o nejaké poznanie, ktorým disponujeme, o nejaké používané pojmy. Nazvime si to *teória* (background theory) a označme ako Γ.

Uvedieme kostru sémantickej špecifikácie indukcie. V základnom, idealizovanom prípade medzi Δ, Γ, Φ platia tieto vzťahy:

Definícia 5.2 Nech Δ, Γ, Φ sú množiny viet nejakého jazyka J. Nech sú ďalej splnené tieto podmienky:

- $\Gamma \not\models \Delta$,
- $\Gamma \cup \Delta$ je konzistentná,
- $\Gamma \cup \Phi \models \Delta$,
- $\Gamma \cup \Phi$ je konzistentná.

Hovoríme, že Φ *induktívne* vyplýva z Δ (s Γ na pozadí), ak Γ, Δ, Φ spĺňajú uvedené podmienky. Môžeme použiť značenie $\Phi \subseteq Cn_{ind}(\Delta \cup \Gamma)$. ■

Poznámka 5.3 Ak by neplatila podmienka $\Gamma \not\models \Delta$, nebolo by potrebné zovšeobecňovať, získané pozorovania by boli uspokojujúco vysvetlené poznatkami, ktorými disponujeme. Podmienka konzistentnosti $\Gamma \cup \Delta$ znamená, že získané pozorovania nie sú v rozpore s prijatou teóriou. Podmienka $\Gamma \cup \{\Phi\} \models \Delta$ chápe vysvetlenie pozorovaní ako ich deduktívne odvodenie z teórie a hypotéz. Konzistentnosť $\Gamma \cup \{\Phi\}$ vyžaduje, aby vysvetlenie nebolo triviálne, aby nebolo založené na tom, že z nekonzistentnej množiny viet vyplýva čokoľvek.

Nepredpokladáme $\Phi = Cn_{ind}(\Delta \cup \Gamma)$, pretože z daného Δ a Γ možno odvodiť viac zovšeobecnení. Máme ďalší príklad alternatívnosti hypotetického usudzovania. Na rozdiel od extenzií (defaultové teórie neposkytujú kritérium výberu „lepších“ extenzií) sa v teórii indukcie dá uvažovať o lepších či horších zovšeobecneniach.

Samozrejme, indukcia nie je korektná. Ak máme $\phi \in Cn_{ind}(\Delta \cup \Gamma)$, možno nájsť model $\Gamma \cup \Delta$ taký, že ϕ v ňom nie je splnené. Podobne sa dá preveriť nemonotónnosť indukcie. ■

Videli sme, že podľa tohto poňatia je indukcia v istom zmysle slova inverzná k dedukcii. Indukovať znamená nájsť hypotézy, z ktorých pomocou poznania v pozadí vysvetlíme (vydedukujeme) pozorovania.

Teraz sa vrátime k nášmu príkladu 5.1, na jeho materiáli si konkretizujeme predstavu o množinách Δ, Γ a Φ.

Príklad 5.4 Množinu Δ tvoria pozitívne a negatívne príklady predikátu *odmena*:
$\Delta^+ = \{$

$$\begin{array}{rcl} odmena(karta(4,l)) & \leftarrow & , \\ odmena(karta(7,l)) & \leftarrow & , \\ odmena(karta(2,p)) & \leftarrow\} & , \end{array}$$

$\Delta^- = \{$

$$\begin{array}{rcl} \neg odmena(karta(5,s)) & \leftarrow & , \\ \neg odmena(karta(j,p)) & \leftarrow\} & . \end{array}$$

Mohli by sme používať aj $\leftarrow odmena(karta(5,s))$ a $\leftarrow odmena(karta(j,p))$. Ľahko sa dá ukázať, že ide o ekvivalentné formuly (z disjunktívneho vyjadrenia kláuz). Negatívne hlavy však nemožno používať v logických programoch. Preto neskôr budeme negatívne príklady formulovať ako integritné obmedzenia. Má to aj rozumný intuitívny zmysel: zovšeobecnenie sa nesmie dostať do rozporu s negatívnymi príkladmi.

Teória Γ pozostáva z definície binárneho predikátu $\prec$:

$$2 \prec 3 \prec \cdots \prec 10 \prec j \prec d \prec k \prec e$$

a z nasledujúcich definícií:

$$\begin{aligned}
num(karta(N,Z)) &\leftarrow N \preceq 10,\\
fig\acute{u}ra(karta(N,Z)) &\leftarrow N \succ 10,\\
\check{c}ierna(karta(N,Z)) &\leftarrow Z = l \vee Z = p,\\
\check{c}erven\acute{a}(karta(N,Z)) &\leftarrow Z = s \vee Z = k,\\
X \preceq Y &\leftarrow X \prec Y \vee X = Y,\\
X \prec Z &\leftarrow X \prec Y \wedge Y \prec Z.
\end{aligned}$$

Špecifikáciu teórie v pozadí možno rozširovať. Napríklad, mohli by sme pridať aj predpoklad tranzitívnosti $\preceq$ a axiómy identity.

Úlohu nájsť (induktívne) zovšeobecnenie z nášho príkladu možno chápať aj ako požiadavku definovať predikát *odmena*. Sémantickej definícii indukcie (definícia 5.2) vyhovuje pri daných Δ a Γ napríklad hypotéza:

$$odmena(X) \leftarrow num(X) \wedge \check{c}ierna(X).$$

Samozrejme, takýchto hypotéz je viac. Okrem krajných možností

$$\begin{aligned}
odmena(karta(X,Y)) &\leftarrow (X = 7 \wedge Y = l) \vee (X = 2 \wedge Y = p) \vee\\
&\qquad \vee (X = 4 \wedge Y = l),\\
odmena(karta(X,Y)) &\leftarrow karta(X,Y) \neq karta(5,s) \wedge\\
&\qquad \wedge\, karta(X,Y) \neq karta(j,p).
\end{aligned}$$

to môžu byť (popri mnohých iných) aj:

$$\begin{aligned}
odmena(karta(X,Y)) &\leftarrow (num(X) \wedge \check{c}ierna(Y)) \vee\\
&\qquad \vee (fig\acute{u}ra(X) \wedge \check{c}erven\acute{a}(Y)),\\
odmena(karta(X,Y)) &\leftarrow Y = l \vee (X \neq j \wedge Y = p).
\end{aligned}$$

Videli sme, že môže existovať viac kandidátov na induktívne zovšeobecnenie. Podstatnou otázkou teda je, ako voliť stratégiu rozumného zovšeobecňovania (indukovania). ■

Naším cieľom je teraz trochu uregulovať viacznačnosť, spojenú s induktívnym zovšeobecňovaním (z principiálnych dôvodov ju však nemôžeme vylúčiť). Neskôr sa budeme venovať rozlišovaniu viac a menej všeobecných zovšeobecnení a prijmeme rozhodnutie, že rozumné je prijímať tie najmenej zaväzujúce, teda najslabšie zovšeobecnenia. Na druhej strane, tak slabé „zovšeobecnenia“ ako formula

$$\begin{aligned}
odmena(karta(X,Y)) &\leftarrow (X = 7 \wedge Y = l) \;\vee\; (X = 2 \wedge Y = p)\\
&\qquad \vee (X = 4 \wedge Y = l)
\end{aligned}$$

teória	evidencia				
	pozitívna			negatívna	
	1	2	3	1	2
num	Y	Y	Y	Y	N
figúra	N	N	N	N	Y
čierna	Y	Y	Y	N	Y
červená	N	N	N	Y	N

Tabuľka 5.1: Schématické zaznamenanie príkladu 5.1. Stĺpce zaznamenávajú pozitívne a negatívne príklady. Y v nejakom riadku pre príklad v stĺpci i znamená, že daný príklad má vlastnosť vyjadrenú predikátom na začiatku toho riadku. N znamená, že nemá.

z príkladu 5.4, ktorá iba preformulovávala pozorovania, nebudeme chcieť považovať za zovšeobecnenia.

Prv než prejdeme k detailnejšej analýze, musíme si všimnúť ešte iný potrebný typ rozhodovania. Množstvo vygenerovaných kandidátov na rozumné hypotézy zredukujeme, keď limitujeme jazyk, v ktorom ich môžeme formulovať. Toto obmedzovanie je užitočné pre výpočet zovšeobecnení. Teda – skôr, než by sme rozmýšľali o návrhu na zúženie množiny kandidátov na rozumné zovšeobecnenie, limitujme jazyk. Zrejme, čím bude tento jazyk bohatší, tým viac možností by algoritmus generujúci zovšeobecnenia musel prehľadávať. Opäť sme pri príklade, ukazujúcom dôležitosť vyvažovania vyjadrovacej sily a výpočtovej efektívnosti reprezentácie. Keď budeme schopní lepšie počítať, dokážeme menej vyjadriť.

Príklad 5.5 Skúsme teda formulovať nejaké užitočné obmedzenia na jazyk, v ktorom sa formulujú možné hypotézy v našom príklade.

Veľmi jednoduchý jazyk (pre potreby príkladu 5.1) narába iba s predikátmi $num, figúra, čierna, červená$. Navyše, náš preferovaný kandidát na zovšeobecnenie obsahuje spomedzi logických spojok iba konjunkciu a implikáciu. Ak sa pri tvorbe hypotéz rozhodneme ignorovať negáciu a disjunkciu, stačí nám prehľadávať redukovaný priestor hypotéz.

Všimnime si tabuľku 5.1. Riadky zodpovedajú predikátom teórie (*num, figúra, čierna, červená*). Tabuľka ilustruje, ako obmedzovaním jazyka zabezpečíme pomerne jednoduchú špecifikáciu (a nájdenie) zovšeobecnenia. Stĺpce sú rozdelené do dvoch blokov. Prvý zodpovedá pozitívnym príkladom z Δ^+, teda tým, ktoré „získali odmenu“. Obsahuje 3 stĺpce. Druhý blok (2 stĺpce) zodpovedá negatívnym príkladom z Δ^-. V riadku x a stĺpci i je Y vtedy, keď i-te pozorovanie je inštanciou predikátu zo zodpovedajúceho riadku x (napríklad $karta(2, l)$ je inštanciou predikátu num, $čierna$, nie je však inštanciou predikátov $figúra$ alebo $červená$).

Keď máme v niektorom riadku v pozitívnej časti tabuľky samé Y, môžeme zodpovedajúci predikát považovať za relevantný pre vygenerovanú hypotézu. Teda, našli sme dva relevantné predikáty, num a $\check{c}ierna$. Odmenu by sme mohli dostať za tie karty, ktoré sú num a $\check{c}ierna$, ak nám v tom nezabránia negatívne príklady. V negatívnej časti tabuľky nesmú byť v riadkoch, zodpovedajúcich týmto predikátom samé Y. V opačnom prípade by tieto predikáty neboli schopné rozlíšiť pozitívne a negatívne príklady.

Táto podmienka je splnená. V negatívnej časti tabuľky nie sú v riadkoch pre num a $\check{c}ierna$ samé Y. ■

Zhrnieme: Indukcia ako výpočtový problém sa zjednoduší vtedy, keď limitujeme jazyk, v ktorom pracujeme.

Definícia 5.6 Relácia je *charakteristická*, ak do nej patrí každý pozitívny príklad, *diskriminantná*, ak do nej nepatrí žiadny negatívny príklad. Relácia je *prijateľná*, ak je charakteristická a diskriminantná. Relácia je *produktívna*, ak je vlastnou nadmnožinou Δ^+. ■

Úlohou indukcie – v práve zavedenej terminológii – je definovať v jazyku teórie Γ reláciu, ktorá je prijateľná a produktívna. Ak by sme investovali nejakú energiu do upresnenia pojmu *definovať reláciu*, mohli by sme upresniť (zúžiť) sémantický pojem induktívneho vyplývania, zavedený v definícii 5.2.

Príklad 5.7 V príklade 5.4 je relácia *odmena* definovateľná (pomocou predikátov num a $\check{c}ierna$), charakteristická (všetky pozitívne prípady 1–3 do nej patria), diskriminantná (žiadny negatívny prípad nie je súčasne num a $\check{c}ierna$). Preto je aj prijateľná. Samozrejme, je aj produktívna, pretože je viac čiernych a numerických kariet ako pozitívnych príkladov. ■

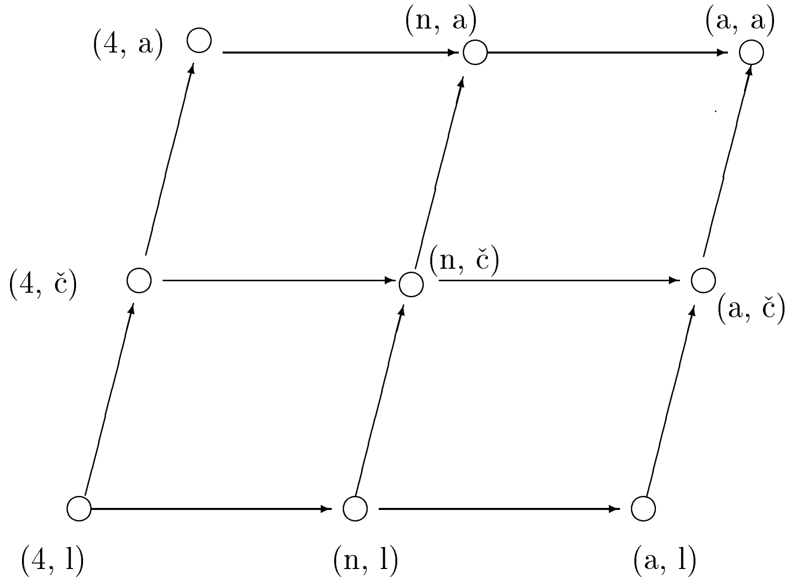
Príklad však bol navrhnutý tak, aby fungoval. Skúsme si ho pokaziť.

Príklad 5.8 V porovnaní s príkladom 5.4 pribudne jeden negatívny príklad a jeden pozitívny príklad zmeníme:
$\Delta^+ = \{$

$$\begin{array}{rcl} odmena(karta(4,l)) & \leftarrow & , \\ odmena(karta(8,l)) & \leftarrow & , \\ odmena(karta(2,p)) & \leftarrow\} & , \end{array}$$

$\Delta^- = \{$

$$\begin{array}{rcl} \neg odmena(karta(5,s)) & \leftarrow & , \\ \neg odmena(karta(j,p)) & \leftarrow & , \\ \neg odmena(karta(7,p)) & \leftarrow\} & . \end{array}$$



Obrázok 5.1: Výpočet induktívneho zovšeobecnenia

Ak by sme si teraz zostrojili tabuľku podobnú na 5.1, ukázalo by sa, že v oboch riadkoch *čierna* a *num* by boli dva Y nielen pre všetky pozitívne príklady, ale aj pre jeden negatívny príklad. Hypotéza

$$odmena(X) \leftarrow num(X) \wedge čierna(X)$$

by teda nebola diskriminantná. Čo je však horšie, vo zvolenom jazyku nie sme schopní sformulovať rozumnú (t.j. prijateľnú a produktívnu) hypotézu.

Ukazuje sa, že potrebná by bola revízia slovníka (napr. zabudovanie predikátov *párny, nepárny*). Tomuto problému, i keď je zaujímavý a dôležitý, sa nebudeme venovať. ■

V [GeN 87] je prezentovaný vtipnejší návod na algoritmické hľadanie hypotézy (pozri obrázok 5.1).

Na obrázku 5.1 sa konštruuje orientovaný graf. Vychádza sa od niektorého pozorovania, v tomto prípade od $odmena(karta(4,l)) \leftarrow$. Priradíme mu vrchol označený ako $(4,l)$. Vrcholy grafu zodpovedajú pozorovaniam (ako aj ich zovšeobecneniam). Z tohto vrcholu vychádzajú dve orientované hrany, jedna do vrcholu $(4,č)$ a druhá do (n,l). Smerujú teda k obom najbližším zovšeobecneniam ($č$ reprezentuje čiernu, n num). Od každého z týchto vrcholov sa graf konštruuje podobným spôsobom ďalej, kým sa nedôjde k vrcholu (a,a), kde a reprezentuje ľubovoľnú farbu i ľubovoľnú hodnotu karty.

Výber ďalších kariet (pozitívnych i negatívnych príkladov) vedie k osekávaniu grafu, skonštruovaného na začiatku. Napríklad pozitívny príklad

$(7, l)$ oseká všetky vrcholy z uvedeného grafu, ktorých označenie obsahuje 4 na prvom mieste. Negatívny príklad $(5, s)$ oseká vrcholy (n, a) a (a, a), lebo vylučuje, že by do úvahy spolu s číslom prichádzala akákoľvek farba. Túto procedúru možno využiť aj na také úlohy, ako je plánovanie experimentu. Môžeme s jej pomocou navrhnúť stratégiu racionálneho postupu pri experimente. (V našom príklade – budeme sa starať o to, aby voľba kariet nebola náhodná, ale cieľavedomá - vyberať sa budú také karty, ktoré v maximálnej možnej miere osekajú konštruovaný graf.)[1]

Objasnili sme si pojem indukcie a analyzovali sme jeden príklad. Teraz si načrtneme, ako možno indukciou získavať hypotézy, implicitné v databázach.

5.1.1 Objavovanie znalostí v databázach

Množstvo databáz v tomto svete obsahuje obrovské objemy dát, ktoré sotva môžu ľudia dostatočne spracovať a využiť (rutinné spracovanie databáz odhaľuje iba nepatrnú časť poznatkov v nich obsiahnutých). Ak v tabuľke sú atribúty A_1, A_2, ľahko vieme zistiť napríklad hodnoty A_2 v tých riadkoch, kde $A_1 = a$. Štandardné databázové nástroje však neumožňujú odhaliť, ako súvisí A_1 a A_2. Zvláštnym prínosom by boli automatické nástroje, ktoré by v databázach objavovali hlbšie skryté tendencie (hypotézy o stave oblasti, zmapovanej dátami z databázy). Preto hľadanie nástroja na odhaľovanie asociácií, pravidelností, hlbších implicitných informácií v databázach je dôležitým praktickým i teoretickým problémom. Vidíme, že nejde o nič iného, ako o indukovanie hypotéz, znalostí, implicitných v databázach.

---

[1] Príbuznú metódu indukcie prezentuje Winston v jednej z najrozšírenejších učebníc umelej inteligencie [Win 93]. Pri aplikácii metódy sa štartuje z pozitívnych a negatívnych pozorovaní. Predstavme si ich ako nejaké k-tice hodnôt (v našom príklade sme mali dvojice hodnôt). Na ich základe sa buduje graf – na jeho vrchu je najvšeobecnejšia k-tica, ktorá môže mať v každom parametri ľubovoľné hodnoty (v našej notácii obsahuje samé a); na spodku je jedna k-tica, ktorá reprezentuje niektoré z pozitívnych pozorovaní (a je súčasne najšpeciálnejším vrcholom v konštruovanom grafe). Ďalšie pozitívne a negatívne pozorovania prispievajú k zovšeobecňovaniu a/alebo špecializácii. Teraz môžem prejsť k tomu, kam mierim:

Na stranách 421 a 422 v [Win 93] sú zarážajúce tvrdenia – niektoré príklady vraj nemôžu byť pozitívnymi príkladmi. Existujúce generalizácie a špecializácie môžu vraj niekedy vylúčiť, aby niektorá k-tica bola pozitívnym príkladom. Stáva sa to vraj vtedy, keď žiadna z generalizácií v grafe nie je generalizáciou tejto k-tice. Táto k-tica jednoducho musí byť negatívnym príkladom.

Žiaľ, Winston nevenuje pozornosť takej drobnosti, že o tom, či táto k-tica je alebo nie je pozitívnym príkladom, rozhoduje príroda. Podobnú situáciu sme mali v príklade 5.7 – pribudol nám negatívny príklad, ktorý pokazil hru umožnenú dovtedajšími príkladmi (pozitívnymi aj negatívnymi). Samozrejme, narazili sme na hranice použitej metódy.

Všeobecnejšie – prístup k problému odvodzovania, ktorému chýba jasná sémantická špecifikácia a je založený iba na opise nejakej procedúry, je vystavený riziku. Môže sa stať, že autori a používatelia navrhovanej odvodzovacej metódy nie sú schopní uvedomiť si jej ohraničenia (a ani jej možnosti).

Domnievam sa, že spôsob prezentácie tejto metódy vo [Win 93] je charakteristický pre istú časť prác v umelej inteligencii.

Najprv zovšeobecníme pojem otázky. Relačná otázka je definovaná nad nejakou množinou relácií a ako odpoveď produkuje opäť nejakú reláciu nad nejakými atribútmi danej databázovej schémy. Tento pojem otázky sa zaujímavým spôsobom zovšeobecňuje v oblasti objavovania znalostí v databázach. Frekventované názvy oblasti, o ktorej hovoríme, sú Data mining a *KDD*, Knowledge discovery in databases (objavovanie znalostí v databázach).

KDD sa zaujíma o všeobecnejšie objekty než relácie. V každej množine faktov obsiahnutých v databáze sú implicitne obsiahnuté nejaké pravidelnosti, asociácie, často sa vyskytujúce vzory. Za *KDD*-objekty sa považujú *pravidlá, klasifikátory a zhluky.* Každý z *KDD*-objektov môže byť odpoveďou na nejakú (*KDD*-)otázku [ImM 96]. *KDD*-objekty sú znalosti implicitné v databáze.

Všimneme si iba pravidlá (a otázky, na ktoré sa odpovedá pravidlami).

Príkladom takejto otázky v jazyku M-SQL [ImM 96] je:

Príklad 5.9 `Select`

```
        From Mine(T) : R
        Where R.Consequent = ( Age = *)
                R.Support > 1000
                R.Confidence > 0.65.
```

Komentáre:

$Mine$ je operátor, ktorý generuje pravidlá z množiny databázových viet T. Samozrejme, $Mine(T)$ sa nemusí vypočítať kompletne. Použitý zápis slúži iba na označenie generátora hypotéz (pravidiel) symbolom R.

Podmienka `R.Consequent = (Age = *)` hovorí, že sa hľadajú iba také pravidlá, ktoré majú v konzekvente vek (a to akýkoľvek vek). Podmienka `R.Support > 1000` vyžaduje, aby aspoň v 1000 databázových vetách bolo splnené telo pravidla. Podmienka `R.Confidence > 0.65` vyjadruje, že aspoň v 65% z databázových viet, ktoré spĺňajú vzor z tela podmienky, je splnený dôsledok.

Príkladom vygenerovaného pravidla môže byť:

```
if X.Diagnosis = "heart disease" and X.Sex = "male",
then X.Age > 50 [1200, 0.7].
```

Teda, vygenerovaná odpoveď (hypotéza) je, že muži so srdcovým ochorením majú viac ako 50 rokov. Toto pravidlo je podopreté aspoň 1200 databázovými vetami a dôsledok platí v aspoň 70% týchto viet.

Uvedená otázka nekladie nijaké podmienky na telo pravidla. Samozrejme, vo všeobecnosti v otázke možno vyjadriť nejaké obmedzujúce podmienky aj na telo pravidla. ■

Z hľadiska reprezentácie poznatkov je spomínaná zmena poňatia otázky mimoriadne dôležitá: Explicitne vyjadrené poznatky sú také isté ako pri tradičnom databázovom spracovaní. Čo sa však radikálne mení, je operátor odvodenia. Zmenený pojem otázky vyžaduje silnejšiu odvodzovaciu mašinériu.

Ako odmenu za toto zosilnenie dostaneme nový pohľad na poznatky implicitné v databáze. Poňatie otázky teda súvisí so silou inferencie (so silou operátora Cn). Ak chceme, aby odpoveďami na naše otázky boli pravidlá, musíme tomu prispôsobiť mašinériu odpovedajúcu na otázky.

Z výpočtového hľadiska je rozhodujúce vygenerovať množstvo frekventovaných vzorov v databáze. Tieto vzory môžu slúžiť ako telá pravidiel.

Jednoduchý typ algoritmov počíta asi takto: Hľadá telá pravidiel. To môže znamenať napríklad (v najjednoduchšom prípade), že hľadá konjunkcie atómov typu *Atribút = hodnota*. Najprv 1-členné konjunkcie (a vypočítava ich podporu, t.j. frekvenciu výskytu). Potom hľadá 2-členné konjunkcie (aj frekvenciu ich výskytu) atď. Kladením otázok (ako v príklade 5.9) limitujeme prehľadávaný priestor: Zaregistruje iba tie telá, ktoré spĺňajú podmienku kladenú na podporu a prípadné podmienky na telo pravidla. Súčasne zisťuje, či v požadovanom percente je splnený aj špecifikovaný konzekvent pravidla alebo zisťuje, aký konzekvent a s akou vieryhodnosťou platí pre dané telo.

Príklad *KDD* prekračuje našu idealizovanú predstavu o indukcii. Nie všetky pozorovania, t.j. databázové vety, vyplývajú deduktívne z vygenerovanej hypotézy (pravidla). Keď zovšeobecňujeme, veľmi často musíme brať do úvahy štatistický charakter pravidelností pozorovanej oblasti. V príklade 5.9: vo všetkých 1200 relevantných riadkoch neplatí súčasne, že hodnotou atribútu *Diagnosis* je *heart disease*, atribútu *Sex* je *male* a atribútu *Age* je číslo väčšie ako 50. Pravdepodobnostným aspektom indukcie rovnako aj pravdepodobnostnej charakterizácii usudzovania sa však nebudeme venovať.[2]

5.1.2 Induktívne logické programovanie

Štúdiu indukcie sa budeme odteraz venovať v prostredí induktívneho logického programovania [MuR 94]. Toto prostredie má niektoré zjavné prednosti: Jazyk logického programovania poskytuje dobrý nástroj na reprezentáciu poznania v pozadí (background knowledge). Bez poznania v pozadí si sotva možno predstaviť komplikovanejšiu a realistickú indukciu. Tradičnejšie techniky strojovej indukcie a učenia mali práve s reprezentáciou poznania v pozadí problémy.

Začneme opäť príkladom:

Príklad 5.10 ([MuC 90]) Predstavme si, že z teórie v pozadí a z nejakých pozitívnych a negatívnych príkladov chceme odvodiť (indukovať) logický program *quicksort*. V našom príklade teóriu budú tvoriť program *part* (umožňuje rozklad vstupného zoznamu podľa nejakého prvku na 2 časti), program *app* (spájajúci zoznamy) a príklady sú o tom, ako sa správa a nespráva quicksort.

[2] V najbližšom období možno očakávať zvýšené úsilie, venované prekonaniu tohto slabého miesta výskumných záujmov v oblasti nemonotónneho usudzovania. Pravdepodobnostné, mnohohodnotové a vo všeobecnosti kvantitatívne aspekty nemonotónneho (hypotetického) usudzovania si zasluhujú zvýšenú pozornosť [Mar 00].

$\Gamma = \{$

$$\begin{array}{rcl} part(X,[\,],[\,],[\,]) & \leftarrow & , \\ part(X,[H|T],[H|S1],S2) & \leftarrow & H \leq X, part(X,T,S1,S2), \\ part(X,[H|T],S1,[H|S2]) & \leftarrow & H > X, part(X,T,S1,S2), \\ app([\,],L,L) & \leftarrow & , \\ app([H|T],L1,[H|L2]) & \leftarrow & app(T,L1,L2)\}. \end{array}$$

Predikát *part* sa definuje takto: prvok X rozdelí prázdny zoznam [ ] na dva prázdne zoznamy. Ak máme prvok X a zoznam $[H|T]$ s prvým prvkom (hlavou) H a zvyškom (telom) T, rozdelíme zoznam na dve časti, prvá obsahuje prvky, ktoré nie sú väčšie ako X, druhá ostatné. Definícia je rekurzívna: postupne porovnávame hlavy zoznamov s X a pokračujeme v delení tela.
$\Delta^+ = \{$

$$\begin{array}{rcl} qsort([\,],[\,]) & \leftarrow & , \\ qsort([0],[0]) & \leftarrow & , \\ qsort([1,0],[0,1]) & \leftarrow\} & . \end{array}$$

Pozitívnymi príkladmi sú: usporiadaním prázdneho zoznamu je prázdny zoznam, usporiadaním jednoprvkového zoznamu [0] je [0], usporiadaním zoznamu [1, 0] je zoznam [0, 1].
$\Delta^- = \{$

$$\begin{array}{rl} \leftarrow & qsort([1,0],[1,0]), \\ \leftarrow & qsort([0],[\,])\}. \end{array}$$

V prostredí logického programovania sa negatívne príklady vyjadrujú ako integritné obmedzenia. Korektné zovšeobecnenie nesmie byť splnené žiadnym negatívnym príkladom. Zmysel tu uvedených negatívnych príkladov by už mal byť zrejmý.

Cieľom je teraz syntetizovať program quicksort (definíciu predikátu *qsort*). Tento program je vlastne množinou hypotéz, ktorú treba vygenerovať.
$\Phi = \{$

$$\begin{array}{rcl} qsort([\,],[\,]) & \leftarrow & , \\ qsort([H|T],S) & \leftarrow & part(H,T,L1,L2), qsort(L1,S1), qsort(L2,S2), \\ & & app(S1,[H|S2],S)\}. \end{array}$$

Vygenerovaný program *quicksort* dostáva ako vstup nejaký zoznam a jeho výstupom je utriedený zoznam. Funguje tak, že rozdelí vstupný zoznam na dve časti: Zoznam $L1$ prvkov nie väčších ako hlava vstupného zoznamu spolu so zoznamom $L2$ ostatných prvkov. Potom sa utriedia $L1$ a $L2$, nakoniec sa druhý utriedený zoznam pripojí k prvému utriedenému zoznamu. Samozrejme, program je rekurzívny. Rekurzívny rozklad sa zastaví na prázdnych zoznamoch. ■

Vo všeobecnom prípade, problém pre systém induktívneho logického programovania je špecifikovaný, keď máme daný logický program s touto štruktúrou (alebo, lepšie, trojicu logických programov): teória v pozadí Γ, pozitívne príklady (množina základných faktov Δ^+) a negatívne príklady (Δ^-, množina integritných obmedzení, každé integritné obmedzenie obsahuje jediný základný atóm).[3]

Teraz sa budeme venovať sémantickej charakterizácii induktívneho odvodenia (v induktívnom logickom programovaní).

Sémantika

Sémantické podmienky budeme formulovať v tom istom duchu ako v definícii indukcie 5.2.

Definícia 5.11 Nech Δ^+ a Δ^- sú príklady, Γ je teória v pozadí, Λ je nekonzistentná množina viet. Potom Φ je *korektná množina hypotéz*, ak platí:

(i) $\Gamma \cup \Delta^- \not\models \Lambda$,

(ii) $\Gamma \cup \Phi \cup \Delta^- \not\models \Lambda$,

(iii) $\Gamma \not\models \Delta^+$,

(iv) $\Gamma \cup \Phi \models \Delta^+$.

■

Požaduje sa teda, aby negatívne príklady boli konzistentné s teóriou v pozadí (podmienka (i)), konzistetntosť sa zachováva aj po pridaní vygenerovanej hypotézy (podmienka (ii)). Tieto dve vlastnosti sa zvyknú nazývať apriórnou a aposteriórnou splniteľnosťou. Keď si uvedomíme, že negatívne príklady sú vlastne integritné obmedzenia, vidíme, že požadujeme, aby teória a hypotézy neboli v spore s integritnými obmedzeniami.

Príklad 5.12 Vráťme sa k príkladu 5.10. Ak by vygenerovaný program *quicksort* dovolil považovať $qsort([1,0],[1,0])$ za inštanciu definovaného predikátu *qsort*, potom by tento program narušil integritné obmedzenie

$$\leftarrow qsort([1,0],[1,0]).$$

■

Ďalej sa požaduje, aby z teórie nevyplývali pozitívne príklady (podmienka (iii)). Podmienka (iv) hovorí, že z teórie a hypotézy pozitívne príklady vyplývajú (tieto vlastnosti sa nazývajú apriórnou nutnosťou a aposteriórnou postačujúcosťou).

[3]V niektorých systémoch induktívneho logického programovania sa berú do úvahy ako pozitívne či negatívne príklady aj komplikovanejšie klauzy.

Ak sú použité logické programy definitné, ich význam je jednoznačne určený minimálnym modelom, a preto podmienky (i) - (iv) možno vyjadriť takto:

Definícia 5.13 Nech Δ^+ a Δ^- sú príklady, Γ je definitná teória v pozadí, Λ je nekonzistentná množina viet. Potom Φ, množina definitných kláuz, je *korektná množina hypotéz*, ak platí:

(i) každé $e \in \Delta^-$ je nepravdivé v $M_{min}(\Gamma)$, kde $M_{min}(\Gamma)$ je minimálny model teórie Γ,

(ii) každé $e \in \Delta^-$ je nepravdivé v $M_{min}(\Gamma \cup \Phi)$,

(iii) aspoň jedno $e \in \Delta^+$ je nepravdivé v $M_{min}(\Gamma)$,

(iv) každé $e \in \Delta^+$ je pravdivé v $M_{min}(\Gamma \cup \Phi)$.

■

Generický algoritmus

Je čas prejsť k spôsobom výpočtu induktívnej hypotézy v kontexte induktívneho logického programovania. Pripomeňme si, ako sme počítali induktívnu hypotézu z vovádzajúceho príkladu 5.1. Vytvorili sme priestor možných hypotéz, určený jazykom, v ktorom sa hypotézy môžu formulovať. Tento priestor sme systematicky orezávali (vylučovali sme jeho vrcholy, ktoré neboli v súlade s pozitívnymi alebo negatívnymi príkladmi). Orezávali sme a hľadali vhodnú hypotézu dovtedy, kým sme nedosiahli stav, ktorý by splĺňal nejaké rozumné kritérium.

Na takejto myšlienke je založený aj generický algoritmus Muggletona a de-Raedta [MuR 94], pozri tabuľku 5.2. Popísali ho na značnej úrovni abstrakcie od výpočtových detailov.

Sú rôzne možné stratégie, ako inicializovať QH. Najopatrnejší prístup štartuje iba z teórie v pozadí a pozitívnych pozorovaní, $QH = \Gamma \cup \Delta^+$. Tento prístup využíva pravidlá indukcie, postupuje smerom od špecifickejších hypotéz k všeobecnejším. Opačne možno začať od najvšeobecnejšej hypotézy (množiny nekonzistentných kláuz Λ) a postupnou špecializáciou (s využitím pravidiel dedukcie) odstraňovať nekonzistentnosti (pripomeňme, že negatívne príklady slúžia ako integritné obmedzenia).

Rozumné je priestor QH štrukturovať – usporiadať podľa vzťahov špecializácie a generalizácie. Takéto usporiadanie umožňuje voliť efektívne stratégie výpočtu induktívnych generalizácií. Celý výpočet vlastne spočíva v prehľadávaní priestoru QH.

Výber hypotézy z QH opäť môže byť založený na ľubovoľnej stratégii prehľadávania priestoru QH. Pri výbere možno využiť nejaké predspracovanie a vybrať hypotézu, ktorá korektne klasifikuje najväčší počet prípadov.

Algoritmus 5.14 (Generický algoritmus)
vstup: *množina inferenčných pravidiel R*
výstup: *množina hypotéz QH*

inicializuj QH
repeat
  vyber H z QH
  vyber inferenčné pravidlá z R, aplikovateľné na H
  aplikuj pravidlá na H – nech výsledkom tejto aplikácie sú hypotézy $H_1, \ldots, H_n$
  $QH := (QH \setminus \{H\}) \cup \{H_1, \ldots, H_n\}$
  osekaj QH
until *QH splňa kritérium zastavenia.*

Tabuľka 5.2: Generický algoritmus indukcie

Inferenčné pravidlá môžu množinu hypotéz rozšíriť.

Osekávanie sa môže opierať o vzťahy generalizácie a špecializácie: ak pozitívne príklady nevyplývajú z nejakej hypotézy H (a teórie Γ), potom nevyplývajú zo žiadnej z hypotéz, špeciálnejších ako H (keby totiž pozitívne príklady vyplývali zo špeciálnejších hypotéz, museli by vyplývať aj z H). Podobne, ak nejaká hypotéza je nekonzistená s teóriou a pozorovaniami, potom to isté platí aj pre jej zovšeobecnenie.

Kritérium zastavenia možno implementovať rôznym spôsobom.

Uvedený generický algoritmus je zovšeobecnením rôznych možných stratégií výpočtu induktívnych zovšeobecnení.

Prejdeme k analýze pravidiel indukcie. Pravidlá R sa konštruujú ako inverzné k deduktívnym pravidlám.

θ-subsumcia

Najjednoduchším príkladom induktívneho pravidla je θ-subsumcia ([Plo 70]):

Definícia 5.15 Nech c_1 a c_2 sú klauzy[4] a θ je substitúcia. Potom c_1 získame induktívne z c_2, ak $c_1\theta \subseteq c_2$.

Hovoríme, že c_1 θ-subsumuje c_2. Hovoríme aj, že c_1 je *všeobecnejšia* ako c_2 (a c_2 je *špeciálnejšia* ako c_1). ■

Konvencia 5.16 Keď existuje substitúcia θ taká, že $c_1\theta \subseteq c_2$, budeme používať značenie $c_1 \leq c_2$. ■

[4]Množiny literálov, pripomeňme si.

Vidno, že ide o prosté zovšeobecňovanie takým spôsobom, že sa (niektoré) konštanty z predpokladov nahradzujú premennými (pričom sa môže abstrahovať od niektorých nepodstatných literálov). Indukcia sa tu skutočne ukazuje ako inverzia dedukcie: c_1 získavame indukciou z c_2, pritom c_2 je deduktívnym dôsledkom c_1. Ilustruje to aj nasledujúci príklad.

Príklad 5.17 Nech c_2 je

$$\{otec(j,p), \neg rodi\check{c}(j,p), \neg rodi\check{c}(j,a), \neg mu\check{z}(j), \neg \check{z}ena(a)\},$$

c_1 je $\{otec(X,Y), \neg rodi\check{c}(X,Y), \neg mu\check{z}(X)\}$.

Očividne c_1 θ-subsumuje c_2, pričom $\theta = \{X|j, Y|p\}$. Teda, j substituujeme za X, p za Y. ■

Vidno však, že prostá θ-subsumcia je príliš slabý vzťah. V príklade 5.17 je možných viac c takých, ktoré θ-subsumujú c_2. Pritom nie sú tak vyhovujúce ako c_1 z príkladu 5.17.

Venujme sa teda problému, ktoré θ-subsumujúce generalizácie sú rozumné.

Jednou z vád príkladu 5.17 je to, že sa zovšeobecňuje z jedinej klauzy. Rozumné je zovšeobecňovať z nejakej množiny kláuz. Budeme si teda všímať množiny kláuz a hľadať ich generalizáciu.

Definícia 5.18 Nech C je množina kláuz. Budeme hovoriť, že klauza c je *zovšeobecnením* C práve vtedy, keď c je všeobecnejšia ako každá klauza $e \in C$. ■

Samozrejme, množina kláuz môže mať viac generalizácií. My sa budeme zaujímať o tie zovšeobecnenia, ktoré sú najmenej zaväzujúce. To znamená, že za rozumnú stratégiu pri generovaní hypotéz (zovšeobecnení) považujeme takú, ktorá dovoľuje prijímať hypotézy, ktoré sú najslabšie (najmenej všeobecné) zo všetkých možných hypotéz:

Definícia 5.19 Nech C je množina kláuz. Budeme hovoriť, že klauza c je *najslabším zovšeobecnením* C práve vtedy, keď

- c je zovšeobecnením C,
- ak e je zovšeobecnením C, potom existuje substitúcia σ taká, že $e\sigma \subseteq c$ (alebo $e \leq c$, teda e je všeobecnejšie ako c).

■

Nasledujúci príklad z [Plo 70] ilustruje a približuje induktívnu generalizáciu, založenú na θ-subsumcii.

Príklad 5.20 Predstavme si hru, ktorá sa hrá na ploche s dvoma štvorcami. Hru hrajú dvaja hráči. Výherné pozície pre hráča č.1 sú na obrázku 5.2.

Tieto pozície si môžeme reprezentovať nasledujúcimi klauzami:

$$\begin{aligned}(vysk(o,1,p_1) \wedge vysk(x,2,p_1)) &\rightarrow vyhr(p_1),\\ (vysk(x,1,p_2) \wedge vysk(x,2,p_2)) &\rightarrow vyhr(p_2).\end{aligned}$$

Prvá klauza definuje vyhrávajúcu pozíciu p_1. V tejto pozícii sa vyskytuje znak o na prvom (ľavom) štvorci hracej plochy a na druhom štvorci sa v tej istej pozícii vyskytuje znak x. Druhá klauza podobne charakterizuje vyhrávajúcu pozíciu p_2.

Našou úlohou je teraz nejak zovšeobecniť tieto dve klauzy, teda nájsť jedinú klauzu, ktorá by presne charakterizovala uvedené dve vyhrávajúce pozície.

V klauzálnom zápise:

$$\begin{aligned}c_1 &= \{\neg vysk(o,1,p_1), \neg vysk(x,2,p_1), vyhr(p_1)\},\\ c_2 &= \{\neg vysk(x,1,p_2), \neg vysk(x,2,p_2), vyhr(p_2)\}.\end{aligned}$$

Použijeme techniku, pri ktorej pre každú dvojicu porovnateľných literálov z oboch kláuz nájdeme ich najslabšie zovšeobecnenie (porovnateľné literály obsahujú rovnaký predikát a buď sú oba pozitívne alebo oba negatívne).

Najslabším zovšeobecnením $\neg vysk(x,2,p_1)$ a $\neg vysk(x,2,p_2)$ je

$$\neg vysk(x,2,P),$$

kde P je premenná. Konštruujeme teda substitúcie $\theta_1 = \{P|p_1\}$ a $\theta_2 = \{P|p_2\}$. To znamená, že $\neg vysk(x,2,P)$ θ_1-subsumuje $\neg vysk(x,2,p_1)$ a θ_2-subsumuje $\neg vysk(x,2,p_2)$ (možno treba pripomenúť: zovšeobecnenia netvoríme substitúciou, ale špeciálnejší prípad možno získať substitúciou zo všeobecnejšieho). Naším cieľom je vytvoriť také zovšeobecnenie d, ktoré θ_1-subsumuje klauzu c_1 a θ_2-subsumuje klauzu c_2. Podobne, najslabším zovšeobecnením $\neg vysk(o,1,p_1)$ a $\neg vysk(x,1,p_2)$ je

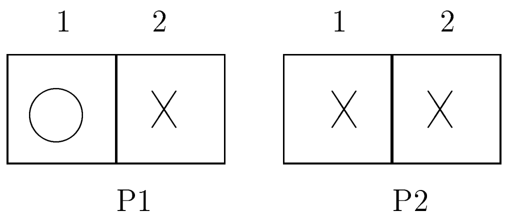
$$\neg vysk(U,1,P),$$

pričom „rozširujeme“ (kompozíciou) naše substitúcie: $\theta_1 = \{P|p_1, U|o\}$, $\theta_2 = \{P|p_2, U|x\}$. Pokračujeme tak, aby sa prebrala každá dvojica porovnateľných literálov: najslabším zovšeobecnením $\neg vysk(o,1,p_1)$ a $\neg vysk(x,2,p_2)$ je

$$\neg vysk(U,S,P).$$

Po nájdení tohto zovšeobecnenia je

$$\begin{aligned}\theta_1 &= \{P|p_1, U|o, S|1\},\\ \theta_2 &= \{P|p_2, U|x, S|2\}.\end{aligned}$$



Obrázok 5.2: Vyhrávajúce pozície

Najslabším zovšeobecnením $\neg vysk(x,2,p_1)$ a $\neg vysk(x,1,p_2)$ je

$$\neg vysk(x,Z,P).$$

„Vyvíjajúce sa“ substitúcie:

$$\begin{aligned}\theta_1 &= \{P|p_1, U|o, S|1, Z|2\},\\ \theta_2 &= \{P|p_2, U|x, S|2, Z|1\}.\end{aligned}$$

Najslabším zovšeobecnením $vyhr(p_1)$ a $vyhr(p_2)$ je

$$vyhr(P)$$

(už sme vystačili s daným θ_1 a θ_2).

Získali sme klauzu $d = \{$

$$\neg vysk(x,2,P), \neg vysk(U,1,P), \neg vysk(U,S,P), \neg vysk(x,Z,P), vyhr(P)\},$$

ktorá je zovšeobecnením množiny kláuz $C = \{c_1, c_2\}$. Platí totiž, že $d\theta_1 \subseteq c_1$ a $d\theta_2 \subseteq c_2$. Platí aj, že d je najslabším zovšeobecnením množiny kláuz C (na základe nižšie uvedeného tvrdenia 5.24).

Všimnime si však klauzu $d' = \{\neg vysk(x,2,P), \neg vysk(U,1,P), vyhr(P)\}$.

Očividne, identická substitúcia ϵ zaručuje, že $d'\epsilon \subseteq d$. Na druhej strane však, $d\theta \subseteq d'$, ak napríklad $\theta = \{S|1, Z|2\}$. Teda, d a d' sú ekvivalentné vzhľadom na vzťah θ-generalizácie. To znamená, že môže existovať viac najslabších generalizácií. Z triedy ekvivalentných najslabších generalizácií je rozumné vyberať tie, ktoré sú tak zredukované, ako sa dá – v našom prípade takou je d'. (Napokon, d' aj presnejšie a intuitívnejšie vyjadruje, ktoré pozície sú vyhrávajúce.) ■

Teraz si vyložíme teoretické východiská, využité v príklade 5.20.

V príklade sme konštruovali najslabšie zovšeobecnenie dvoch literálov. Plotkin v [Plo 70] uvádza algoritmus 5.21 (tabuľka 5.3), ktorý sa pre tento cieľ používa.

Algoritmus 5.21
vstup: *porovnateľné literály* L_1, L_2
výstup: L, *najslabšie zovšeobecnenie* L_1 *a* L_2

1. *pre* $i = 1, 2$ *bude* $V_i := L_i$, $\theta_i := \epsilon$, *kde* ϵ *je prázdna susbstitúcia,*

2. **if** t_1, t_2 *sú dva rôzne termy, ktoré sa vyskytujú na tom istom mieste vo* V_1, V_2,
   then *zvoľ takú premennú* ξ, *ktorá sa nevyskytuje vo* V_1, V_2
   a nahraď t_1 *a* t_2 *novozavedenou premennou všade tam, kde sa nachádzajú na tom istom mieste vo* V_1 *a* V_2*;*
   else return V_1 *(** V_1 *je najslabším zovšeobecnením* $\{L_1, L_2\}$*, pričom* $V_1 = V_2$, $V_i\theta_i = L_i$ **);*

3. $\theta_i := \{\xi|t_i\} \circ \theta_i$, *kde* $\circ$ *je kompozícia substitúcií (* konštruujú sa dve rôzne substitúcie, v každej z nich sa premenná* ξ *substituuje zodpovedajúcim termom z* V_i **);*

4. *choď na krok 2.*

Tabuľka 5.3: Algoritmus počítajúci zovšeobecnenie dvoch literálov

Tvrdenie 5.22 (Plotkin [Plo 70]) *Množina literálov má najslabšie zovšeobecnenie práve vtedy, keď každá dvojica literálov z tejto množiny je porovnateľná. Algoritmus 5.21 je korektný (počíta to, čo má počítať).*[5]

K hľadaniu najslabšieho zovšeobecnenia dvoch literálov sme sa v príklade 5.20 dostali od hľadania najslabšieho zovšeobecnenia dvoch kláuz. Najslabšie zovšeobecnenie dvoch kláuz sa hľadá práve cez konštrukciu najslabších zovšeobecnení každej dvojice porovnateľných literálov z oboch kláuz. Pozri algoritmus 5.23, tabuľka 5.4. V tomto algoritme sa volá modifikovaný algoritmus 5.21, ktorý hľadá zovšeobecnenia literálov. Modifikácia je v tom, že substitúcia, skonštruovaná pri výpočte zovšeobecnenia dvoch literálov, sa stáva vstupnou substitútciou pre ďalšiu dvojicu literálov. Dôvodom pre túto modifikáciu je, aby sa zachovala previazanosť jednotlivých argumentov: napríklad, vyhrávajúcu pozíciu P by sme ťažko definovali korektne, keby sa v danej klauze medzi argumentmi predikátov *vyhr* a *vysk* vyskytovali rôzne premenné, reprezentujúce pozíciu.

[5] Aj v tejto časti sú tvrdenia často bez dôkazov. Slúžia iba ako základná informácia, prípadný záujemca o detaily môže siahnuť po citovanej literatúre.

Algoritmus 5.23
vstup: *dvojica kláuz* $S = \{C_1, C_2\}$
výstup: C, *najslabšie zovšeobecnenie* S

1. *z* C_1, C_2 *vyber všetky možné dvojice porovnateľných literálov* $L_{1,j}, L_{2,j}$, *kde* $j = 1, \ldots, n$,
2. *ku každej dvojici* $L_{1,j}, L_{2,j}$ *nájdi podľa modifikovaného algoritmu 5.21 najslabšie zovšeobecnenie* M_j *(jedinou modifikáciou je, že vstupnými substitúciami* θ_1 *a* θ_2 *pre* $L_{1,j+1}$ *a* $L_{2,j+1}$ *sú výstupné substitúcie* $L_{1,j}$ *a* $L_{2,j}$;
3. $C = \{M_j : j = 1, \ldots, n\}$ *je najslabším zovšeobecnením* S.

Tabuľka 5.4: Algoritmus počítajúci najslabšie zovšeobecnenie dvojice kláuz

Tvrdenie 5.24 (Plotkin [Plo 70]) *Ak* C_1, C_2 *sú klauzy, ktoré obsahujú aspoň jednu dvojicu porovnateľných literálov, potom algoritmus 5.23 nájde nejaké ich najslabšie zovšeobecnenie.*

Klauzy (a aj najslabšie zovšeobecnenia) môžu byť ekvivalentné vzhľadom na vzťah najslabšieho zovšeobecnenia.

Definícia 5.25 Ak c a e sú klauzy, pričom $c \leq e$ a $e \leq c$, budeme hovoriť, že c a e sú ekvivalentné a značiť to ako $c \sim e$. ■

Do tej istej triedy ekvivalencie patria napríklad aj

$$\{p(X), p(f(a))\} \text{ a } \{p(f(a))\}.$$

Teda, klauza $\{p(X), p(f(a))\}$ je ekvivalentná svojej vlastnej podmnožine $\{p(f(a))\}$. Užitočné je hľadať minimálnych reprezentantov tried ekvivalencie.

Definícia 5.26 Klauza c je *redukovaná* práve vtedy, keď pre ľubovoľnú klauzu d platí: ak $d \subseteq c$ a $d \sim c$, potom $c = d$. ■

Redukovaná klauza nie je ekvivalentná žiadnej svojej vlastnej podmnožine. Redukované klauzy počíta algoritmus 5.29, tabuľka 5.5.

Definícia 5.27 c, d sú alfabetické varianty, ak existujú substitúcie σ, θ také, že $c\sigma = d$ a $d\theta = c$. ■

Tvrdenie 5.28 *Ak* $c \sim d$, *pričom* c, d *sú redukované, potom* c *a* d *sú alfabetické varianty.*

Algoritmus 5.29
vstup: *klauza c*
výstup: *redukovaná klauza e taká, že* $e \sim c$

1. $e := c$,
2. *nájdi literál* $l \in e$ *a substitúciu* σ *takú, že* $e\sigma \subseteq e \setminus l$*; v prípade, že taký literál neexistuje, zastaviť:* **return** e,
3. $e := e\sigma$*; návrat ku kroku 2.*

Tabuľka 5.5: Algoritmus počítajúci redukovanú klauzu

Tvrdenie 5.30 (Plotkin [Plo 70]) *Ak vstupom algoritmu 5.29 je klauza c a výstupom klauza e, potom e je redukovaná klauza a* $c \sim e$.

Silnejšie metódy

Ľahko vidieť, že θ-subsumcia je príliš slabá metóda na induktívne zovšeobecňovanie. Ak sa vrátime k príkladu 5.1 a najmä k jeho klauzálnej formulácii z príkladu 5.4 vidíme, že θ-subsumcia by nám nestačila na odvodenie zaujímavého induktívneho záveru.

Príklad 5.31 Najslabšie zovšeobecnenie (podľa θ-subsumcie) množiny $\Delta^+ = \{$

$$\begin{aligned} odmena(karta(4,l)) &\leftarrow \quad, \\ odmena(karta(7,l)) &\leftarrow \quad, \\ odmena(karta(2,p)) &\leftarrow \} \end{aligned}$$

je príliš silné (a nezaujímavé):

$$odmena(karta(X,Y)) \leftarrow .$$

θ-subsumcia neumožňuje brať do úvahy negatívne príklady a ani poznanie z pozadia.

Cieľom je zosilniť pravidlá indukcie tak, aby sa dali odvodiť zaujímavejšie induktívne hypotézy. ■

Príklad 5.32 Podobne pre $c_1 = p(f(X)) \leftarrow p(X)$ a $c_2 = p(f(f(X)) \leftarrow p(X)$ vidíme, že $c_1 \models c_2$. Teda, ak indukciu a dedukciu chápeme ako v istom zmysle slova inverzné operácie, mohli by sme očakávať, že z c_2 by sme mali indukciou získať c_1. Žiaľ, θ-subsumciou to nejde. ■

Jednou z metód, silnejších než θ-subsumcia je *zovšeobecnená subsumcia*, pozri [Plo 71] alebo [Bun 88].

Budeme používať iba definitné klauzy. (Kvôli osvieženiu pamäti: definitné klauzy obsahujú presne jeden pozitívny literál.) Klauzy budeme písať v implikačnom tvare $A \leftarrow A_1, \ldots, A_k$, kde A, A_i sú atómy a $k \geq 0$. Označme si túto klauzu ako c. Pripomenieme aj, že A sa nazýva *hlava* klauzy (tu ju budeme značiť c_h), $A_1, \ldots, A_k$ je jej *telo* (značenie: c_t).

Zovšeobecňovanie v priestore hornovských kláuz pojmeme tak, že klauzu s hlavou A môže zovšeobecniť iba klauza s tou istou hlavou. Základná intuícia je asi táto: klauzu c budeme považovať za všeobecnejšiu (lepšiu) ako klauzu d vzhľadom na logický program P vtedy, keď platí:

ak d zodpovie otázku Q pomocou programu P, potom aj c zodpovie otázku Q pomocou P.

Znamená to toľko, že c zodpovie prinajmenšom tie otázky, čo d, obvykle však aj ďalšie. Preto je c všeobecnejšie ako d. Túto intuíciu možno formalizovať pomocou pojmu pokrytia atómu (nejakou klauzou pri nejakej interpretácii). Atóm reprezentuje otázku (presnejšie: odpoveď na otázku), interpretácia reprezentuje program. Keďže hovoríme o definitných programoch, program budeme najadekvátnejšie reprezentovať tou interpretáciou, ktorá je jeho minimálnym modelom.

Definícia 5.33 Nech c je definitná klauza, A je základný atóm a I je herbrandovská interpretácia. Budeme hovoriť, že c *pokrýva* A v I práve vtedy, keď existuje substitúcia θ taká, že $c_h\theta = A$ a $\exists(c_t\theta)$ je splnená v I.

Množina kláuz C *pokrýva* A v I práve vtedy, keď aspoň jedna klauza $c \in C$ pokrýva A v I. ■

Príklad 5.34 Nech $P = \{$

$$\begin{array}{rcl} otec(X, eva) & \leftarrow & rodič(X, eva), muž(X), \\ rodič(karol, eva) & \leftarrow & , \\ muž(karol) & \leftarrow \} & . \end{array}$$

$\mathcal{I} = M_P = \{muž(karol), rodič(karol, eva), otec(karol, eva)\}$.
Nech ďalej otázkou Q je $\leftarrow otec(X, Y)$, teda pýtame sa na dvojice X, Y, pre ktoré z nášho programu vyplýva, že X je otcom Y. Rozumnou odpoveďou je

$$otec(karol, eva),$$

teda nejaký základný atóm A. Nech klauza c je

$$otec(X, Y) \leftarrow rodič(X, Y), muž(X).$$

Vidíme, že c *pokrýva* A v interpretácii M_P. ■

Pri induktívnom zovšeobecňovaní je cieľom prehľadávať priestor hypotéz (induktívnych zovšeobecnení) a nájsť také, ktoré pokrývajú dané fakty (pozorovania).[6]

Vybudujeme teóriu takéhoto zovšeobecňovania v prostredí logických programov. Budeme si všímať nejaký logický program P (ktorý bude predstavovať znalosti v pozadí) a pojem zovšeobecnenia budeme definovať vzhľadom na P. Pritom možné pozorovania bude predstavovať základný atóm A.

Definícia 5.35 Klauza c je vzhľadom na definitný logický program P *všeobecnejšia* ako klauza d (značenie $c \leq_P d$) práve vtedy, keď pre každý základný atóm A a pre každú interpretáciu I jazyka programu $P \cup \{c\} \cup \{d\}$, ktorá je modelom programu P, platí:

ak d pokrýva A vzhľadom na I, potom aj c pokrýva A vzhľadom na I. ∎

Príklad 5.36 Vzhľadom na definitný program P z príkladu 5.34 je klauza $c = (otec(X, Y) \leftarrow rodi\check{c}(X, Y), mu\check{z}(X))$ všeobecnejšia ako klauza

$$d = (otec(X, eva) \leftarrow rodi\check{c}(X, eva), mu\check{z}(X)).$$

∎

Fakt 5.37 *Nech I je model programu P, c a d sú nejaké klauzy. Potom $c \leq_P d$ práve vtedy, keď $T_{P\cup\{d\}}(I) \subseteq T_{P\cup\{c\}}(I)$.*

Idea dôkazu: $T_{P\cup\{d\}}(I)$ a $T_{P\cup\{c\}}(I)$ sa líšia iba o základné atómy, ktoré sú inštanciami hláv d_h a c_h. Na základe definícií 5.33 a 5.35 každá inštancia hlavy d_h je aj inštanciou hlavy c_h práve vtedy, keď $c \leq_P d$. ∎

Definícia 5.38 Nech C a D sú konečné množiny kláuz. Potom $C \leq_P D$ práve vtedy, keď $(\forall d \in D)(\exists c \in C)\ c \leq_P d$. ∎

Nasledujúce tvrdenie dáva základ pre testovanie vzťahu zovšeobecnenia. Vyslovuje asi toto: Na test zovšeobecnenia treba ukázať, že všeobecnejšiu hlavu možno substitúciou stotožniť so špeciálnejšou. Program spolu s telom špeciálnejšej klauzy umožňuje odvodiť nejakú inštanciu tela všeobecnejšej klauzy (pripraviť odvodenie zodpovedajúcej inštancie jej hlavy).

Tvrdenie 5.39 (Buntine [Bun 88]) *Nech P je definitný logický program, c a d sú definitné klauzy s disjunktnými množinami premenných. Nech ďalej θ je substitúcia, ktorá nepoužíva konštanty z c, d, P, pričom $d\theta$ je základná klauza.*

Potom $c \leq_P d$ práve vtedy, keď existuje substitúcia σ taká, že

- $c_h\sigma = d_h$,

[6] Tieto pozorovania môžeme chápať aj ako otázky, ktoré kladie príroda.

- $P \cup \{d_t\theta\} \models \exists(c_t\sigma\theta)$.

Dôkaz:
$\Leftarrow$
Predpokladáme, že $c_h\sigma = d_h$ a $P \cup \{d_t\theta\} \models \exists(c_t\sigma\theta)$.

Máme dokázať, že $c \leq_P d$. Nech teda d pokrýva nejaký základný atóm $A \in B_{P\cup\{c\}\cup\{d\}}$ v I. I je interpretácia jazyka $P\cup\{c\}\cup\{d\}$. Nech I je modelom programu P. Z predpokladu pokrývania dostávame, že existuje substitúcia ϕ taká, že $d_h\phi = A$ a $\exists(d_t\phi)$ je pravdivé v I.

Treba teraz dokázať, že c pokrýva A vzhľadom na I. Rozšírme substitúciu ϕ na ϕ' tak, aby $d_t\phi'$ bola základná klauza pravdivá v I. (Očividne $d_h\phi = A = d_h\phi'$.) Z predpokladu $P \cup \{d_t\theta\} \models \exists(c_t\sigma\theta)$ dostávame, že $\exists(c_t\sigma\phi')$ je splnené v I. Pretože platí $c_h\sigma\phi = A = c_h\sigma\phi'$, c pokrýva A v I.
$\Rightarrow$
Predpokladáme, že $c \leq_P d$. Najprv dokážeme existenciu substitúcie σ, pre ktorú $c_h\sigma = d_h$: Majme substitúciu θ takú, že $d\theta$ je základná klauza. Nech $I = B_{P\cup\{c\}\cup\{d\}\cup\{d\theta\}}$. Zrejme platí, že d pokrýva $d_h\theta$ v I. Na základe definície $\leq_P$ dostávame, že aj c pokrýva $d_h\theta$ vzhľadom na I. Teda, existuje substitúcia σ taká, že $c_h\sigma = d_h\theta$. Preto pre nejakú substitúciu σ' platí $c_h\sigma' = d_h$ (σ' sa líši od σ tým, že miesto θ-obrazu z $d_h\theta$ priraďuje θ-vzor, prípadné kolízie so substitúciou za nerelevantné premenné možno riešiť premenovaním).

Napokon, dokážeme, že $\exists(c_t\sigma\theta)$ vyplýva z $P \cup \{d_t\theta\}$. Nech J je interpretácia jazyka $P\cup\{c\}\cup\{d\}\cup\{d\theta\}$, ktorá je minimálnym modelom $P\cup\{d_t\theta\}$. Očividne, d pokrýva $d_h\theta$ vzhľadom na J. Preto aj všeobecnejšie c pokrýva $d_h\theta$ v J. Keďže $c_h\sigma = d_h\theta$, c pokrýva $c_h\sigma\theta$ v J ($d_h\theta$ je základný atóm, preto θ „nezaberie“ na $c_h\sigma$). Teda, $\exists(c_t\sigma\theta)$ je splnená v J (priamo z definície pokrývania). Keďže J je minimálny model $P \cup \{d_t\theta\}$, platí, že $P \cup d_t\theta \models \exists(c_t\sigma\theta)$.
■

Príklad 5.40 Videli sme, že θ-subsumcia nefunguje v príklade 5.1 o kartách. Práve zavedený pojem zovšeobecnenia však funguje.

Máme program P (teóriu Γ z príkladu 5.4):

$$\begin{array}{rcl}
2 \prec 3 \prec \cdots \prec 10 & \prec & j \prec d \prec k \prec e, \\
num(karta(N, Z)) & \leftarrow & N \preceq 10, \\
fig\acute{u}ra(karta(N, Z)) & \leftarrow & N \succ 10, \\
\check{c}ierna(karta(N, Z)) & \leftarrow & Z = l \vee Z = p, \\
\check{c}erven\acute{a}(karta(N, Z)) & \leftarrow & Z = s \vee Z = k, \\
X \preceq Y & \leftarrow & X \prec Y \vee X = Y.
\end{array}$$

O hypotéze $\Phi = odmena(X) \leftarrow \check{c}ierna(X), num(X)$ sa ľahko môžeme presvedčiť, že je zovšeobecnením každého z pozitívnych príkladov

$\Delta^+ = \{$

$$\begin{aligned} odmena(karta(4,l)) &\leftarrow \quad , \\ odmena(karta(7,l)) &\leftarrow \quad , \\ odmena(karta(2,p)) &\leftarrow \} \quad . \end{aligned}$$

Nech δ je ľubovoľný z pozitívnych príkladov. Platnosť $\Phi \leq_P \delta$ preveríme ľahko z $P \cup \{\delta_t\epsilon\} = P \models \exists(\Phi_t\sigma\epsilon) = \exists(\check{c}ierna(X) \wedge num(X))$ pre vhodnú substitúciu σ, napríklad $\{X|karta(4,l)\}$. Pritom pre tú istú substitúciu je $\Phi_h = \delta$.

Navyše, môžeme ukázať, že žiadne integritné obmedzenie
$\Delta^- = \{$

$$\begin{aligned} &\leftarrow \quad odmena(karta(5,s)), \\ &\leftarrow \quad odmena(karta(j,p))\}. \end{aligned}$$

nie je narušené v $P \cup \{\Phi\}$. ■

θ-subsumcia je špeciálnym prípadom vzťahu $\leq_P$: ide o zovšeobecnenie vzhľadom na prázdny program $\emptyset$.

Tvrdenie 5.41 *Nech c a d sú definitné klauzy. Potom $c \leq_\emptyset d$ práve vtedy, keď c θ-subsumuje d.*

Nakoniec ešte návrat k pravidlám indukovania. Ak na indukciu nazeráme ako na inverziu k dedukcii, prirodzené je poňať ju ako inverziu k plnej rezolvencii. Niekoľko všeobecnejších pravidiel uvádza nasledujúci príklad. Ak tieto pravidlá „obrátime“, teda z ich záverov urobíme predpoklady (a z predpokladov závery), dostávame deduktívne pravidlá, o čom sa možno ľahko presvedčiť.

Príklad 5.42 Príklady pravidiel:

absorbcia $\dfrac{q \leftarrow A; p \leftarrow A, B}{q \leftarrow A; p \leftarrow q, B}$,

identifikácia $\dfrac{p \leftarrow A, B; p \leftarrow A, q}{q \leftarrow B; p \leftarrow A, q}$,

intra-konštrukcia $\dfrac{p \leftarrow A, B; p \leftarrow A, C}{q \leftarrow B; p \leftarrow A, q; q \leftarrow C}$,

inter-konštrukcia $\dfrac{p \leftarrow A, B; q \leftarrow A, C}{p \leftarrow r, B; r \leftarrow A; q \leftarrow r, C}$.

■

Pravidlá inter- a intra-konštrukcie zavádzajú nové predikáty. Zavedenie nových predikátov sa v oblasti induktívneho logického programovania nazýva „objavovaním predikátov“.

5.1.3 Indukcia založená na nemonotónnej sémantike

Nazeranie na indukciu ako inverziu dedukcie nemusí vyhovovať v každej situácii. Helft [Hel 89] nevidí dôvod, aby deduktívne vysvetľovanie hralo takú dôležitú úlohu pri štúdiu indukcie. Miesto toho navrhuje považovať indukciu za osobitý druh nemonotónneho odvodzovania dôsledkov z pozorovaní. Indukcia v podmienkach neúplnej informácie hľadá akési podobnosti a na základe nájdených podobností prijíma závery.

Problém indukcie formuluje Helft takto: Nech je daná množina formúl Δ (nerozlišuje sa medzi pozorovaniami a poznaním v pozadí). Treba nájsť množinu formúl Γ (zovšeobecnení množiny Δ) takú, že $\Gamma \models_{ind} \Delta$. Očakáva sa, že Γ reprezentuje pravidelnosti, charakterizujúce Δ. Samozrejme, celý vtip spočíva v definovaní $\models_{ind}$.

Nemonotónna sémantika je vhodná v tých situáciách, kde sa chceme vyvarovať induktívnych hypotéz, ktoré nie sú dostatočne podopreté danými dátami, pozorovaniami. Používa sa pritom istá minimalizácia v štýle predpokladu uzavretého sveta: všetko, čo nie je potvrdené, sa považuje za nepravdivé. Myšlienku demonštruje príklad z [MuR 94]:

Príklad 5.43 Majme $\Gamma = \{vt\acute{a}k(t) \leftarrow, vt\acute{a}k(o) \leftarrow\}$, $\Delta^+ = \{lieta(t) \leftarrow\}$, $\phi = \{lieta(X) \leftarrow vt\acute{a}k(X)\}$.

Vzhľadom na štandardnú sémantiku (budeme pracovať s jej verziou pre minimálne modely, podľa definície 5.13) je hypotéza ϕ v poriadku: $lieta(t)$ nie je pravdivé v minimálnom modeli Γ, ale je pravdivé v minimálnom modeli $\Gamma \cup \{\phi\}$.

Navyše, hypotéza $H_1 = lieta(X) \leftarrow vt\acute{a}k(X)$ je lepšia (vzhľadom na teóriu Γ) ako $H_2 = lieta(X) \leftarrow$: Obe hlavy možno ľahko stotožniť. Preože $telo(H_2)$ je prázdne, platí $\Gamma \cup telo(H_2) = \Gamma$. Z toho dostávame

$$\Gamma \cup telo(H_2) \models \exists(telo(H_1)\sigma\theta).$$

Na druhej strane, prázdne $telo(H_2)$ nemôže vyplývať z $\Gamma \cup telo(H_1)$. Teda $H_1 \leq_\Gamma H_2$, ale neplatí $H_2 \leq_\Gamma H_1$.

Pritom však $\Gamma \cup \{\phi\}$ má aj nežiadúci dôsledok: $lieta(o) \leftarrow$. Problém sa rieši v štýle CWA. Definuje sa $\Delta^- = CWA(\Gamma \cup \Delta^+)$.

Keďže $lieta(o) \leftarrow$ neplatí v $\Gamma \cup \Delta^+$, v Δ^- máme integritné obmedzenie $\leftarrow lieta(o)$. Teda, pri nemonotónnej sémantike nedostaneme ϕ, je v konflikte s integritnými obmedzeniami. ∎

Problémy z predchádzajúceho príkladu rieši v nemonotónnom štýle (s nejakou revíziou) algoritmus 5.44, tabuľka 5.6. Predpokladáme, že z teórie T vyplýva fakt A, ktorý nepovažujeme za pravdivý (nie je splnený v zamýšľanom modeli M). Cieľom algoritmu je revidovať teóriu T na teóriu T' tak, aby $T' \not\models A$. Značenie a predpoklady, ktoré používame:

- C je klauza z T tvaru $A' \leftarrow Goal$, kde $Goal$ je konjunkcia literálov,

Algoritmus 5.44 ([BaM 90])
vstup: *T – množina kláuz, A – základný atóm (pričom $T \models A$, $M \not\models A$, kde M je nejaký zamýšľaný model reprezentujúci daný stav poznania)*
výstup: *T' – revidované T*

for each $C \in T$, **if** $C = A' \leftarrow Goal$, $A'\theta = A$ *pre nejaké* θ
  then while *$Goal$ obsahuje nejaké* $\neg D$ **do**
    $T' := T \cup \{D\theta\}$ *(* blokovanie odvodenia A minimalizovaním negatívnych literálov *)*
  else $Hd := A'; N := q(X_1, \ldots, X_n); Goal' := Goal \cup \{\neg N\}$
    (q je nový predikátový symbol, $\{X_1, \ldots, X_n\}$ je doména substitúcie θ *)*
    $T' := T \setminus \{C\} \cup \{Hd \leftarrow Goal'\} \cup \{N\theta\}$
    (zavádza sa nový fakt, ktorý zabráni odvodeniu A *)*
return T'

Tabuľka 5.6: Algorimus nemonotónnej indukcie

- θ je substitúcia za premenné v C taká, že $A'\theta = A$.

Príklad 5.45 Algoritmus 5.44 zbehneme na príklade 5.43.

$$T = \{vták(t) \leftarrow, vták(o) \leftarrow, lieta(t) \leftarrow, lieta(X) \leftarrow vták(X)\}.$$

Nech zamýšľaná interpretácia je $M = (\{vták(t), vták(o), lieta(t)\}, \{lieta(o)\})$. Teda $lieta(o)$ je nepravdivé v M.

Nezamýšľaný dôsledok $lieta(o)$ sme získali z $\{lieta(X) \leftarrow vták(X); vták(o) \leftarrow\}$ pomocou substitúcie $\theta = \{X|o\}$. Keďže $lieta(X) \leftarrow vták(X)$ neobsahuje v tele negatívny literál:

- q nech je nový predikátový symbol,
- $Hd := lieta(X)$,
- $Bd := vták(X)$,
- $N := q(X)$ (* doménou substitúcie θ je $\{X\}$ *).

$T' :=$

$$\begin{array}{rcl}
\{vták(t) & \leftarrow & , \\
vták(o) & \leftarrow & , \\
lieta(t) & \leftarrow & , \\
lieta(X) & \leftarrow & vták(X), \neg q(X), \\
q(o) & \leftarrow & \},
\end{array}$$

q možno intuitívne čítať napríklad ako *výnimka*. ■

Načrtnime ešte nakoniec, akým smerom by sa dala modifikovať definícia 5.2, aby sme dostali nemonotónnu sémantickú špecifikáciu pre Cn_{ind} (prípadne $\models_{ind}$). Predpokladali sme, že Δ, Γ, Φ sú množiny viet nejakého jazyka J a že sú splnené tieto podmienky:

(1) $\Gamma \not\models \Delta$,

(2) $\Gamma \cup \Delta$ je konzistentná,

(3) $\Gamma \cup \{\Phi\} \models \Delta$,

(4) $\Gamma \cup \{\Phi\}$ je konzistentná.

Myšlienku minimalizácie z algoritmu 5.44, ktorú sme ilustrovali príkladmi 5.43 a 5.45, vyjadríme pomocou preferenčného vyplývania. Predpokladajme, že máme definovanú nejakú reláciu $\prec$ preferencie na triede modelov a na jej základe vzťah preferenčného vyplývania $\models_\prec$ štýlom, ktorý poznáme z kapitol 3 a 4. Potom podmienku (3) môžeme nahradiť podmienkou
(3′) $\Gamma \cup \{\Phi\} \models_\prec \Delta$. Podobne možno zoslabovať aj podmienky (1), (2), prípadne aj (4).

5.1.4 Komentáre

Indukcia, strojové učenie, objavovanie nových poznatkov pomocou počítača sú oblasti intenzívneho výskumu. Je to pochopiteľné – úspech tu môže znamenať významný posun v možnostiach počítačov (i v chápaní ľudskej tvorivosti). Mnoho zaujímavých výsledkov sa dosiahlo pomocou neurónových sietí alebo genetického programovania. Prístup k učeniu, založený na neurónových sieťach alebo genetickom programovaní sa však opiera o „slepé“ procesy.

Je to netransparentný prístup minimálne v nasledujúcom zmysle slova: ak cieľom týchto procesov je dosiahnuť nové poznatky v nejakom odbore, odborník v tomto odbore nemá možnosť priamo vidieť, identifikovať, ako architektúra a detaily systému ovplyvňujú získané výsledky. Napokon, túto možnosť nemá ani informatik. Zmenené správanie systému, zmenu jeho výsledkov možno dosiahnuť iba experimentovaním. Z týchto dôvodov sa tieto techniky niekedy nazývajú aj technikami čiernej skrinky.

Na druhej strane, indukcia a učenie, založené na znalostnej paradigme, t.j. na nejakej báze poznatkov a na nejakom jasne (sémanticky) špecifikovanom spôsobe odvodzovania, umožňuje vidieť do vnútra indukčnej mašinérie. Zvlášť induktívne logické programovanie: báza poznatkov (teória v pozadí, pozitívne a negatívne príklady) obsahuje zrozumiteľné, dobre čitateľné pravidlá. To dovoľuje odborníkom v oblasti aplikácie zasahovať do modifikácie bázy poznatkov a zlepšovať ju. Výsledky, dosiahnuté týmto prístupom, sú povzbudzujúce. Induktívne logické programy už vyprodukovali nové poznatky, ktoré boli publikované v recenzovaných vedeckých časopisoch (ide napríklad o oblasti návrhu liečiv, syntézy proteínov) práve vďaka tomu, že zodpovedajúce logické programy (bázy znalostí) sú zrozumiteľné pre vedcov z príslušnej oblasti, ktorí môžu aktívne prispievať k ich tvorbe a modifikácii. Výsledky boli dosiahnuté všeobecnými, nie účelovo konštruovanými, nástrojmi. Pozri napríklad [MuR 94, Val 95, Val 96, Mug 99a, Mug 99b, SMS 96]. Podobne zaujímavé výsledky pochádzajú z oblasti učenia prirodzeného jazyka, pozri [SMS 96].

Špecifikovali sme indukciu (zovšeobecňovanie) sémantickými prostriedkami. Načrtli sme, ako možno zovšeobecňovať nad štandardnými databázami. Potom sme sa venovali induktívnemu logickému programovaniu. Uviedli sme generický algoritmus, abstrahujúci rôzne spôsoby výpočtu zovšeobecnení. Analyzovali sme základné metódy a pravidlá, používané pri zovšeobecňovaní (θ-subsumciu, zovšeobecnenú subsumciu, inverznú rezolvenciu). Na záver sme si všimli indukciu, založenú na nemonotónnej sémantike.

5.2 Abdukcia

Ľudské poznatky sú nejakým spôsobom zosystematizované. Vďaka tomu dokážeme vysvetľovať naše pozorovania. Špeciálnym prípadom vysvetľovania je diagnostikovanie: máme nejaké pozorovateľné príznaky a úlohou je nájsť diagnózu (stav, ktorý tieto príznaky vysvetlí). V tejto časti sa budeme venovať výpočtovému modelovaniu vysvetľovania. Budeme sa zaujímať o to, ako sa dá špecifikovať sémantickými prostriedkami a ako sa počíta.

Už sme videli, že reprezentácie poznatkov môžeme rozlišovať podľa toho, aké druhy otázok dokážu zodpovedať. Ukážeme si ďalšiu z možností, ako možno zovšeobecniť (zosilniť) odpovedanie na otázky. Najjednoduchší typ otázok vyžaduje odpovede „áno" alebo „nie". Keď si nie sme istí, či naše poznatky o danej situácii sú úplné, očakávame zložitejšie odpovede. Napríklad, môžu to byť podmienečné odpovede: ak je splnená nejaká podmienka, potom áno.

Všimnime si, ako sa dá tento prístup realizovať v logickom programovaní. Aplikácie logického programovania na tvorbu právnych expertných systémov ukázali, že so štandardným spôsobom odpovedania na otázky sa v tejto oblasti nedá vystačiť. V niektorých situáciach text zákonov neposkytuje dostatočné informácie pre definitívne odpovede. Vtedy je rozumné prijímať odpovede tvaru: „ak sú splnené podmienky P, potom možno aplikovať špecifikovanú časť zákona". Začnime opäť príkladom.

Príklad 5.46 Predstavme si krajinu, kde nárok na dôchodok je definovaný takto: má ho človek, ktorý je starší ako 60 rokov a odpracoval viac ako 25 rokov.

$$\begin{aligned} \mathit{n\acute{a}rok\_na\_d\hat{o}chodok}(X) &\leftarrow \mathit{vek}(X), \mathit{odpracovan\acute{e}}(X),\\ \mathit{vek}(X) &\leftarrow \mathit{star\check{s}\acute{i}\_ako}(X, 60),\\ \mathit{odpracovan\acute{e}}(X) &\leftarrow \mathit{viac\_ako}(X, 25). \end{aligned}$$

Predstavme si ďalej, že občan tejto krajiny (napríklad *zuza*) doručil na dôchodkový úrad doklady o svojom veku. Kvôli úplnosti, zareprezentujme tieto splnené podmienky nasledujúcim faktom:

$$\mathit{star\check{s}\acute{i}\_ako}(zuza, 60) \leftarrow .$$

Ak sa *zuza* spýta, či má nárok na dôchodok, správna odpoveď je: ak máte odpracovaných viac ako 25 rokov, potom áno. V jazyku logického programovania otázka a odpoveď sú:

$$\begin{aligned} &\leftarrow \mathit{n\acute{a}rok\_na\_d\hat{o}chodok}(zuza),\\ \mathit{n\acute{a}rok\_na\_d\hat{o}chodok}(zuza) &\leftarrow \mathit{viac\_ako}(zuza, 25). \end{aligned}$$

■

Samozrejme, umožnenie podmienených odpovedí by mohlo znamenať, že nikdy nedostaneme negatívnu odpoveď. Vždy, keď logický program zlyhá pri dokazovaní podmienky P, dá na zodpovedajúcu otázku podmienenú odpoveď „ak P, potom áno“.

Preto je rozumné špecifikovať, ktoré predikátové symboly možno použiť pri podmienenom odpovedaní. Predstavme si teda nejaký zoznam predikátových symbolov, ktoré sa môžu vyskytnúť na podmienkovej strane podmienenej odpovede. Tieto predikátové symboly možno použiť pri tvorbe hypotéz, pri podmienenom odpovedaní.

Ak logickému programu P položíme otázku Q a dostaneme podmienenú odpoveď $Q \leftarrow A$, môžeme A považovať za hypotézu, pomocou ktorej sme schopní vydedukovať Q (v súčinnosti s programom P).

Definícia 5.47 Nech je daný program P a otázka Q. Hovoríme, že $Q \leftarrow A$ je *podmienenou odpoveďou* na Q, ak platí $P \models Q \leftarrow A$. ■

Fakt 5.48 *$Q \leftarrow A$ je podmienenou odpoveďou na otázku Q (pre program P) práve vtedy, keď $P \cup A \models Q$.*

Podmienené odpovede tvaru $Q \leftarrow A$ znamenajú: po doplnení programu P hypotézou A môžeme zodpovedať Q.

Vráťme sa k metafore systematizácie: Potrebujeme sa zorientovať v nejakom systéme poznatkov. Ak sa v ňom orientujeme dostatočne, sme schopní formulovať hypotézy, potrebné na to, aby sme nejaký fakt (udalosť, otázku) adekvátnym spôsobom zaradili do tohto systému. Pod zaradením do systému rozumieme vysvetlenie pomocou poznatkov o danom systéme.

Od smiešne jednoduchého príkladu nároku na dôchodok prejdime k príkladu diagnostikovania. Ten, kto diagnostikuje poruchy nejakého typu systémov, má o ňom určité poznatky. Keď narazí na poruchu konkrétneho systému, pokúša sa o jej vysvetlenie na základe svojich poznatkov o tomto type systémov. Vysvetlením je hypotéza o možnej príčine tejto poruchy. Predpokladaná príčina spolu s poznatkami o príslušnom type systémov umožní vysvetliť, prečo k danej poruche prišlo. Samozrejme, takto získané vysvetlenie niekedy môže byť zavádzajúce. Dostali sme sa však už k pojmu abdukcie.

5.2.1 Základné pojmy

Termín *abdukcia* pochádza od Peirce'a, ktorý ho používal v takýchto kontextoch: Ak máme vety

$$q,$$
$$p \to q,$$

potom na základe abdukcie uzatvárame na p.

Podobne: Ak sme pozorovali d a vieme, že nejaký jav h zapríčiňuje d, abduktívne uzatvárame, že jav h nastal.

V umelej inteligencii sa o abdukcii hovorí v súvislosti s problémom diagnózy (v medicíne či technike – čo zapríčiňuje poruchy v systéme?), plánovania (aké možné akcie potrebujeme navrhnúť, aby sme dosiahli očakávané, požadované výsledky?), porozumenia prirodzeného jazyka (aká možná interpretácia vety zodpovedá jej užitiu v danom kontexte?).

Môžeme pristúpiť k príprave formálneho vymedzenia problému abdukcie (podobnosť na špecifikáciu indukcie je zreteľná; ku vzťahu indukcie a abdukcie sa ešte vrátime). Budeme predpokladať štyri druhy viet:

- nech $\mathcal{E}$ je množina viet (intuitívne, priestor všetkých možných pozorovaní, účinkov, porúch atď.),
- nech je dané $\Delta \subseteq \mathcal{E}$ (intuitívne, množina uskutočnených pozorovaní),
- nech Φ je množina viet, pričom $\mathcal{E} \cap \Phi = \emptyset$ (intuitívne Φ je množina možných hypotéz, príčin, vysvetlení); obvykle sa predpokladá, že vo Φ sú iba základné literály (vysvetľujeme totiž konkrétne pozorovania),
- Γ nech je nejaká teória.

Definícia 5.49 Ak je dané $\Delta \subseteq \mathcal{E}$ a teória Γ, *abduktívnym vysvetlením* množiny viet Δ na základe teórie Γ nazývame minimálnu konečnú množinu $A \subseteq \Phi$ takú, že

- $A \cup \Gamma$ je konzistentná,
- $\Gamma \not\models \Delta$,
- $\Gamma \cup A \models \Delta$.

■

Vo všeobecnosti je možných viac abduktívnych vysvetlení daného Δ. Ak $\{A_1, \ldots, A_n\}$ je množina všetkých abduktívnych vysvetlení pozorovaní Δ, $conj(A_i)$ je konjunkcia všetkých formúl z množiny A_i, potom hovoríme, že

$$conj(A_1) \vee \cdots \vee conj(A_n)$$

je *opatrným* vysvetlením Δ.

Operátor Cn_{abd} je nedeterministický: všimli sme si už alternatívnosť vysvetlení. Môžeme použiť značenie

$$Cn_{abd}^{\Gamma,\Phi}(\Delta) \in \{A | A \subseteq \Phi \wedge \Delta \not\subseteq Cn_{ded}(\Gamma) \wedge \Delta \subseteq Cn_{ded}(A \cup \Gamma)\},$$

kde $A \cup \Gamma$ je konzistentné. Vidíme, že abdukcia je relativizovaná parametrami Γ (teória v pozadí) a Φ (množina možných hypotéz, resp. množina abdukovateľných predikátov).

Príklad 5.50 Ak máme:

$$\begin{aligned}\mathcal{E} &= \{mokr\acute{y}\_tr\acute{a}vnik, mokr\acute{a}\_cesta, \neg mokr\acute{y}\_tr\acute{a}vnik, \neg mokr\acute{a}\_cesta\},\\ \Phi &= \{pr\check{s}\acute{i}, svieti\_slnko, je\_teplo, polieva\_sa\},\end{aligned}$$

$\Gamma = \{$

$$\begin{aligned} pr\check{s}\acute{i} &\rightarrow mokr\acute{a}\_cesta,\\ pr\check{s}\acute{i} &\rightarrow mokr\acute{y}\_tr\acute{a}vnik,\\ svieti\_slnko &\leftrightarrow \neg pr\check{s}\acute{i},\\ polieva\_sa &\rightarrow mokr\acute{y}\_tr\acute{a}vnik,\\ svieti\_slnko \wedge je\_teplo &\rightarrow polieva\_sa\},\end{aligned}$$

hľadajme všetky vysvetlenia pre $\Delta = \{mokr\acute{y}\_tr\acute{a}vnik\}$.

Sú tri možné vysvetlenia, $A_1 = \{$ prší $\}$, $A_2 = \{polieva\_sa\}$, $A_3 = \{svieti\_slnko, je\_teplo\}$.

Ľahko si možno všimnúť istú redundantnosť nájdených vysvetlení. Ak máme vysvetlenie A_3, nepotrebujeme už vysvetlenie A_2 – dá sa totiž z predchádzajúceho odvodiť. Na druhej strane, na vysvetlenie pozorovania Δ nám stačí slabšia hypotéza A_2. ■

V rozumnom zmysle slova môžeme hovoriť o deduktívne slabšej alebo deduktívne silnejšej hypotéze. Obe možnosti sa dajú sformalizovať, napríklad pod názvami nezávislé vysvetlenie a slabé vysvetlenie.

Definícia 5.51 Nech Δ je množina pozorovaní, Γ nejaká teória a $A \neq A'$ sú vysvetlenia pozorovaní Δ vzhľadom na Γ. Potom

- A' je *silnejšie* vysvetlenie ako A, ak $A' \models A$ a $A \not\models A'$,
- A je *nezávislým vysvetlením*, ak neexistuje silnejšie vysvetlenie pozorovaní Δ vzhľadom na Γ.

■

Príklad 5.52 Nie je problém ukázať model, v ktorom je splnené $\Gamma \cup \Delta$, ale nie je splnené $Cn_{abd}(\Gamma \cup \Delta)$. Tým ukážeme nie-korektnosť abdukcie.

Stačí si všimnúť príklad 5.50. Konštruujme interpretáciu $\mathcal{I}$. Nech vysvetlenie A_1 (teda atóm prší) je nepravdivé v $\mathcal{I}$. Nech ďalej pozorovanie Δ (t.j. atóm $mokr\acute{y}\_tr\acute{a}vnik$) je pravdivé v $\mathcal{I}$. Všetky ostatné atómy z príkladu 5.50 nech sú nepravdivé v $\mathcal{I}$. Takto definovaná interpretácia $\mathcal{I}$ je modelom $\Gamma \cup \Delta$, ale nie je modelom abduktívneho vysvetlenia A_1. ■

Priamo na základe definície vidíme, že abdukcia zachováva konzistentnosť. Z tohto faktu a zo zistenej nie-korektnosti Cn_{abd} môžeme uzavrieť, že abdukcia je nemonotónna (na základe tvrdenia 3.30 z kapitoly 3). Nie je to nijakým prekvapením – ak diagnostik na základe nejakej množiny príznakov uzatvára na nejakú diagnózu (ktorá ich spolu so známymi poznatkami vysvetlí), potom zistenie ďalších príznakov môže predchádzajúcu diagnózu modifikovať. Nemonotónnosť abdukcie ilustrujeme aj na príklade:

Príklad 5.53 Vráťme sa ešte raz k príkladu 5.50. Rozšírme množinu pozorovaní: bude mokrý trávnik, ale nebude mokrá cesta. Nech teda

$$\Delta' = \Delta \cup \{\neg mokrá\_cesta\}.$$

Potom už A_1 nie je vysvetlením Δ' (vzhľadom na Γ). Nemôže totiž byť súčasne splnená množina viet

$$\{prší; \neg mokrá\_cesta; prší \to mokrá\_cesta\}.$$

Videli sme však (príklad 5.52), že $\Gamma \cup A_1$ je splnená v interpretácii $\mathcal{I}$. Teda, $\Gamma \cup A_1 \not\models \Delta'$. ∎

5.2.2 Abdukcia ako pokrytie

Abdukciu sme formalizovali v logických termínoch. Transformujme naše úvahy do jednoduchšieho tvaru, budeme narábať s množinami a funkciami.

Definícia 5.54 *Problémom abdukcie* sa rozumie štvorica $< \Delta, \Phi, e, pl >$, kde

- Δ je množina pozorovaní, ktoré treba vysvetliť,
- Φ je množina všetkých hypotéz,
- $e : 2^{\Phi} \longrightarrow 2^{\Delta}$ je zobrazenie množiny všetkých podmnožín Φ do množiny všetkých podmnožín Δ,
- $pl : 2^{\Phi} \longrightarrow L$ je zobrazenie z množiny všetkých podmnožín Φ do ľubovoľnej čiastočne usporiadanej množiny L.

∎

Treba zdôrazniť, že Φ obsahuje nejaké elementárne hypotézy, tie sa môžu skladať v zložené hypotézy. Preto sa aj podmnožiny Φ považujú za hypotézy. Ďalej budeme značiť podmnožiny Φ ako H a prvky Φ ako h (s prípadnými indexami).

Zobrazenie e vlastne zodpovedá teórii Γ z logickej definície, toto zobrazenie totiž opisuje, aké sú vzťahy medzi hypotézami a pozorovaniami. Pre hypotézu (množinu elementárnych hypotéz) H množina pozorovaní

$$e(H) \subseteq \Delta$$

predstavuje súhrn pozorovaní, ktorých vysvetlením je zložená hypotéza H. Samozrejme, e vyjadruje iba veľmi jednoduché poznatky – závislosť pozorovaní na hypotézach. Nemáme prostriedok na vyjadrenie opačnej závislosti, na vyjadrenie alternatívnej alebo negatívnej informácie a podobne.

Zobrazenie pl meria vieryhodnosť hypotéz. Pokiaľ sú $pl(H)$ a $pl(H')$ porovnateľné, potom H nie je vieryhodnejšia ako H' vtedy, keď $pl(H) \leq pl(H')$, kde $\leq$ je čiastočné usporiadanie na L.

Definícia 5.55 H je *úplná*, ak $e(H) = \Delta$. H je *neredundantná*, ak

$$(\neg\exists H' \subset H)\ (e(H) \subseteq e(H')).$$

H je *vysvetlenie*, ak je H úplná a nereduntantná. Vysvetlenie H je *najlepšie* vysvetlenie, ak

$$\neg\exists H'(H' \text{ je vysvetlenie a súčasne } pl(H') > pl(H)).$$

■

Poznámka 5.56 Rovnosť $e(H) = \Delta$ môžeme chápať ako *pokrytie* množiny pozorovaní nejakou množinou hypotéz, na ktorú aplikujeme e. ■

Aby sa vôbec dalo uvažovať o výpočtovej zvládnuteľnosti problému abdukcie, predpokladajme polynomickú vypočítateľnosť funkcií e a pl. Predpokladá sa aj, že veľkosť ich vnútornej reprezentácie je polynomická vzhľadom na súčet kardinalít Δ a Φ.

Budeme abstrahovať od interakcií hypotéz: budeme si všímať iba nezávislé problémy abdukcie.

Definícia 5.57 Problém abdukcie je *nezávislý*, ak platí

$$(\forall H \subseteq \Phi)\ e(H) = \bigcup_{h \in H} e(h).$$

■

Teda, pri nezávislom probléme abdukcie sa pozorovania, vysvetliteľné množinou hypotéz H, presne rovnajú pozorovaniam vysvetliteľným všetkými hypotézami $h \in H$ dohromady. V tabuľke 5.7 je algoritmus, ktorý počíta nezávislý problém abdukcie. Uvedený algoritmus je polynomický.[7]

Tvrdenie 5.59 *Predpokladajme, že je daný nezávislý problém abdukcie. Nech $n = |\Delta| + |\Phi|$. Ak existuje nejaké vysvetlenie pre tento problém, potom existuje $\mathcal{O}(nC_e + n^2)$ algoritmus na nájdenie nejakého vysvetlenia, pričom C_e je (polynomická) zložitosť výpočtu funkcie e.*

[7]⟨ výpočtová zložitosť : dodatok H ⟩. Zložitostné charakteristiky, ale aj celá táto časť o abdukcii ako pokrytí sú podľa [Byl 91]. Tvrdenia uvádzame bez dôkazov. Slúžia iba ako základná informácia.

Algoritmus 5.58
vstup: *nezávislý problém abdukcie*
výstup: *NIL, ak neexistuje vysvetlenie; inak W - zložená hypotéza, ktorá vysvetľuje* Δ, *t.j.* $e(W) = \Delta$, *pričom W je minimálne vzhľadom na* $\subseteq$

(rozhodnutie o existencii vysvetlenia *)*
if $e(\Phi) \neq \Delta$, **then** *Return NIL*

(nájdenie nejakého vysvetlenia *)*
$W := \Phi$
$\forall h \in \Phi$ **if** $e(W \setminus \{h\}) = \Delta$, **then** $W := W - \{h\}$
(cyklus cez všetky potenciálne hypotézy z* Φ **)*
Return W

Tabuľka 5.7: Algoritmus na výpočet nezávislého problému abdukcie

Definícia 5.60 Problém abdukcie je *monotónny*, ak

$$\forall H, H'(H \subseteq H' \rightarrow e(H) \subseteq e(H')).$$

■

Tvrdenie 5.61 *Majme monotónny problém abdukcie. Nech* $n = |\Delta| + |\Phi|$. *Ak existuje riešenie tohto problému, potom existuje* $\mathcal{O}(nC_e + n^2)$ *algoritmus na nájdenie nejakého vysvetlenia.*

Pre monotónny a nezávislý problém abdukcie platí, že nájdenie nejakého vysvetlenia je polynomické, ale nájdenie všetkých vysvetlení je *NP*-úplné. Nie je známa zložitostná charakteristika nájdenia najlepšieho vysvetlenia.

Komplikovanejšie problémy abdukcie (môžeme pripustiť nezlučiteľnosť niektorých hypotéz alebo ich vzájomné interakcie) boli tiež definované v [Byl 91]. Nájdenie nejakého vysvetlenia, nájdenie všetkých vysvetlení a nájdenie najlepšieho vysvetlenia je *NP*-úplné pre tieto problémy.

Ak zavedieme do hry aj vieryhodnosť vysvetlení, situácia je ešte nepríjemnejšia. Polynomický algoritmus existuje iba pre veľmi obmedzenú časť problémov abdukcie, pre usporiadané problémy abdukcie. Problém abdukcie budeme považovať za usporiadaný, ak vieryhodnosti (hodnoty funkcie *pl*) sú lineárne usporiadané na množine všetkých elementárnych hypotéz $h \in \Phi$. Navyše, budeme predpokladať, že porovnateľnosť vieryhodnosti zložených hypotéz je založená na vieryhodnosti elementárnych hypotéz spôsobom, charakterizovaným nižšie zavedeným kritériom. Prejdeme teda k definícii usporiadaného problému abdukcie, a potom k vyjadreniu spomínaného kritéria.

Definícia 5.62 Problém abdukcie je *usporiadaný*, ak pre ľubovoľné hypotézy $h, h' \in \Phi$ platí: ak $h \neq h'$, potom $pl(h) < pl(h')$ alebo $pl(h) > pl(h')$. ■

Vieryhodnoť nemožno priamočiaro preniesť z jednotlivých hypotéz na množiny hypotéz. Tu budeme uvažovať triedu transformácii takéhoto prenosu vieryhodnosti, ktoré spĺňajú nasledujúce kritérium:

Definícia 5.63 (Kritérium propagovania vieryhodnosti) Pre hodnoty $pl(H)$ a $pl(H')$ platí podmienka $pl(H) > pl(H')$ práve vtedy, keď existuje zobrazenie $m : H \longrightarrow H'$, ktoré

- je injekcia,
- $\forall h \in H\ (pl(h) \geq pl(m(h)))$,
- $card(H) = card(H') \Rightarrow \exists h \in H\ (pl(h) > pl(m(h)))$.

■

Podľa tejto definície hypotéza nikdy nemôže byť lepšia ako konkurujúca hypotéza obsahujúca menej elementárnych hypotéz. Menšia hypotéza je lepšia, ak sa podarí m-vnoriť všetky jej elementárne hypotézy do konkurujúcej hypotézy tak, že vieryhodnosť vzoru je väčšia alebo rovnaká ako vieryhodnosť obrazu. V prípade, že kardinalita oboch hypotéz je rovnaká, lepšia množina hypotéz musí byť aspoň v jednom vzore ostro lepšia ako obraz, v ostatných je prinajhoršom rovnako vieryhodná.

Algoritmus 5.64 je malou modifikáciou algoritmu 5.58: cyklus cez hypotézy z Φ beží od najmenej vieryhodnej po najvieryhodnejšiu, pozri tabuľku 5.8. Vzhľadom na to, že pl zobrazuje množiny hypotéz do čiastočne usporiadanej množiny hodnôt, je možných viac najlepších vysvetlení.

Algoritmus je polynomický. Jeho korektnosť (algoritmus nájde najlepšie vysvetlenie):[8] Nech $\Phi = \{h_1, \ldots, h_n\}$, pričom Φ je usporiadaná podľa pl od najmenej vieryhodnej hypotézy po najvieryhodnejšiu. Predpokladajme, že existuje množina hypotéz H, pre ktorú platí $pl(H) > pl(W)$. Predpokladajme ďalej, že existuje h_k také, že $h_k \in W \backslash H$ alebo $h_k \in H \backslash W$ (v opačnom prípade predpoklad $pl(H) > pl(W)$ priamo vedie k sporu). Uvažujme najmenšie k, ktoré má túto vlastnosť.

Nech $h_k \in W \setminus H$. Nech $G = \{h_i \mid i < k\ \wedge\ h_i \in W \cap H\}$. G teda obsahuje hypotézy h_i, ktoré sú menej vieryhodné ako h_k a sú relevantné pre obe vysvetlenia W a H (ak $i < k$, tak $h_i \in W$ práve vtedy, keď $h_i \in H$). Potom $e(G \cup \{h_{k+1}, \ldots, h_n\}) \neq \Delta$ (inak by sme totiž h_k do W nezaradili). Pretože $H \subseteq (G \cup \{h_{k+1}, \ldots, h_n\})$, platí $e(H) \neq \Delta$. To je v spore s predpokladom, že H vysvetľuje Δ.

Predpokladajme teda, že $h_k \in H \setminus W$. Vtedy neexistuje injekcia

$$m : H \longrightarrow W$$

[8] Dôkaz pochádza od B. Brejovej.

Algoritmus 5.64
vstup: *usporiadaný nezávislý problém abdukcie*
výstup: *NIL, ak neexistuje vysvetlenie; v opačnom prípade W - zložená hypotéza, ktorá najlepšie vysvetľuje* Δ*,*

(rozhodnutie o existencii vysvetlenia *)*
if $e(\Phi) \neq \Delta$ **then** *Return NIL*
(nájdenie nejakého najlepšieho vysvetlenia *)*
$W{:=}\Phi$
nasledujúci cyklus prebieha od najmenej vieryhodnej hypotézy $h \in \Phi$ *po najviac vieryhodnú:*
$\forall h \in \Phi$ **if** $e(W \setminus \{h\}) = \Delta,$ **then** $W{:=}\ W \setminus \{h\}$
Return W

Tabuľka 5.8: Algoritmus na výpočet najlepšieho vysvetlenia usporiadaného nezávislého problému abdukcie

požadovaných vlastností: Každé $h_i \in H \cap \{h_1, \ldots, h_{k-1}\}$ sa musí zobrazovať samo na seba, inak by nebola splnená podmienka $pl(m(h_i)) \leq pl(h_i)$. Hodnotu $m(h_k)$ však nemožno rozumne definovať – pre každé $h_m \in W$, kde $m > k$, platí, že $pl(h_m) \geq pl(h_k)$. Teda nemôže platiť $pl(H) > pl(W)$. ∎

Uvedené zložitostné charakteristiky vyznievajú pesimisticky. Realistické problémy sú totiž tie, kde nie sú splnené podmienky nezávislosti, monotónnosti alebo usporiadanosti. Na druhej strane, isté zmiernenie dopadu týchto výsledkov možno odvodiť z nasledujúcich okolností: Možno redukovať počet hypotéz, ktoré sú v hre. Obvykle býva malý počet relevantných hypotéz. Množiny nekompatibilných dvojíc hypotéz tiež nebývajú veľké. Možno vyčleniť istú podtriedu problémov abdukcie, kde nekompatibilita a interakcie bývajú pomerne riedke. Zaujímavým otvoreným problémom je zvládnuteľnosť problémov abdukcie s fixovanými niektorými parametrami. Ďalej, východisko možno hľadať v bádaní v oblasti heuristickej abdukcie. Je tu však čas schladiť priveľké oči pri zmierňujúcej interpretácii získaných zložitostných výsledkov. Analyzovali sme iba veľmi jednoduchý model abdukcie. Uvidíme, že vo všeobecnejšom prípade sú výsledky ešte horšie. Stále však platí: pri každej konkrétnej aplikácii treba vyvažovať potrebnú vyjadrovaciu silu a výpočtovú efektívnosť, hľadať kompromis medzi stratou vo vyjadrovacích možnostiach a ziskom na strane výpočtovej efektívnosti.

5.2.3 Logický prístup k abdukcii

Keď sme zameriavali pozornosť na problém abdukcie, zavedený definíciou 5.54, zdôraznili sme, že teóriu v pozadí reprezentujeme funkciou e, ale jej vyjadrovacie schopnosti sú dosť limitované v porovnaní s tým, čo by sme od teórie v pozadí očakávali. Napriek tomu, zložitostné vlastnosti výpočtov tak definovaného problému abdukcie sú skôr nepríjemné.

Prechádzame k abdukcii, založenej na bohatších vyjadrovacích prostriedkoch. Získame tak všeobecnejší pohľad a všeobecnejšie výsledky a umožní nám to niektoré zaujímavé porovnania i schopnosť vyjadriť bohatšie vzťahy ako funkčnú závislosť množín pozorovaní na množinách hypotéz.

Najprv sa budeme venovať vzťahom abdukcie k defaultom a k negácii ako konečnému zlyhaniu, pozri [EKo 89]. Tieto vzťahy môžu prispieť k pochopeniu abdukcie.

Abdukcia a defaultová logika

Čitateľa iste neprekvapí, že je možné poukázať na vzťah abdukcie a defaultových teórií.

Príklad 5.65 Vrátime sa k príkladu o Nixonovi. Jeho sieťová formulácia bola v príklade 4.92. V tvare defaultovej teórie sme sa s ním stretli v príklade 4.117. Trochu si spomínanú defaultovú teóriu zmodifikujeme, aby sme dostali priamu korešpondenciu s logickým programom (v jeho hlavách sa nemôžu vyskytovať negatívne literály): $E = \{$

$$\begin{array}{rcl}
republik\acute{a}n(nixon) & \leftarrow & , \\
kvaker(nixon) & \leftarrow & , \\
podporuje(X, pacifizmus) & \leftarrow & kvaker(X), norm\acute{a}lny_kvaker(X), \\
podporuje(X, zbrojenie) & \leftarrow & republik\acute{a}n(X), norm\acute{a}lny_republik\acute{a}n(X), \\
 & \leftarrow & podporuje(X, pacifizmus), \\
 & & podporuje(X, zbrojenie)\}.
\end{array}$$

Ak by sme položili otázku $podporuje(nixon, Y)$, dostali by sme podmienené odpovede

$$\begin{array}{rcl}
podporuje(nixon, pacifizmus) & \leftarrow & norm\acute{a}lny_kvaker(nixon), \\
podporuje(nixon, zbrojenie) & \leftarrow & norm\acute{a}lny_republik\acute{a}n(nixon)
\end{array}$$

a teda aj akési abduktívne hypotézy:

$$norm\acute{a}lny\_kvaker(nixon), norm\acute{a}lny\_republik\acute{a}n(nixon).$$

Ide o alternatívne hypotézy. Ak by platili obe, narušili by sme integritné obmedzenie. Tieto hypotézy by sa týkali predikátov, o ktorých nie je informácia, t.j. predikátov $norm\acute{a}lny\_kvaker$ a $norm\acute{a}lny\_republik\acute{a}n$. Opäť vidíme, že generovanie hypotéz, abdukcia, nemonotónnosť súvisia s neúplným poznaním.

Hypotézy, používajúce tieto dva predikáty možno prijímať aj na základe defaultových teórií. Ak by sme z uvedenej množiny viet E vypustili integritné obmedzenie a pridali množinu defaultových pravidiel $D = \{$

$$\frac{: normálny\_republikán(nixon)}{normálny\_republikán(nixon)},$$
$$\frac{: normálny\_pacifista(nixon)}{normálny\_pacifista(nixon)}\},$$

dostali by sme defaultovú teóriu s dvoma extenziami:

$$\begin{aligned} \mathcal{E}_1 &= Cn_{FOL}(E \cup \{normálny\_kvaker(nixon)\}), \\ \mathcal{E}_2 &= Cn_{FOL}(E \cup \{normálny\_republikán(nixon)\}). \end{aligned}$$

Alternatívne hypotézy sú umiestnené v alternatívnych extenziách. Vidno, že je tesný vzťah medzi alternatívnymi hypotézami, ktoré generujeme pomocou abdukcie a extenziami defaultových teórií.

Dá sa povedať, že defaultové pravidlá generujú abduktívne vysvetlenie, aj opačne, abdukciu možno vyjadriť pomocou defaultových teórií. ■

Záver, ktorý sme ilustrovali predchádzajúcim príkladom, vôbec nie je prekvapujúci. Aj usudzovanie o typických prípadoch môžeme formulovať ako abdukciu. Prijímame hypotézy o tom, ako sa veci majú v obvyklých prípadoch a tieto hypotézy nám dovolia vysvetliť tie situácie, keď veci bežia normálne.

Podobne možno ukázať priamu korešpondenciu medzi negáciou ako konečným zlyhaním a abdukciou, čomu sa budeme venovať v nasledujúcej časti. Neskôr uvidíme aj súvislosť abdukcie s revíziami a s metausudzovaním. Abdukcia je skutočne pomerne všeobecným vyjadrením hypotetického usudzovania.

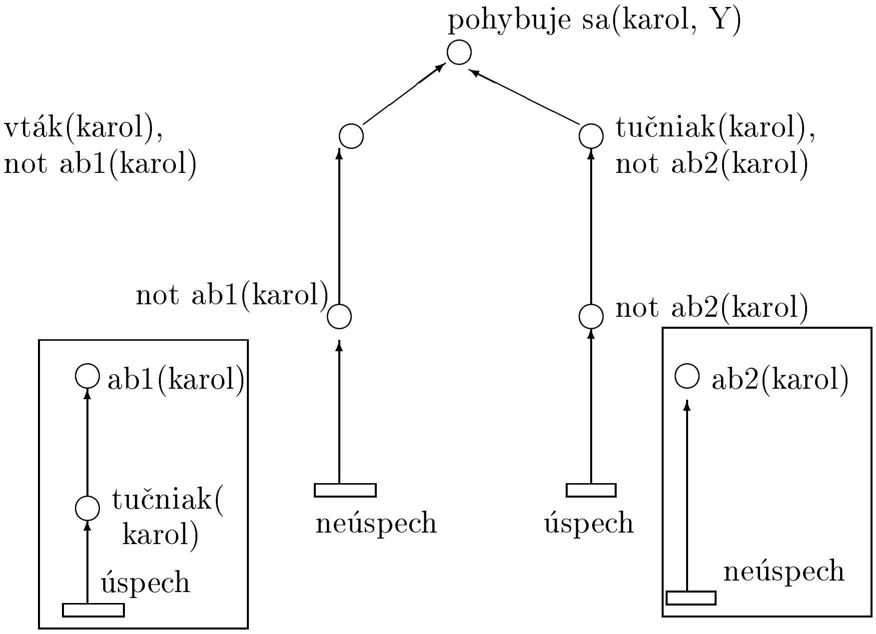
Abdukcia a negácia ako konečné zlyhanie.

Na abdukciu možno nazerať aj optikou negácie ako konečného zlyhania. Tento vzťah je zaujímavý aj z výpočtového hľadiska.

Príklad 5.66 Najprv si uvedieme ďalšiu formalizáciu nášho známeho klasického príkladu o lietajúcich vtákoch a tučniakoch 4.118. Preformulujeme ho podobne ako predchádzajúci príklad, aby sme mohli používať jazyk logického programovania a negáciu (ako konečné zlyhanie):

$$\begin{aligned} pohybuje\_sa(X, lietaním) &\leftarrow vták(X), \neg\, ab_1(X), \\ pohybuje\_sa(X, chodením) &\leftarrow tučniak(X), \neg\, ab_2(X), \\ vták(X) &\leftarrow tučniak(X), \\ tučniak(karol) &\leftarrow\ , \\ ab_1(X) &\leftarrow tučniak(X). \end{aligned}$$

Otázka: pohybuje sa(karol, Y)



Obrázok 5.3: Negácia ako konečné zlyhanie – schéma výpočtu

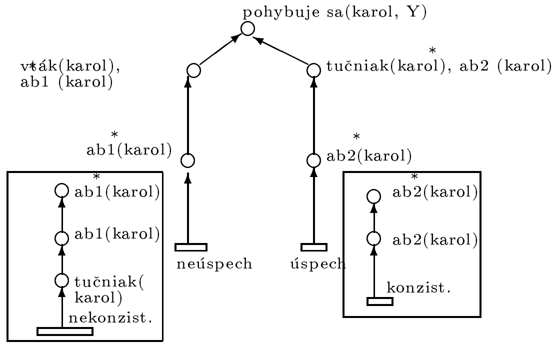
Predpokladajme otázku $\leftarrow$ *pohybuje_sa*$(karol, Y)$.[9] Vyhodnocovanie prvej klauzy vedie k vyhodnocovaniu podcieľa $\neg\, ab_1(karol)$. Negáciu $\neg\, ab_1(karol)$ prijmeme vtedy, keď cieľ $ab_1(karol)$ zlyhá (keď ho vyvrátime). Lenže z nášho programu vyplýva $ab_1(karol)$, preto pôvodný cieľ $\neg\, ab_1(karol)$ nie je splnený. To znamená, že nedostaneme odpoveď *pohybuje_sa*$(karol, lietaním)$. Pri vyhodnocovaní druhej klauzy cieľ $ab_2(X)$ zlyháva (nemáme k dispozícii jeho definíciu), preto prijímame jeho negáciu. Teda na otázku dostávame odpoveď

$$pohybuje\_sa(karol, chodením).$$

(Pozri obrázok 5.3.) ■

Uvedený príklad môžeme ľahko transformovať na abduktívne usudzovanie. Transformácia bude čisto syntaktická. Ciele $\neg\, ab_1(X)$ a $\neg\, ab_2(X)$ prepíšeme pomocou cieľov $ab_1^*(X)$ a $ab_2^*(X)$. K textu programu z príkladu 5.66 dodáme

[9]⟨ logické programovanie : dodatok D ⟩



Obrázok 5.4: Abdukcia a negácia ako konečné zlyhanie

ešte integritné obmedzenia

$$\begin{aligned} &\leftarrow \quad ab_1^*(X), ab_1(X), \\ &\leftarrow \quad ab_2^*(X), ab_2(X), \end{aligned}$$

ktoré hovoria, že žiadny objekt by nemal mať súčasne vlastnosť abnormality (prvej či druhej) a jej „hviezdičkovú verziu", čo môžeme považovať za zvláštne zakódovanie negácie: budeme preverovať konfliktné hypotézy ab_i, ab_i^*.

Preverovanie „hviezdičkovaných" podcieľov takto transformovaného programu prebieha cez integritné obmedzenia. V prvom kroku teda padne rozhodnutie o nekonzistentnosti.

Príklad 5.67 Cieľ $ab_1^*(karol)$ vedie k verdiktu o nekonzistentnosti. Naopak, $ab_2^*(karol)$ k vediktu o konzistentnosti. (Pozri obrázok 5.4.)

Ak na „hviezdičkové" literály nazeráme ako na možné abduktívne hypotézy, záver o nekonzistentnosti hypotézy vedie k zlyhaniu príslušného podcieľa. Nie je totiž konzistentné predpokladať hypotézu $ab_1^*(karol)$, pretože platí konliktná hypotéza. A naopak, záver o konzistentosti vedie k úspechu (je konzistentné prijať hypotézu $ab_2^*(karol)$). ■

Explicitne ide o hypotetické usudzovanie. Preveruje sa konzistentnosť akýchsi hypotéz. Na základe verdiktu o konzistentosti sa tieto hypotézy prijímajú. Potom sa môžu odvodzovať ich dôsledky (s využitím logického programu v pozadí).

Vo všeobecnosti, pre ľubovoľný logický program môžeme zadefinovať nejaký abduktívny rámec a môžeme aj hovoriť o abduktívnom logickom programovaní.

Definícia 5.68 Ak je daný logický program P (s negáciou ako konečným zlyhaním), môžeme k nemu priradiť *abduktívy rámec* (P^*, I, Φ) takto:

(a) P^* získame z P tak, že každý výskyt $\neg\,\phi$ nahradíme výskytom ϕ^*,

(b) $(\leftarrow \phi^*, \phi) \in I$ pre všetky ϕ^* zavedené v (a),

(c) $\Phi = \{p^*|$, kde p^* sú predikáty literálov zavedených v bode (a) $\}$.

Ak je daný abduktívny rámec (P^*, I, Φ), množina hypotéz A je *abduktívne vysvetlenie* otázky Q práve vtedy, keď

- A je množina základných atómov, pričom predikáty z A patria do Φ,
- $P^* \cup A \models Q$,
- $P^* \cup A$ spĺňa I.

■

Abduktívne vysvetlenie môžeme počítať tak ako negáciu (konečné zlyhanie) vtedy, keď za teóriu v pozadí považujeme logický program P, pozorovania chápeme ako otázky (v jazyku programu P), hypotézy ako „hviezdičkové" atómy, ak zavedieme integritné obmedzenie pre každú hypotézu. Čitateľ si iste všimol, že vlastne postupujeme podľa princípu falzifikácie: hypotézu (abduktívne vysvetlenie) akceptujeme, ak nie sme schopní vyvrátiť ju. Vrátime sa k tomu na začiatku kapitoly 6.

Abduktívne logické programovanie

V deväťdesiatych rokoch sa zvýšená pozornosť začala venovať logickému programovaniu ako nástroju na reprezentáciu poznatkov a hypotetického usudzovania. Tento nástroj je zaujímavý minimálne z hľadiska teoretického. Hlavným dôvodom je jeho nemonotónnosť, jeho jednoduchosť a jasná sémantika. Sémantika je charakteristická dôrazom na minimálne modely – z toho plynie, že dôležité sú zamýšľané, programom podopreté významy, nie všetky možné modely. Okrem toho, jazyk logického programovania nie je iba deklaratívnym jazykom logiky, ale aj programovacím jazykom. Preto sa navrhovali rôzne rozšírenia logického programovania. Stretli sme sa už s induktívnym logickým programovaním, teraz prichádza na rad abduktívne logické programovanie, narazíme aj na ďalšie rozšírenia. Uvedené všeobecné dôvody sú argumentom aj pre to, aby sme z hľadiska štúdia (i realizácie) abdukcie považovali prostredie logického programovania za výhodnejšie ako prostredie klasických logických teórií.

Na začiatku tejto časti sme si problém abdukcie motivovali podmieneným odpovedaním na základe logických programov. Táto myšlienka viedla ku vzniku abduktívneho logického programovania, pozri napr. [EKo 89, KaM 90, KKT 93, EiG 95, EGL 97].

V kontexte logického programovania zložitosť výpočtov, potrebných na abduktívne vysvetľovanie, študovali hlavne [EiG 95] a [EGL 97]. Sústredili sa na tri hlavné rozhodovacie problémy: Predpokladajme množinu pozorovaní (otázok) Δ, logický program P, množinu hypotéz Φ a vysvetlenie A:

- Existuje vysvetlenie A pre danú množinu pozorovaní Δ ?
- Je daná hypotéza $h \in \Phi$ *relevantná* (existuje vysvetlenie A také, že $h \in A$) ?
- Je daná hypotéza $h \in \Phi$ *nutná* pre danú množinu pozorovaní Δ (patrí h do každého vysvetlenia A)?

Verzie týchto problémov analyzovali pre rôzne typy logických programov, pre rôzne sémantiky (pozri kapitolu 6) a pre rôzne druhy preferencie definovanej na vysvetleniach. Preferencie môžu byť definované napríklad na základe všeobecnosti vysvetlení alebo na základe inklúzie medzi množinami hypotéz.

Nie je prekvapením, že výsledky sú pesimistické. Zväčša ide o ťažké problémy (až po štvrtú úroveň polynomickej hierarchie) [EiG 95, EGL 97].

Obvykle je pre prístupy k abduktívnemu logickému programovaniu charakteristické, že nejakým spôsobom vychádzajú zo všeobecného logického programovania. Pozornosť si zasluhuje Denecker [Den 94]. Ten považuje abduktívne logické programovanie za deklaratívnu logiku „stojacu na vlastných nohách". Základným pojmom je pojem *definície.* Množina kláuz s predikátom p v hlave je jeho definíciou. Abduktívny logický program je množina definícii nejakej podmnožiny predikátov jazyka. Komplement množiny predikátov definovaných daným programom tvoria *nedefinované* predikáty. O nedefinovaných predikátoch možno v procese usudzovania prijímať ľubovoľné hypotézy. Z tohto hľadiska logické programy sú špeciálnym prípadom abduktívnych logických programov – všetky ich predikáty sa považujú za definované. (Tie, čo nie sú v hlavách, majú prázdnu definíciu a všetky ich inštancie sú nepravdivé.)

Príklad 5.69 Všimnime si program $P = \{p \leftarrow \neg q\}$. Môžeme k nemu pristupovať tak, že q má prázdnu definíciu, preto je nepravdivé (a $\neg q$ pravdivé). Teda: p je pravdivé.

Ak považujeme q za nedefinované, môžeme prijať hypotézu, že je pravdivé, ale môžeme prijať aj opačnú hypotézu, že je nepravdivé. V každom prípade je pravdivostná hodnota p opačná ako pravdivostná hodnota q. ■

Možno povedať, že tento prístup dovoľuje chápať abduktívne logické programy ako reprezentáciu neúplnej informácie a odvodzovanie ich dôsledkov ako usudzovanie s neúplnou informáciou.

Väčšina zaujímavých výsledkov z oblasti abduktívneho logického programovania je založená na súčasných sémantikách negácie v logickom programovaní, s ktorými sa stretneme v kapitole 6.

5.2.4 Komentáre

Všeobecnejší model abdukcie Základná, idealizovaná predstava o abdukcii predpokladá konzistentnosť množiny hypotéz, pozorovaní a teórie. Navyše, vysvetlenie pozorovaní na základe teórie a hypotéz chápe ako dedukciu. Užitočné zovšeobecnenie predstáv o abdukcii dosiahneme tak, že tieto základné predstavy modifikujeme. Môžeme brať do úvahy všeobecnejší pojem modelu, môžeme uvoľniť podmienku konzistentnosti a klasický pojem vyplývania môžeme nahradiť nejakým neštandardným vyplývaním. Takýmto smerom svoje výskumy orientovali napríklad Boutilier s Becherom [Bou 95] alebo Baral [Bar 98]. V mnohých prípadoch je potrebné po získaní nových poznatkov revidovať staršie poznatky. Tento proces sa nazýva asimiláciou poznatkov. Baral [Bar 98] poňal abdukciu ako asimiláciu pozorovaní do teórie, ktorá je v pozadí.

Vlastnosti Cn_{abd} Jedným zo zaujímavých výskumných smerov je štúdium vlastností operátorov odvodenia a revízie. Ide o charakterizáciu týchto operátorov pomocou nejakých abstraktných vlastností. Ako príklad môžeme uviesť kumulatívnosť z kapitoly 3 alebo charakterizáciu operátorov revízie pomocou postulátov v časti 7.1.1. Cieľom takéhoto typu prístupov je dôkladne pochopiť príslušný druh počítania s poznatkami a vymedziť jeho vítané, rozumné vlastnosti.

Pokiaľ ide o abduktívne usudzovanie, treba zaznamenať práce [LoU 96, LoU 97, PeU 98]. V práci [LoU 96] sa podarilo vytvoriť jednotiaci pohľad na revízie a abdukciu. Menej abstraktný prístup, využívajúci abdukciu pri revidovaní prezentoval [Bry 90], pozri aj kapitolu 8.

Indukcia a abdukcia Nakoniec sa zastavíme pri vzťahu indukcie a abdukcie. Videli sme, že sémantické podmienky na oba druhy usudzovania sú definované takmer identicky. Teraz sa dôkladnejšie pozrieme na odlišnosti (ale aj príbuznosti) abdukcie a indukcie.

Formálne podmienky, ktorými sa definuje základná sémantická schéma sú príliš slabé. Už v časti o indukcii sme prijali, že sémantická charakterizácia indukcie nie je postačujúca a venovali sme sa ďalším, upresňujúcim podmienkam v definícii 5.6.

Pripomeňme si jadro základnej schémy. Máme pozorovania Δ, teóriu Γ, hypotézu H. Požadujeme, aby platilo (okrem ďalších podmienok)

$$(*)\ \ \Gamma \cup H \models \Delta.$$

Vyššie sme videli, že keď uvoľníme podmienku vyplývania a konzistentnosti, dostaneme všeobecnejší pojem abdukcie, ktorý zahŕňa aj revízie teórie. Teraz však ostaneme pri klasickom chápaní vzťahu $(*)$.

Rozdiel, ktorý nesporne môžeme vidieť, je v type úloh, v cieľoch oboch usudzovaní. V smere, ako toto usudzovanie prebieha. V prípade abdukcie vychádzame z teórie Γ a usilujeme sa získať z nej hypotézu o danom pozorovaní. V prípade indukcie hypotézu konštruujeme na základe pozorovaní [DiK 96]. Pri abdukcii sa ako hypotézy berú základné atómy, ktoré charakterizujú pozorovaný prípad. Pomocou abdukovateľných predikátov sa zúplňuje poznanie konkrétnych prípadov. Napríklad, pre pacienta, ktorého symptómy poznáme, sa prijíma diagnóza. Pri indukcii sa vytvára definícia predikátu, ktorý ešte nie je definovaný klauzou s neprázdnym telom, ale pozorovania sú fakty s týmto predikátom. (Pripomeňme si príklad 5.4 s definovaním predikátu *odmena* v časti 5.1.) Tu sa v istom zmysle slova teória dopĺňa o novú definíciu (nové pravidlo).

Na druhej strane, na indukciu nestačí nazerať ako na generovanie pravidiel, definícií z pozorovaných prípadov. Indukcia je tým spoľahlivejšia, čím viac sme si istí, že sme vylúčili alternatívne, konfliktné hypotézy. Cieľom je teda blížiť sa k presvedčeniu, že sme z obrovského priestoru možných zovšeobecnení vybrali to najlepšie, že naša množina pozorovaní nebola vybraná tak, že nás zaviedla nesprávnym smerom. Samozrejme, to nie je možné bez hľadania protipríkladov, bez pokusov vysvetľovať čo najväčšie množstvo faktov pomocou získaného zovšeobecnenia (a teórie v pozadí). To znamená, ide o to, aby sa indukcia obohatila o abdukciu (spojila s ňou). Je možné odmietnuť také vysvetlenia, ktorých niektoré dôsledky neboli pozorované. Námety na integráciu procesov (a implementácií) abdukcie a indukcie sú jedným z dôležitých produktov analyzovania ich rozdielov a príbuzností.

Napríklad – väčšina existujúcich systémov induktívneho logického programovania implicitne predpokladá, že daná teória v pozadí Γ je postačujúca na konštrukciu hypotézy z daných pozorovaní. Samozrejme, tento predpoklad je príliš silný na reálne aplikácie. Preto je dôležité, aby systém, ktorý hľadá induktívne zovšeobecnenia, bol schopný v prípade potreby rozširovať teóriu v pozadí, prijímať nové hypotézy. To znamená, aby bol schopný kombinovať (integrovať) indukciu a abdukciu [KaK 97].

Vzťah abdukcie a indukcie, možnosti ich integrácie sú predmetom intenzívneho výskumu. Tomuto komplexu problémov boli venované workshopy na európskych konferenciách o umelej inteligencii ECAI'96 a ECAI'98 a na svetovej konferencii IJCAI'97. Texty z týchto workshopov možno nájsť na adrese `http://www.cs.bris.ac.uk/~flach/abdind`. Najnovším produktom výskumu je kniha [FlK 00].

V tejto časti sme si predstavili abdukciu a zaoberali sme sa jej výpočtovou efektívnosťou a jej vzťahom k iným formám hypotetického uvažovania. Napriek prvému dojmu, že abdukcia je všeobecnejšia a silnejšia ako iné doteraz študované prípady hypotetického uvažovania, sa ukázalo, že je redukovateľná na výpočet negácie ako konečného zlyhania.

Venovali sme sa abdukcii, založenej na logickom formalizme a aj abdukcii, chápanej ako pokrytie množiny pozorovaní nejakou množinou hypotéz. Podrobnejšie sme sa zastavili pri výpočtových vlastnostiach takto poňatej abdukcie. Pre problém abdukcie, ktorý spĺňa veľmi obmedzujúce predpoklady (nezávislý resp. monotónny) existuje polynomický algoritmus, ktorý nájde nejaké vysvetlenie. Problém nájdenia všetkých vysvetlení je NP-úplný aj pre tento najjednoduchší prípad.

Ak do hry vstúpia vierohodnosti jednotlivých hypotéz, je najjednoduchším prípadom usporiadaný problém abdukcie. Ak je splnené kritérium propagácie vieryhodností jednoduchých hypotéz na vieryhodnosti zložených hypotéz, existuje polynomický algoritmus na nájdenie nejakého najvieryhodnejšieho vysvetlenia. Všetky ostatné prípady sú NP-úplné.

Kapitola 6

Sémantika falzifikácie

Po úvodnej a všeobecnej charakterizácii hypotetického usudzovania v kapitole 3 sme sa v kapitolách 4 a 5 zaoberali niektorými jeho príkladmi. Teraz je čas prejsť opäť do všeobecnejšej polohy. Základným problémom bude sémantická charakterizácia hypotetického usudzovania. K tomuto problému sa priblížime cez problém negácie v logickom programovaní. Tento prípad je reprezentatívny a charakteristický pre hypotetické uvažovanie vo všeobecnosti. Nejaká negácia sa v logickom programovaní prijíma totiž vtedy, keď je vyvrátený jej opak („komplementárna hypotéza"). Preto hovoríme o sémantike falzifikácie (možno nadsadene). Po sémantickej charakterizácii negácie v logickom programovaní sa dostaneme k jej prekvapujúcim vzťahom k dobre známym a často študovaným formalizmom nemonotónnej inferencie. To posilní našu dôveru v sémantiku negácie v logickom programovaní ako dobrého kandidáta na položenie základov pre sémantickú charakterizáciu hypotetického usudzovania.

V časti 5.2 sme porovnali abdukciu s niektorými prípadmi hypotetického usudzovania. Toto porovnanie ukázalo ich blízkosť, až vzájomnú redukovateľnosť. Porovnanie s negáciou ako konečným zlyhaním nás vrátilo k prvému typu hypotetického usudzovania, s ktorým sme sa zoznámili. Tento návrat je zaujímavý. Pôvodne sme príklad negácie[1] chápali ako elementárny, skôr kuriózny prípad hypotetického usudzovania. Teraz sa ukázalo, že všeobecná formulácia hypotetického usudzovania (v tvare abdukcie) je v istom zmysle slova redukovateľná na štandardný výpočet negácie v logickom programovaní. To je dobrý dôvod, aby sme sa vrátili k analýze tejto negácie.

[1] V tejto kapitole budeme kvôli pohodliu často hovoriť iba o negácii, myslieť tým však budeme na negáciu ako konečné zlyhanie. Prípadné odchýlky od tejto dohody budú výrazne odlíšené.

Aby sme mohli s pokojným svedomím hovoriť o negácii ako konečnom zlyhaní a súčasne si namýšľať, že hovoríme aj o falzifikácii, pripomenieme si intuíciu negácie ako konečného zlyhania a uvedieme predstavu o falzifikácii.

Negácia ako konečné zlyhanie je výpočtové pravidlo, ktoré k úlohe dokázať základný atóm $\neg A$ pristupuje tak, že sa pokúša dokázať A. Ak v konečnom čase zlyhá jeho pokus ukázať, že A platí, prijíma hypotézu $\neg A$. V opačnom prípade hypotézu $\neg A$ odmieta.

Je to v súlade s Popperovým pohľadom [Pop 59] na hypotetické usudzovanie: hypotézu prijímame dovtedy, kým nie je vyvrátená (falzifikovaná). Pokiaľ ide o negáciu ako konečné zlyhanie: prijímame hypotézu tvaru $\neg A$, ak nie sme schopní vyvrátiť ju tým, že ukážeme platnosť A.

Pod *falzifikáciou* hypotézy H budeme rozumieť demonštráciu toho, že je nepravdivá. Hypotézu môžeme akceptovať dovtedy, kým nie je falzifikovaná.

Naším cieľom je podať sémantickú charakterizáciu negácie, založenej na výpočtovom pravidle konečného zlyhania. (Azda stojí za upozornenie, že hľadáme sémantickú charakterizáciu výpočtovej procedúry.) Domnievam sa, že sa súčasne pokúšame aj o sémantickú charakterizáciu akéhokoľvek procesu, ktorý akceptovanie nejakej hypotézy podmieňuje tým, aby nebola falzifikovaná. Jazykový rámec, v ktorom sa pohybujeme, je logické programovanie. Hypotézy reprezentujeme ako negácie. Pokiaľ dokážeme všeobecnejšie vyjadrené poznatky a hypotézy redukovať do tohto tvaru, mohli by sme predložené sémantiky považovať aj za sémantiky falzifikácie.

Aj keď sa budeme zaujímať o sémantiku negácie, najprv si zhrnieme základné fakty o sémantike definitných logických programov (pripomeňme si, že ide o programy bez negácie a bez integritných obmedzení).[2]

Definícia 6.1 Definitný logický program je množina kláuz typu

$$A \leftarrow A_1, \ldots, A_k,$$

kde $k \geq 0$, A, A_i sú atómy. ■

Podľa dohody z kapitoly 2 si budeme všímať iba herbrandovské interpretácie, t.j. interpretácie skonštruované z predikátových symbolov, konštánt a funkčných symbolov jazyka J. Interpretácia je dvojica (T, F) množín základných atómov uvažovaného jazyka J. Ak sú T a F disjunktné a $T \cup F = B_P$, ide o 2-interpretáciu a môžeme ju reprezentovať aj samotným T (s tým, že $F = B_P \setminus T$). Model logického programu P je taká interpretácia, ktorá spĺňa P. Za sémantickú charakterizáciu definitných logických programov sa považujú minimálne modely (vzhľadom na inklúziu). Idea v pozadí je asi takáto: význam programu je rozumné stotožniť s tou minimálnou množinou základných atómov, ktorých pravdivosť je vynútená predpokladom o pravdivosti daného programu.

[2]⟨ dodatok D ⟩

Tvrdenie 6.2 *Nech P je definitný logický program.*

- *Zaručene existuje model P.*
- *Prienik dvoch modelov programu P je opäť jeho modelom.*
- *Zaručene existuje jediný minimálny (vzhľadom na inklúziu) model programu P.*

Idea dôkazu:
a) Stačí interpretovať hlavy kláuz z P tak, aby boli pravdivé. (Že to nejde vždy „urobiť na jeden prechod“ ilustruje nasledujúci príklad:

$$P = \{p(a) \leftarrow; p(f(X)) \leftarrow p(X)\}.$$

V každom prípade však B_P je modelom P.)
b) Ak telo klauzy $A \leftarrow A_1, \ldots, A_k \in P$ je splnené v prieniku modelov $M_1 \cap M_2$, nutne musí byť v tomto prieniku splnená aj jej hlava. Prázdne telo je splnené v každom modeli.
c) Na základe b) sa ukáže, že M_P je modelom P. Je zrejmé, že neexistuje model, ktorý je vlastnou podmnožinou M_P. (Jediným) minimálnym modelom P je $M_P = \bigcap\{M : M \text{ je modelom } P\}$. ■

Minimálny model M_P má túto pozoruhodnú vlastnosť (aj ona potvrdzuje, že rozhodnutie považovať M_P za charakterizáciu významu P je rozumné):

Tvrdenie 6.3 $M_P = \{A : P \models A, A$ *je základný atóm* $\}$.

Idea dôkazu: Majme základný atóm A. A je pravdivý v minimálnom modeli P práve vtedy, keď je pravdivý v každom modeli P. Teda $P \models A$ práve vtedy, keď $A \in M_P$. ■

Teraz si pripomenieme dôležitú konštrukciu T_P-iterácie.

Definíciu T_P si (spolu s ďalšími definíciami) uvedieme znovu:

Definícia 6.4 T_P je zobrazenie z herbrandovských interpretácií do herbrandovských interpretácií, pričom $T_P(I) =$

$$\{A \in B_P : A \leftarrow A_1, \ldots, A_n \in Ground(P) \land \{A_1, \ldots, A_n\} \subseteq I\},$$

kde $Ground(P)$ je množina základných inštancií programu P. ■

Tvrdenie 6.5 *T_P je monotónny: ak $I \subseteq J$, potom $T_P(I) \subseteq T_P(J)$.*

Dôkaz: Nech platí, že $A \in T_P(I)$. To znamená, že $A \leftarrow B_1, \ldots, B_n \in Ground(P)$ a $I \models B_1 \land \cdots \land B_n$. Potom nutne platí, že $B_1 \land \cdots \land B_n$ je splnená v modeli J a z toho $A \in T_P(J)$. ■

Definícia 6.6 Mocniny T_P-operátora sa definujú takto:

$$\begin{aligned} T_P \uparrow 0(I) &= I, \\ T_P \uparrow (n+1)(I) &= T_P(T_P \uparrow n(I)) \cup T_P \uparrow n(I), \\ T_P \uparrow \omega(I) &= \bigcup_{n=0}^{\omega} T_P \uparrow n(I). \end{aligned}$$

■

Definícia 6.7 I je *pevným* bodom operátora Γ práve vtedy, keď $\Gamma(I) = I$. I je *minimálny* pevný bod operátora Γ, ak neexistuje $J \subset I$, ktorá by bola jeho pevným bodom. I je *najmenší* pevný bod operátora Γ, ak pre každý jeho pevný bod J platí $I \subseteq J$. ■

Tvrdenie 6.8 *Ak P je definitný logický program, potom*

- *najmenší pevný bod operátora T_P je $T_P \uparrow \omega(\emptyset)$,*
- *$T_P \uparrow \omega(\emptyset) = M_P$, kde M_P je minimálny herbrandovský model P.*

Ukázalo sa, že sémantické charakterizácie logických programov pomocou T_P-iterácie a pomocou minimálnych modelov sa zhodujú. T_P-iterácia zodpovedá výpočtu logického programu zdola nahor, od faktov k hlavám všetkých ostatných pravidiel iteratívne dovtedy, kým je čo rátať.

Túto konštrukciu sme intuitívne využívali aj pri výpočtoch v hierarchických sieťach a bude sa nám opakovane vynárať aj v iných kontextoch. Videli sme, že sémantická charakterizácia logických programov, ktorú poskytuje táto konštrukcia, sa presne zhoduje s charakterizáciou pomocou minimálnych modelov.

Nie je naším cieľom venovať sa teraz vzťahu výpočtových (syntaktických) procedúr a sémantiky. Upozorňujeme iba na korektnosť a úplnosť výpočtov definitných logických programov vzhľadom na tu popísanú sémantiku.

Doposiaľ sme abstrahovali od negácie. Získali sme tak sémantiku s krásnymi vlastnosťami – jediný minimálny model, monotónny operátor T_P s najmenším pevným bodom, zhoda najmenšieho pevného bodu T_P-iterácie s minimálnym modelom.

Všetky tieto vlastnosti sa strácajú, keď vstupuje na scénu negácia. Môžeme dodať, že sa strácajú vtedy, keď začíname mať do činenia s hypotetickým usudzovaním.

Definícia 6.9 Množina kláuz tvaru $A \leftarrow L_1, \ldots, L_k$, kde L_i sú literály a A atóm, sa nazýva *všeobecným* logickým programom. ■

Modifikujme definíciu T_P (doposiaľ sme ju mali zavedenú iba pre prípad definitných programov).

Definícia 6.10 Nech P je všeobecný logický program a I jeho interpretácia. Potom

$$\begin{aligned} T_P(I) \;\; = \;\; & \{A \in B_P : A \leftarrow A_1, \ldots, A_n, \neg A_{n+1}, \ldots, \neg A_{n+m} \in Ground(P) \wedge \\ & \wedge \{A_1, \ldots, A_n\} \subseteq I \wedge \{A_{n+1}, \ldots, A_{n+m}\} \cap I = \emptyset\}. \end{aligned}$$

■

Nasledujúci príklad nám ilustruje ťažkosti so sémantickou charakterizáciou všeobecných programov.

Príklad 6.11 Uvažujme program $P = \{p \leftarrow \neg q\}$. Ten má dva minimálne modely: $M_1 = \{p\}$ a $M_2 = \{q\}$. Prienik týchto dvoch modelov už nie je modelom uvažovaného programu. Vidíme teda, že všeobecné programy nemožno sémanticky charakterizovať jediným minimálnym modelom.

T_P nie je monotónny pre ľubovoľný všeobecný program P : V uvažovanom programe je $T_P(\emptyset) = \{p\}$, ale $T_P(\{q\}) = \emptyset$.

Možnosť písať programy ako $p \leftarrow \neg p$ alebo – trochu menej priamočiaro – $p \leftarrow r, r \leftarrow q, q \leftarrow \neg p$ naznačuje, že vytvorenie uspokojujúcej sémantiky bude asi komplikované. ■

Videli sme, že so sémantickou charakterizáciou negácie i s jej výpočtovým správaním je spätých viac závažných problémov. Postupne sa niektorým budeme venovať.[3]

Prvým návrhom na sémantiku negácie ako konečného zlyhania bolo zúplnenie. Veľmi hrubo si ho môžeme charakterizovať ako nahradenie implikácií ekvivalenciami.

Príklad 6.12 Zúplnením $p \leftarrow \neg p$ je $p \leftrightarrow \neg p$. Zúplnením $p \leftarrow q, p \leftarrow r$ je $p \leftrightarrow (q \vee r)$. Ak však máme program

$$\begin{aligned} p(Y) &\leftarrow q(Y), \neg r(a, Y), \\ p(f(Z)) &\leftarrow \neg q(Z), \\ p(b) &\leftarrow \;, \end{aligned}$$

musíme urobiť nejaké kroky navyše: musíme nájsť tvar hlavy, ktorý by bol unifikovateľný so všetkými uvedenými. Použijeme premennú nevyskytujúcu sa v pôvodných klauzách. Môže to byť napríklad X. Primerane treba upraviť aj disjunkciu na opačnej strane ekvivalencie:

$$p(X) \leftrightarrow \exists Y\ (X = Y \wedge q(Y) \wedge \neg r(a, Y)) \vee \exists Z\ (X = f(Z) \wedge \neg q(Z)) \vee X = b.$$

Zmysel tejto úpravy je zrejmý: predikát p sa definuje troma klauzami. Pravá strana každej z nich je jedným z disjunktov tvorenej ekvivalencie. Aby

[3] Podrobnejšie sa čitateľ s problematikou negácie v logickom programovaní môže oboznámiť v [Llo 87, She 87, ABo 94].

sme zachovali pôvodný zmysel po zavedení novej premennej X, musíme stotožniť X s Y, s $f(Z)$ a s b. V prípade výskytu premenných Y a Z musíme existenčným kvantifikátorom požadovať existenciu Y a Z takých, čo spĺňajú uvedené rovnosti.

Keďže sa na pravých stranách objavuje nový predikát – identita, potrebné je ešte zaviesť axiómy pre identitu.

Ak nejaký predikát p (s aritou k) z programu sa nevyskytuje v žiadnej hlave tohto programu, do zúplnenia sa dostane $\neg p(X_1, \ldots, X_k)$. ■

Nie je ťažké vidieť, že existujú nekonzistentné zúplnenia. Ďalej, zúplnenie a predpoklad uzavretého sveta sa vzhľadom na nekonzistentnosť nepríjemne líšia.

Príklad 6.13

- Pre program $p \leftarrow \neg p$ máme nekonzistentné zúplnenie, ale CWA nevedie k nekonzistentnosti.
- Naopak, pre $p \leftarrow \neg q$ je CWA nekonzistentné, ale zúplnenie nie.
- Ak uvažujeme program $P = \{p \leftarrow q; q \leftarrow \neg p; q \leftarrow q\}$, zúplnenie aj CWA sú konzistentné (zúplnenie je $p \leftrightarrow q, q \leftrightarrow (\neg p \vee q)$, ďalej ľahko možno preveriť, že $P \models p$, ale $P \not\models q$, preto podľa CWA dostávame $\neg q$); zjednotenie CWA a zúplnenia je však nekonzistentné.

■

Clark ukázal, že výpočet negácie ako konečného zlyhania (*SLDNF*-rezolvenciu) možno sémanticky charakterizovať pomocou zúplnenia. SLDNF-rezolvencia je korektná a úplná vzhľadom na zúplnenie:

Tvrdenie 6.14 (Korektnosť a úplnosť SLDNF) *Nech P je všeobecný logický program a G je otázka. Potom SLDNF-rezolvencia $P \cup \{G\}$ zlyhá v konečnom čase*[4] *práve vtedy, keď $comp(P) \models G$, pričom $comp(P)$ je zúplnenie P.*

Uvedené príklady však ukazujú, že napriek tejto charakterizácii výpočet negácie pomocou konečného zlyhania vedie k niektorým problémom (s intuíciami, s adekvátnosťou).

6.1 Stratifikácia

Ako prvý problém si podrobnejšie všimnime rekurziu „cez negáciu“, ktorej najzvrátenejším prípadom je zrejme klauza tvaru $p \leftarrow \neg p$. Riešenie tohto

[4]Presný výklad pomocou dobre definovaných pojmov možno nájsť v [Llo 87].

problému ponúkli [ABW 87] v stratifikačnej sémantike. Jej hlavnou ideou je isté ohraničenie používania negácie a definovanie jasnej sémantiky pre takto syntakticky ohraničené logické programy. Intuitívne, akceptovateľné sú iba také logické programy, ktoré možno rozčleniť do vrstiev a na vrstvách zaviesť usporiadanie tak, aby predikáty, ktorých negácie sa vyskytujú v telách kláuz určitej vrstvy boli definované v niektorej z predchádzajúcich vrstiev. Ide o to, že keď používame negáciu, mali by sme negovať predikát, ktorého význam je už uzavretý, relácia, ktorá ten význam tvorí, je už vypočítaná.

Definícia 6.15 *Definícia* predikátového symbolu p v logickom programe P je množina kláuz z P, ktorá obsahuje p v hlave. Predikátový symbol p sa *vyskytuje pozitívne (negatívne)* v klauze C, ak sa vykytuje v jej tele v pozitívnom (negatívnom) literáli. ■

Definícia 6.16 Logický program P je *stratifikovaný* práve vtedy, keď existuje rozklad P na $(P_1, \ldots, P_k)$, t.j. $P = P_1 \cup \cdots \cup P_k$ a $P_i \cap P_j = \emptyset$ pre $i \neq j$, pričom:

- ak sa predikátový symbol p vyskytuje pozitívne v klauze z P_i, potom jeho definícia je vo vrstve P_j, kde $j \leq i$,
- ak sa vyskytuje negatívne, $j < i$.

Uvedený rozklad nazývame *stratifikáciou* programu P. ■

Definícia 6.17 *Závislostný graf* programu P je orientovaný graf $G = (N, E)$, kde množina vrcholov N zodpovedá predikátovým symbolom z P a orientovaná hrana (p, q) vedie od vrcholu p k vrcholu q, ak existuje klauza v C taká, že p je predikátový symbol z jej hlavy a q predikátový symbol z jej tela. Hrany sú ohodnotené symbolmi $+$ $(-)$, ak sa zodpovedajúci predikátový symbol vyskytuje v tele pozitívne (negatívne). ■

Orientácia hrany ide od predikátu k tomu predikátu, ktorý treba mať vypočítaný skôr (resp. nie neskôr).

Tvrdenie 6.18 *Program P je stratifikovaný práve vtedy, keď v jeho závislostnom grafe neexistuje cyklus obsahujúci negatívnu hranu.*

Teraz si ukážeme, že stratifikované programy majú modely s uspokojujúcimi vlastnosťami.

Nech P je stratifikovaný program, $(P_1, \ldots, P_k)$ jeho stratifikácia. Jeho minimálny model budeme konštruovať iteratívne.

$$
\begin{aligned}
M_1 &= T_{P_1} \uparrow \omega(\emptyset), \\
M_2 &= T_{P_2} \uparrow \omega(M_1), \\
&\cdots, \\
M_k &= T_{P_k} \uparrow \omega(M_{k-1}).
\end{aligned}
$$

T_{P_i} sú monotónne „po vrstvách“:

Tvrdenie 6.19 *Nech $Def_{P_{i+1}} \subseteq B_P$ je množina všetkých základných inštancií hláv vrstvy P_{i+1}. Ak $M_i \subseteq I \subseteq J \subseteq M_i \cup Def_{P_{i+1}}$, potom $T_{P_{i+1}}(I) \subseteq T_{P_{i+1}}(J)$.*

Idea dôkazu: Nech $A \leftarrow G$ je inštancia niektorej z kláuz vrstvy P_{i+1}, pričom G je konjunkcia literálov jej tela. Stačí zvážiť prípad, že $I \models G$, ale $J \not\models G$ (potom $A \in T_{P_{i+1}}(I)$, ale $A \notin T_{P_{i+1}}(J)$). Z $I \subseteq J$ dostávame, že všetky pozitívne literály z G musia byť splnené aj v J. Preto $A \notin T_{P_{i+1}}(J)$ môže platiť iba v tom prípade, keď nejaký negatívny literál $\neg L_j$ z G, ktorý je splnený v I, nie je splnený v J. J môže – na rozdiel od I – obsahovať L_j. Ale J sa podľa predpokladu môže od I líšiť iba o predikáty „definované vo vrstve P_{i+1}“. Medzi tie však nepatria predikáty negatívnych literálov z G, pretože uvažujeme stratifikovaný program. Predikát z L_j musel byť definovaný najneskôr v P_i. ■

Definícia 6.20 Model M programu P je *podopretý*, ak pre každý základný atóm A platí: $A \in M$ práve vtedy, keď existuje klauza $A \leftarrow L_1, \ldots .L_k \in Ground(P)$ taká, že $M \models \{L_1, \ldots, L_k\}$. ■

Tvrdenie 6.21 *Nech P je program so stratifikáciou $(P_1, \ldots, P_k)$ a $M_P = M_k = T_{P_k} \uparrow \omega(M_{k-1})$. Potom minimálnym a podopretým modelom programu P je M_P. Ak $(P_1, \ldots, P_k)$ a $(P_1^*, \ldots, P_m^*)$ sú dve rôzne stratifikácie programu P, iteráciou v oboch sa získa ten istý M_P, teda minimálny model nezávisí na stratifikácii.*

M_P sa nazýva štandardným modelom stratifikovaného programu P.

Príklad 6.22 Všimnime si teraz program P:

$$\begin{array}{rcl} even(0) & \leftarrow & , \\ even(s(X)) & \leftarrow & \neg even(X), \end{array}$$

kde $s(X)$ je nasledovník X. Tento program nie je stratifikovaný, ale jeho zmysel je celkom jasný a rozumný. ■

Zavedieme sémantiku, ktorá tento zmysel postihne. Najprv syntaktický pojem lokálne stratifikovaných programov (program uvedený v príklade 6.22 pod ne spadá). Potom pojem prefektného modelu. Uvidíme, že sémantika založená na perfektných modeloch zovšeobecňuje sémantiku stratifikovaných programov a charakterizuje lokálne stratifikované programy.

Pri stratifikovaných programoch sme do vrstiev rozložili klauzy programu (ekvivalentný efekt sa dá dosiahnuť rozvrstvením množiny predikátov). Teraz objektom rozvrstvenia budú základné atómy:

Definícia 6.23 Majme funkciu f definovanú na B_P a jej hodnotami nech sú spočítateľné ordinály.[5] Rozšírenie na negatívne literály: Ak $\phi \in B_P$, potom $f(\neg\phi) = f(\phi) + 1$. Túto funkciu môžeme nazvať *lokálnou stratifikáciou.*

Budeme hovoriť, že základná inštancia klauzy $A \leftarrow L_1, \ldots, L_n$ je *lokálne stratifikovaná* vzhľadom na f, ak

- $f(A) \geq f(L_i)$ pre každý pozitívny literál L_i,
- $f(A) > f(L_i)$ pre každý negatívny literál L_i.

Program je *lokálne stratifikovaný* vzhľadom na f, ak je každá inštancia každej jeho klauzy lokálne stratifikovaná vzhľadom na f.

Program je lokálne stratifikovaný, ak existuje lokálna stratifikácia, vzhľadom na ktorú je lokálne stratifikovaný. ∎

Poznámka 6.24 Ľahko môžeme zaviesť závislostný graf základných atómov nejakého programu P. P je lokálne stratifikovaný práve vtedy, keď jeho závislostný graf nemá záporne ohodnotené cykly.

Lokálne stratifikovaný program P môžeme definovať aj rozkladom jeho bázy B_P do vrstiev $\{H_1, \ldots, H_n, \ldots\}$, kde H_i sú množiny základných atómov. ∎

Teraz si zavedieme perfektné modely. Začneme však príkladom.

Príklad 6.25 Uvažujme dva z mnohých modelov programu P z príkladu 6.22.

$$M = \{even(0), even(2), \ldots, even(2n), \ldots\} = \{even(2i) : i \in Nat\},$$

kde *Nat* je množina prirodzených čísel. Príkladom modelu, ktorý nevystihuje zamýšľaný zmysel uvedeného programu, je

$$\begin{aligned} N &= \{even(0), even(1), even(3), \ldots\} \\ &= \{even(0)\} \cup \{even(2i-1) : i \in Nat \wedge i \geq 1\}. \end{aligned}$$

N nie je jediný nezamýšľaný model: Modelom programu P je ľubovoľná interpretácia (T, F), kde T obsahuje $even(0)$, pre nejaké k a nejaké i obsahuje všetky atómy $even(k), \ldots, even(k+i)$. Okrem toho, pre každé $even(j) \in F$ platí $even(j+1) \in T$.

Naším cieľom je vyjadriť, že model M preferujeme viac ako model N. Využijeme pritom funkciu f definovanú takto: Môžeme položiť $f(even(0)) = 0$. Podľa definície 6.23 môžeme ďalej f konštruovať tak, že pre každú inštanciu bude $f(even(s(X))) = f(even(X)) + 1$. Preto

$$f(even(1)) < f(even(2)), f(even(3)) < f(even(4))$$

[5] Čitateľ môže vystačiť s predstavou prirodzených čísel. Základné informácie o ordináloch možno nájsť v dodatku E.

atď. Dá sa to čítať aj tak, že $even(2)$ má vyššiu prioritu ako $even(1)$ atď.

Ľahko vidno, že platí $(\forall \phi \in M \setminus N)(\exists \psi \in N \setminus M)\ (f(\psi) < f(\phi))$. Ku každému základnému atómu z horšieho modelu N existuje atóm z lepšieho modelu, ktorý sa nevyskytuje v horšom modeli. Tento „lepší" atóm má vyššiu hodnotu v lokálnej stratifikácii podľa f (napríklad, $f(even(2)) > f(even(1))$). ■

Môžeme prejsť k definícii. V prvom kroku predpokladajme nejaké usporiadanie na základných atómoch z B_P. Potom prijmeme, že preferovanejšia bude tá interpretácia, ktorá minimalizuje počet atómov s nižšou prioritou.

Definícia 6.26 Relácia ρ na množine M je *dobre založená* práve vtedy, keď pre každú jej neprázdnu podmnožinu N platí $\exists x \in N \forall y \in N\ x\rho y$. ■

Teda, relácia ρ je dobre založená, ak každá neprázdna podmnožina jej nosiča má „najmenší" prvok (úvodzovky preto, lebo sme nepožadovali, aby ρ bola usporiadaním).

Definícia 6.27 Majme program P a dobre založené usporiadanie $<$ na B_P. Pre $I, J \subseteq B_P$ sa dohodneme, že $I \prec J$, ak pre každý atóm $\phi \in I \setminus J$ existuje atóm $\psi \in J \setminus I$ taký, že $\psi < \phi$.

Herbrandovský model M programu P je *perfektný*, ak neexistuje model N programu P, pre ktorý $N \prec M$. ■

Intuícia: Preferovanejšie sú tie interpretácie, ktoré minimalizujú atómy nižšej priority: Ak sa pozrieme na I a J z „dynamického hľadiska" (napríklad tak, že si predstavíme, že I vznikla z J nejakými modifikáciami), môžeme povedať, že I je preferovanejšia ako J ($I \prec J$), keď spomínaná modifikácia zmenšuje počet atómov nižšej priority: priorita každého atómu ϕ, ktorý sme „vložili do J" a tak prispeli ku vzniku I, je (vzhľadom na $<$) vyššia ako priorita nejakého atómu ψ, ktorý sme z J odstránili.

Tvrdenie 6.28 *Nech I a J sú modely (z nejakej triedy herbrandovských modelov nejakého programu P).*

(a) *Ak $I \subset J$, potom $I \prec J$.*

(b) *Každý perfektný model je minimálny.*

Idea dôkazu:
(a) Nech $I \not\prec J$, teda existuje atóm $\phi \in I \setminus J$ taký, že pre každý atóm $\psi \in J \setminus I$ je $\psi \geq \phi$. Lenže to je v spore s $I \setminus J = \emptyset$.
(b) Nech J nie je minimálny model. Potom – podľa predchádzajúceho – existuje I také, že $I \subset J$, a preto aj $I \prec J$. ■

Tvrdenie 6.29 (Przymusinski [Prz 87])

- *Každý lokálne stratifikovaný program má jediný perfektný model (pričom lokálna stratifikácia definuje usporiadanie na B_P).*
- *Ak je program P stratifikovaný, jeho perfektný model sa zhoduje so štandardným modelom M_P.*

6.2 Stabilné a dobre fundované modely

Najprv si uvedieme dva príklady.

Príklad 6.30 Uvažujme program

$$\begin{array}{rcl} p(Y) & \leftarrow & otec(X,Y), \neg p(X), \\ otec(a,b) & \leftarrow & . \end{array}$$

Tento program môžeme chápať tak, že nejaká vlastnosť sa do nasledujúcich generácií posúva mechanizmom „generačného odporu“. Ak priamy predchodca nemá nejakú vlastnosť, jeho priamy potomok ju má.

Inštanciami prvej klauzy sú

$$\begin{array}{rcl} p(a) & \leftarrow & otec(b,a), \neg p(b), \\ p(b) & \leftarrow & otec(a,b), \neg p(a). \end{array}$$

■

Príklad 6.31 Predstavme si podľa [Lif 88], že máme nasledujúci graf z obrázku 6.1. Vrcholmi sú 1, 2, 3, 4, 5, 6 a (orientované) hrany sú definované týmito vetami:

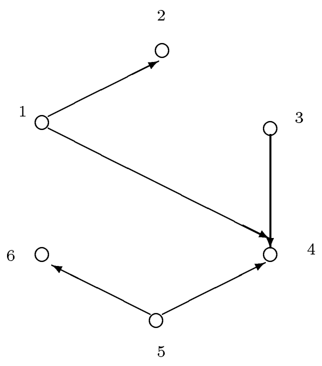
$$\begin{array}{rcl} e(1,2) & \leftarrow & , \\ e(1,4) & \leftarrow & , \\ e(3,4) & \leftarrow & , \\ e(5,4) & \leftarrow & , \\ e(5,6) & \leftarrow & . \end{array}$$

Predstavme si teraz smiešnu hru. Na niektorý vrchol sa umiestni značka. Hrajú dvaja hráči. Krok hry spočíva v posunutí značky pozdĺž jednej hrany. Vyhráva ten hráč, ktorý sa dostal do terminálneho vrcholu (značku už nemožno ďalej posunúť).

Pre daný graf toto pravidlo možno vyjadriť implikáciou

$$win(X) \leftarrow e(X,Y), \neg win(Y).$$

(Táto implikácia spolu s definíciami hrán uvedenými vyššie tvorí všeobecný logický program.) ■



Obrázok 6.1: Graf z príkladu 6.31

Obidva programy očividne nie sú stratifikované, ani lokálne stratifikované. Nemajú teda ani iterovaný model, ani perfektný model. Napriek tomu, uvedeným programom nemožno uprieť, že majú jasný zmysel.

Tento zmysel objasníme pomocou pojmu stabilného modelu. Základná myšlienka pripomína extenziu defaultových teórií alebo expanziu autoepistemických teórií.[6]

Predstavme si množinu S základných atómov v jazyku všeobecného programu P. Pýtajme sa, za akých okolností môže S reprezentovať zmysel P. Zrejme to bude vtedy, keď predpoklad pravdivosti všetkých atómov z S nevyvrátime, ak okrem S považujeme za pravdivý aj P a navyše, P už nedodá k S žiadne ďalšie pravdivé atómy (S je stabilná množina). Pod „dodávaním" myslíme okolnosť, že v modeli sa vyskytujú iba také atómy, ktoré získame výpočtom podľa P: každý atóm z S sa musí vyskytovať v hlave nejakej klauzy, ktorej telo je splnené v S (pripomeňme, že túto vlastnosť nazývame podopretosťou).

Zvyšok je už iba technická záležitosť: P transformujeme na definitný program, pričom využívame predpoklad pravdivosti S. S definitným programom si už poradíme: jeho zmysel reprezentuje (jediný) minimálny model. Ak tento mininimálny model je S, môžeme ho považovať za stabilný model pôvodného programu.

Napokon myšlienka transformácie: Považujme S za pravdivé. Predpokladajme klauzu $c = A \leftarrow A_1, \ldots, A_k, \neg B_1, \ldots, \neg B_n$, kde pre aspoň jedno i platí $B_i \in S$. Potom $\neg B_i$ musíme považovať za nepravdivé, preto aj celé telo klauzy c je nepravdivé a c je (triviálne) pravdivá. Jej ďalšia analýza neprinesie nič nového o zmysle programu P (hlava c nie je podopretá), preto celú klauzu môžeme zabudnúť. Ak pre všetky i platí $B_i \notin S$, potom z tela c môžeme vypustiť všetky negatívne literály, lebo sú na základe S pravdivé. Uvedené dva kroky urobia zo všeobecného programu P definitný program

[6]⟨ autoepistemické teórie : dodatok I ⟩

(budeme ho označovať $G(P, S)$ – vznikol po transformácii G aplikovanej na P a využívajúcej S).

Definícia 6.32 (Gelfond-Lifschitzova transformácia) Nech P je základný všeobecný logický program (nie nutne konečný). Nech S je množina základných atómov (nie nutne konečná).

$G(P, S)$ je definitný program, ktorý vznikne z P tak, že:

- eliminujeme z P všetky klauzy $A \leftarrow A_1, \ldots, A_k, \neg B_1, \ldots, \neg B_n$ také, kde pre aspoň jedno i platí $B_i \in S$,
- zo všetkých ostatných kláuz eliminujeme negatívne literály.

Takto transformovaný program označme $G(P, S)$. ■

Po transformácii G pre dané P a S dostávame definitný program, ktorý zaručene má minimálny model $M_{G(P,S)}$.

Definícia 6.33 Ak $M_{G(P,S)} = S$, hovoríme, že S je stabilný model programu P. ■

Poznámka 6.34 Okrem termínu stabilný model sa používajú aj termíny stabilná množina alebo odpoveďová množina (answer set). ■

Tvrdenie 6.35 *[GLi 88] Každý stabilný model programu P je jeho minimálnym modelom.*

Dôkaz: Nech S je stabilný model programu P. Ľahko sa ukáže, že S je modelom P. Predpokladajme ďalej, že S nie je minimálny model programu P, t.j. existuje S_1 také, že S_1 je modelom P a $S_1 \subset S$.

Predpokladajme, že pre nejakú klauzu $(A \leftarrow A_1, \ldots, A_k) \in G(P, S)$ je $\{A_1, \ldots, A_k\} \subseteq S_1$, ale $A \notin S_1$. Teda, predpokladáme, že S_1 nie je modelom $G(P, S)$. V opačnom prípade sa dostaneme do sporu s predpokladom $S_1 \subset S$, pretože S je na základe definície minimálnym modelom $G(P, S)$.

Nech $A \leftarrow A_1, \ldots, A_k \in G(P, S)$ sme získali Gelfond-Lifschitzovou transformáciou z $A \leftarrow A_1, \ldots, A_k, \neg B_1, \ldots, \neg B_n \in P$. To znamená, že

$$\{B_1, \ldots, B_n\} \cap S = \emptyset.$$

Z toho, samozrejme,

$$\{B_1, \ldots, B_n\} \cap S_1 = \emptyset.$$

Teda, S_1 spĺňa $\{A_1, \ldots, A_k, \neg B_1, \ldots, \neg B_n\}$, ale nespĺňa A. Preto nie je modelom P. To je však spor, lebo sme predpokladali, že S_1 je modelom P. ■

Tvrdenie 6.36 *Ak P je stratifikovaný, potom má jediný stabilný model, ktorý sa zhoduje s jeho štandardným modelom. Ak P je lokálne stratifikovaný, potom má jediný stabilný model, ktorý sa zhoduje s jeho perfektným modelom.*

Ak by sme od stabilných modelov chceli, aby jednoznačne charakterizovali význam logických programov, mali by sme prehnané ambície. Existujú programy, ktoré nemajú žiadny stabilný model (napr. $p \leftarrow \neg p$) alebo viac stabilných modelov (program $p \leftarrow \neg q, q \leftarrow \neg p$ má stabilné modely $\{p\}$ a $\{q\}$). Obzvlášť nepríjemné je, že stabilný model nemajú aj také programy, ktorých význam možno rozumným spôsobom charakterizovať.

Príklad 6.37 Nech P je stratifikovaný program, v ktorom sa nevyskytuje symbol p. Napriek tomu $P' = P \cup \{p \leftarrow \neg p\}$ nemá stabilný model S: Ak $p \in S$, potom S nie je minimálnym modelom $G(P', S) = G(P, S)$. Ak $p \notin S$, potom $G(P', S) = G(P, S) \cup \{p \leftarrow\}$, teda p by malo byť v minimálnom modeli $G(P', S)$. ■

Nie je celkom intuitívne, keď sa zrozumiteľný význam nejakej množiny kláuz stratí po dodaní novej klauzy, ktorá so zvyškom programu vôbec nesúvisí.

Jemného rozlišovania, ktoré je potrebné vo vyššie uvedenom príklade, je schopná dobre fundovaná sémantika [vGR 91]. Tu sa pridržiavame Przymusinského výkladu z [Prz 89b, Prz 91] (ktorý modifikoval pôvodnú formuláciu Van Geldera a jeho kolegov a ukázal ekvivalenciu oboch formalizácií).

Budeme teraz používať 3-interpretácie: dvojice (I^+, I^-), kde $I^+, I^- \subseteq B_P$. Intuitívne, tieto dve množiny sa považujú za atómy pravdivé (nepravdivé) pri danej interpretácii. Predpokladáme prázdny $I^+ \cap I^-$. Rozdiel $B_P \setminus (I^+ \cup I^-)$ predstavuje atómy s treťou pravdivostnou hodnotou (neznámy).

Hlavný vtip prístupu spočíva v tom, že na báze daného programu P sa skonštruuje množina atómov, ktoré sú nutne pravdivé a množina tých, ktoré sú nutne nepravdivé, ak je program pravdivý. Zvyšok atómov ostane nerozhodnutých.

Zadefinujeme funkciu val, ktorá priraďuje pravdivostné hodnoty formulám jazyka. Budeme predpokladať usporiadanie $t > u > f$ na pravdivostných hodnotách a negáciu, operujúcu na pravdivostných hodnotách takto: $\neg t = f$, $\neg f = t$, $\neg u = u$.

Definícia 6.38 Nech A je atóm, I je trojhodnotová interpretácia.

Potom $val_I(A) = t$ práve vtedy, keď $A \in I^+$ a $val_I(A) = f$, ak $A \in I^-$. Inak je $val_I(A) = u$.

Pre ľubovoľné formuly F, G je:

$$\begin{aligned}
val_I(\neg F) &= \neg val_I(F), \\
val_I(F \wedge G) &= min\{val_I(F), val_I(G)\}, \\
val_I(F \vee G) &= max\{val_I(F), val_I(G)\}.
\end{aligned}$$

Implikáciu zadefinujeme dvojhodnotovú: $val_I(F \leftarrow G) = t$, ak $val_I(F) \geq val_I(G)$, inak je $val_I(F \rightarrow G) = f$. ■

Ďalej sa budeme zaoberať iba základnými inštanciami logických programov, preto pravdivostné podmienky pre formuly s premennými a kvantifikátormi nezavedieme.

Definícia 6.39 Majme základný všeobecný logický program P. Budeme hovoriť, že trojhodnotová interpretácia I je jeho *modelom*, ak pre každú klauzu c z P platí, že $val(c) = t$. ■

Teraz sa oboznámime s pomerne komplikovanou konštrukciou, ktorá vedie k pojmu dobre fundovaného modelu. Táto konštrukcia zahŕňa dva iteratívne procesy. Vo vonkajšej iterácii budeme štartovať s prázdnou množinou pravdivých a prázdnou množinou nepravdivých základných atómov (s interpretáciou I_0). Odvodíme z nej interpretáciu I_1, množinu základných atómov nutne pravdivých a množinu základných atómov nutne nepravdivých za predpokladu pravdivosti programu P (a vzhľadom na interpretáciu I_0). Spomínané dve množiny základných atómov skonštruujeme pomocou ďalšej iterácie (nazvime ju vnútornou) využívajúcej operátory $\mathcal{T}_I$ a $\mathcal{F}_I$, zavedené v definícii 6.40. Vnútornú iteráciu aplikujeme ďalej: Aplikáciou na I_1 získame I_2. Postupujeme, kým sa dá niečo nového odvodiť, teda kým nedostaneme definitívnu predstavu o atómoch nutne pravdivých a nutne nepravdivých za predpokladu pravdivosti programu P. Potom s touto novou dvojicou množín atómov pokračujeme vo vonkajšej iterácii dovtedy, kým nedosiahneme pevný bod.

Zadefinujeme teraz operátory $\mathcal{T}_\mathcal{I}$ a $\mathcal{F}_\mathcal{I}$. Ich parametrom je trojhodnotová interpretácia I a okrem nej ešte dve množiny základných atómov, T, F (množiny zatiaľ pravdivých a zatiaľ nepravdivých viet).

Definícia 6.40 Nech I je interpretácia, T a F nech sú disjunktné množiny základných atómov, P je program.

$$\begin{aligned}\mathcal{T}_I(T) &= \{A : \exists(A \leftarrow L_1, \ldots, L_k) \in P\ \forall i(val_I(L_i) = t \vee L_i \in T)\},\\ \mathcal{F}_I(F) &= \{A : \forall(A \leftarrow L_1, \ldots, L_k) \in P\ \exists i(val_I(L_i) = f \vee L_i \in F)\}.\end{aligned}$$

■

Prvý z operátorov „zbiera prírastky“ vyprodukované pravidlami programu P, ak literály z ich tela sú podopreté buď argumentom z I alebo z T. Druhý sa opiera o nepravdivosť literálov vzhľadom na I alebo o prvky z F. Očividne smerujeme k rozlíšeniu pravdivých a nepravdivých literálov vzhľadom na dva druhy informácií: I a (T, F).

Definícia 6.41 Nech $I_1 = (I_1^+, I_1^-)$ a $I_2 = (I_2^+, I_2^-)$. Budeme hovoriť, že

- I_1 je *informačne lepšie* ako I_2 ($I_1 \geq_i I_2$), ak $I_2^+ \subseteq I_1^+$ a $I_2^- \subseteq I_1^-$,

- I_1 je *pravdivostne bohatšie* ako I_2 ($I_1 \geq_p I_2$), ak $I_2^+ \subseteq I_1^+$) a $I_1^- \subseteq I_2^-$.

■

Teda, interpretácia I je informačne lepšia ako interpretácia J, ak množina atómov, ktoré majú na základe I neznámu pravdivostnú hodnotu, sa zmenšuje (v porovnaní s J). Na druhej strane, kritériom toho, že I je pravdivostne lepšia ako J, je zväčšovanie podielu pravdivých atómov a zmenšovanie podielu nepravdivých (podľa I a v porovnaní s J).

Tvrdenie 6.42 *$\mathcal{T}_\mathcal{I}$ a $\mathcal{F}_\mathcal{I}$ sú monotónne vzhľadom na $\geq_i$.*

Teraz nad operátormi $\mathcal{T}_\mathcal{I}$ a $\mathcal{F}_\mathcal{I}$ zadefinujeme dva druhy mocnín.

Definícia 6.43 Nech je daný program P a interpretácia I. Potom

$$\begin{array}{lcllcl} T_I \uparrow 0 & = & \emptyset & F_I \downarrow 0 & = & B_P \\ T_I \uparrow (n+1) & = & \mathcal{T}_\mathcal{I}(T_I \uparrow n) & F_I \downarrow (n+1) & = & \mathcal{F}_\mathcal{I}(F_I \downarrow n) \\ T_I & = & \bigcup\limits_{n<\omega} (T_I \uparrow n) & F_I & = & \bigcap\limits_{n<\omega} (F_I \downarrow n). \end{array}$$

■

Práve T_I je tá množina, ktorá obsahuje nové fakty, odvoditeľné z P, ak poznáme I; podobne F_I obsahuje nové fakty, ktoré môžu byť považované za nepravdivé na základe P a I.

Tvrdenie 6.44 *Nekonečná postupnosť $\{T_I \uparrow n\}$ monotónne rastie, $\{F_I \downarrow n\}$ monotónne klesá.*

$\mathcal{T}_I(T_I) = T_I$ a $\mathcal{F}_I(F_I) = F_I$, pritom F_I (a T_I) je najmenší – vzhľadom na $\supseteq$ (resp. $\subseteq$) – pevný bod operátora $\mathcal{F}_I$ (resp. $\mathcal{T}_I$).

Teraz prejdeme k definovaniu už spomínanej „vonkajšej iterácie“.

Definícia 6.45 Nech $I = (I^+, I^-)$ a $\Upsilon(I) = (I^+ \cup T_I, I^- \cup F_I)$. Potom

$$\begin{array}{rcl} M_0 & = & (\emptyset, \emptyset), \\ M_{\alpha+1} & = & \Upsilon(M_\alpha), \\ M_\alpha & = & \bigcup\limits_{\beta<\alpha} M_\beta. \end{array}$$

Posledná rovnosť platí pre limitné ordinály. ■

Príklad 6.46 Majme program $P = \{$

$$\begin{array}{rcl} p & \leftarrow & \neg q, \\ q & \leftarrow & \neg r, \\ r & \leftarrow & \neg s, \\ s & \leftarrow & \}. \end{array}$$

Nech $M_0 = (\emptyset, \emptyset)$. Potom $\Upsilon(M_0) = (\emptyset \cup T_{M_0}, I^- \cup F_{M_0})$. T_{M_0} a F_{M_0} počítame „vnútornou" iteráciou: $\mathcal{T}_{M_0}(\emptyset) = \{s\}$, ďalej $\mathcal{T}_{M_0}(\{s\}) = \{s\}$. Teda $T_{M_0} = \{s\}$. Podobne $\mathcal{F}_{M_0}(B_P) = \emptyset$ a $\emptyset$ je pevným bodom operátora $\mathcal{F}_{M_0}$.

Preto $M_1 = (\{s\}, \emptyset)$. Pevným bodom operátora $\mathcal{T}_{M_1}$ je opäť $\{s\}$. Pevným bodom operátora $\mathcal{F}_{M_1}$ je $\{r\}$ (pretože $val_{M_1}(\neg s) = f$ a $r \leftarrow \neg s$ je z P). To znamená, že $M_2 = (\{s\}, \{r\})$.

Ďalej dostávame: Pevným bodom $\mathcal{T}_{M_2}$ je $\{s, q\}$ a $M_3 = (\{s, q\}, \{r\})$. Pevným bodom $\mathcal{F}_{M_3}$ je $\{r, p\}$.

Napokon $M_4 = (\{s, q\}, \{r, p\})$ je pevným bodom, získaným obidvoma vnorenými iteráciami. ■

Tvrdenie 6.47 *Υ je monotónny operátor. Preto existuje najmenšie δ také, že $M_\delta = \Upsilon(M_\delta) = M_{\delta+1}$. Označíme si ho ako WFM_P a nazveme* dobre fundovaný model *programu P.*

Tvrdenie 6.48 *WFM_P je minimálny trojhodnotový model P a najmenší pevný bod operátora Υ.*

Logický program je *saturovaný*, ak jeho dobre fundovaný model je dvojhodnotový.

Tvrdenie 6.49 *Pre saturované programy sa zhodujú dobre fundovaná sémantika a sémantika stabilných modelov.*

Przymusinski ukázal, že konštrukcia dobre fundovaných modelov je súčasne konštrukciou akejsi stratifikácie. Podľa definície je $M_{\alpha+1} = \Upsilon(M_\alpha)$, pričom $\Upsilon(M_\alpha) = (M_\alpha^+ \cup T_{M_\alpha}, M_\alpha^- \cup F_{M_\alpha})$. Stratifikácia sa teraz dá definovať pomocou tých základných atómov, ktoré sa ako nové dodávajú (medzi pravdivé alebo nepravdivé) v procese iterácie.

Definícia 6.50 *Dynamická stratifikácia* programu P pozostáva z vrstiev:

$$\begin{array}{rcl} S_0 & = & T_{M_0} \cup F_{M_0}, \\ S_\beta & = & (T_{M_\beta} \cup F_{M_\beta}) \setminus \bigcup_{\alpha < \beta} S_\alpha. \end{array}$$

Ak $M_\delta = M_P$ je dobre fundovaný model programu P, potom

$$S_\delta = B_P \setminus \bigcup_{\alpha < \delta} S_\alpha.$$

■

Príklad 6.51 Dobre fundovaný model programu $p \leftarrow \neg p$ je $(\emptyset, \emptyset)$. Vidíme teda, že vďaka trojhodnotovosti nás nezaskočí príklad 6.37, ktorým sme demonštrovali nedostatočnú intuitívnosť sémantiky stabilných modelov. ■

Tvrdenie 6.52 ([vGR 91]) *Ak dobre fundovaný model programu P je totálny, potom je zhodný s jediným stabilným modelom P.*

Žiaľ, treba podotknúť, že aj dobre fundovaná sémantika má vady, ktoré vedú k neintuitívnym pravdivostným ohodnoteniam.

Príklad 6.53 Majme program P: $\{p \leftarrow \neg q, q \leftarrow \neg p, r \leftarrow p, r \leftarrow q\}$.

Jeho stabilnými modelmi sú $\{p, r\}$ a $\{q, r\}$. Vzhľadom na tieto dva modely sú $p \vee q$ aj r pravdivé. Avšak dobre fundovaný model P je $(\emptyset, \emptyset)$, teda pravdivostná hodnota r je vzhľadom na tento model u. Je to v spore s prirodzenými intuíciami. ∎

Príklad 6.54 Nech P je $\{p \leftarrow \neg q; q \leftarrow \neg p\}$ a $P' = \{p \leftarrow \neg p; q \leftarrow \neg q\}$.

Dobre fundovaným modelom oboch programov je model, ktorý považuje aj p, aj q za nedefinované.

Naproti tomu, stabilnými modelmi P sú $M_1 = \{p, \neg q\}$ a $M_2 = \{\neg p, q\}$. P' nemá stabilný model.

Dobre fundovaný model nie je schopný rozlíšiť výrazný významový rozdiel medzi týmito dvoma programami. Zato však, zdá sa, sémantika stabilných modelov je v tomto prípade schopná adekvátneho rozlíšenia. ∎

Záverom toľko, že neexistuje jediná, všeobecne prijatá sémantika všeobecných logických programov. Oblasť je stále predmetom intenzívneho výskumu a jej význam pre umelú inteligenciu a pre štúdium hypotetického usudzovania je veľmi dôležitý.

6.3 Zložitosť a implementácie

V tejto časti sa stručne budeme venovať výpočtovým aspektom riešenia problémov, súvisiacich so stabilnými modelmi a úplne okrajovo s dobre fundovanými modelmi. Dôvodom je veľmi zaujímavý a podľa všetkého dôležitý výskumný trend, zameraný na výpočty stabilných modelov.

Uvedieme základnú informáciu o výpočtovej zložitosti a o implementačných experimentoch.

Najprv schéma problému, ktorého špecializácie budeme predpokladať. Tá schéma je: patrí základný atóm A do modelu M? Jej špecializácie vzniknú tak, že budeme predpokladať niečo navyše o modeli M pre rôzne druhy logických programov a ich sémantík. Podrobnejší prehľad o zložitostných charakteristikách možno nájsť v [CaS 93], kde sú aj referencie na pôvodné zdroje uvádzaných výsledkov.

Pre definitné logické programy budeme predpokladať, že M je minimálny model. Vo výrokovologickom prípade existuje lineárny algoritmus (kapitola 2, algoritmus 2.47). V prípade logického programu v jazyku predikátovej logiky je tento problém RE-úplný.[7]

[7]⟨ zložitosť : dodatok H ⟩

Pre stratifikované programy budeme predpokladať, že M je perfektný model. Vo výrokovologickom prípade existuje polynomický algoritmus. Predikátovologický prípad je samozrejme ťažší: ide o problém z triedy Σ_n^0, kde n je počet vrstiev programu. Stratifikované programy teda dokážu špecifikovať množiny v aritmetickej hierarchii, čiže ide o vyjadrovaciu silu za hranicou vypočítateľnosti. V prípade lokálne stratifikovaných programov sa dostávame ešte ďalej za hranicu vypočítateľnosti – do triedy hyperaritmetických množín, pozri napríklad [Rog 67].

Na druhej strane, dobre založený model výrokovologického programu možno vypočítať v polynomickom čase.

Pre všeobecné logické programy a ich stabilné modely rozšírime množinu problémov, ktoré nás zaujímajú. Najprv výrokovologický prípad. Rozhodnutie, či existuje stabilný model programu P je NP-úplný problém. Rovnako aj rozhodnutie, či daný základný atóm A patrí do nejakého stabilného modelu. Rozhodnutie, či A patrí do každého stabilného modelu je coNP-úplný problém.

Pre predikátovologické programy sa rozdiel medzi stabilnými a dobre fundovanými modelmi stiera. Samozrejme, ide o výpočtovo veľmi ťažké problémy (v aritmetickej a hyperaritmetickej hierarchii).

Teraz budeme venovať pozornosť implementáciám. Najprv jediná poznámka o výpočte dobre fundovaných modelov. Známy je systém odpovedania na otázky XSB, ktorý využíva dobre fundovanú sémantiku. Jeho schopnosť ukončiť výpočet je lepšia ako v Prologu vďaka tomu, že pozná tretiu pravdivostnú hodnotu unknown (neznámy, nedefinovaný). Je verejne dostupný na `http://www.cs.sunysb.edu/~sbsprolog/xsb-page.html`.

Významným súčasným trendom sú implementačné experimenty so stabilnými modelmi. Ide predovšetkým o:
smodels – `http://www.tcs.hut.fi/Software/smodels/`
dlv – `http://www.dbai.tuwien.ac.at/proj/dlv`
DeReS – `http://www.cs.engr.uky.edu/~lpnmr/DeReS.html`.

Základná metóda výpočtu je hľadanie stabilných modelov v priestore všetkých možných modelov s využitím backtrackingu.[8] Samozrejme, v primitívnej forme ide o veľmi neefektívnu metódu. Žiaľ, principiálna nezvládnuteľnosť problémov, súvisiacich s výpočtom stabilných modelov je nezvratná, videli sme už, že pre výrokovologický prípad ide o NP-úplné alebo coNP-úplné problémy.

Napriek tomu sa v posledných rokoch dosiahol významný pokrok v metódach výpočtu stabilných modelov. Najlepšie systémy sú schopné v rozumnom čase vypočítať stabilné modely s desiatkami tisícok nestratifikovaných pravidiel.

Vzhľadom na tieto výsledky sa začína hovoriť o novej paradigme programovania [Lif 99, Nie 98, MaT 99]. Nazvime ju paradigmou stabilného logického programovania (*SLP*) ([Lif 99] a podľa neho už aj [MaT 99] pre-

[8] $\langle$ backtracking : G $\rangle$

sadzujú termín answer set programming).

Predpokladajme teda konečný základný logický program bez funkčných symbolov. Alternatívou je program s premennými, ale s využívaním procedúry, ktorá takýto program transformuje do tvaru základného programu. Pri tradičnom logickom programovaní sa programu položí otázka a program odpovie. Objekt, ktorý je riešením (odpoveďou), program skonštruuje ako term a substituuje ho za premennú z otázky. Pri stabilnom logickom programovaní je riešením celý stabilný model.

Príklad 6.55 ([FPL 99]) Nasledujúci disjunktívny logický program (s disjunkciou v hlave pravidla) rieši problém Hamiltonovskej cesty (cesta v orientovanom grafe, na ktorej leží každý vrchol grafu presne raz).
$P = \{$

$$\begin{array}{rcl}
reached(X) & \leftarrow & start(X), \\
reached(X) & \leftarrow & reached(X), inPth(Y,X), \\
inPth(X,Y) \vee outPth(X,Y) & \leftarrow & arc(X,Y), \\
& \leftarrow & inPth(X,Y), inPth(X,Y1), Y \neq Y1, \\
& \leftarrow & inPth(X,Y), inPth(X1,Y), X \neq X1, \\
& \leftarrow & node(X), \neg reached(X)\}.
\end{array}$$

Jediný vrchol, na ktorom štartujeme Hamiltonovskú cestu, spĺňa predikát *start*. Všetky vrcholy, ktoré ležia na nej, spĺňajú predikáty *reached*. Každá hrana je buď na tejto ceste alebo mimo nej (buď spĺňa *inPth* alebo *outPth*). Dve integritné obmedzenia zakazujú, aby rôzne hrany, na ktorých leží nejaký vrchol, boli na konštruovanej Hamiltonovskej ceste. Posledné integritné obmedzenie zakazuje, aby nejaký vrchol ostal nedosiahnutý.

Ak dodáme fakty pre nejaký graf (množinu faktov s predikátmi *node*, *arc*), uvedený program nájde Hamiltonovské cesty v tomto grafe – budú definované jeho stabilnými modelmi. Pre nejaký stabilný model S všetky atómy tvaru $inPth(u,v)$ určujú, že vrcholy u a v, ležiace na jednej hrane, ležia na Hamiltonovskej ceste, ktorú definuje S. Aj ostatné atómy z modelu S sú pravdivé pre daný graf a danú Hamiltonovskú cestu.

Pri klasickom prologovskom programovaní by sme museli program navrhnúť tak, aby sme skonštruovali nejaký term, reprezentujúci Hamiltonovskú cestu. Napríklad:

$$\begin{array}{rcl}
path(X,X,[X]) & \leftarrow & , \\
path(X,Y,[X|P]) & \leftarrow & arc(X,N), path(N,Y,P).
\end{array}$$

Najprv definujeme predikát *path*: $path(X,Y,P)$ znamená, že z vrcholu X ide do vrcholu Y cesta P, ktorú reprezentujeme nejakým zoznamom. Podľa prvej klauzy ľubovoľný vrchol predstavuje (elementárnu) cestu. Druhá (rekurzívna) klauza hovorí, že keď máme cestu P z N do Y a pridáme hranu z X do N, dostávame cestu $[X|P]$ z X do Y. Ďalej:

$$\begin{array}{rcl} is\_path(P) & \leftarrow & path(X,Y,P), \\ hamilton(P) & \leftarrow & is\_path(P), node(X), unique\_member(X,P), \\ unique\_member(X,P) & \leftarrow & member(X,P), append(L,[X|T],P), \\ & & \neg member(X,L), \neg member(X,T). \end{array}$$

Zoznam P reprezentuje Hamiltonovskú cestu (spĺňa predikát *hamilton*), ak je cestou, leží na nej každý vrchol, a leží na nej iba raz.

Keď tomuto programu položíme otázku $\leftarrow hamilton(P)$, odpoveďou bude zoznam P, ktorého prvkami sú vrcholy a prechod cez vrcholy v danom poradí tvorí Hamiltonovskú cestu. ■

Teda, pri paradigme *SLP* je riešením problému stabilný model. Každý *SLP*-program špecifikuje konečnú triedu konečných množín. Každá z týchto množín obsahuje jedno riešenie daného problému.

Táto zmena pohľadu na logické programovanie je plne v duchu zásad deklaratívneho programovania: separuje logiku od riadenia. Logika je špecifikovaná stabilným modelom. Riadenie je založené na prehľadávaní všetkých podmnožín herbrandovskej bázy programu s využitím backtrackingu. Podstatné je však to, že boli navrhnuté účinné techniky osekávania prehľadávaného priestoru, ktoré umožňujú slušné výpočtové správanie. Dosiahnuté výsledky sú skutočne pôsobivé, výkonnosť systémov sa rýchlo zlepšuje. Napríklad prvá verzia systému `dlv` nedokázala nájsť Hamiltonovskú cestu (stabilný model) na grafe s 25 vrcholmi a 60 hranami (rádovo ide o desiatky tisícov základných inštancií pravidiel a integritných obmedzení vyššie uvedeného programu) ani za 1000 sekúnd, 2. verzia našla jedno riešenie za 716 sekúnd. S použitím heuristík 4. verzia našla cestu za 12.7 sekúnd, pozri [FPL 99].

Všetky problémy, ktoré sú v NP, možno riešiť paradigmou SLP. Ide o triedu významných problémov, medzi nimi sú mnohé problémy, kde ide o kombinatorickú optimalizáciu a spĺňanie obmedzení. V istom zmysle slova vlastne ide o nový prístup k programovaniu s obmedzeniami (constraint programming) [Nie 98]. Množina všetkých možných podmnožín herbrandovskej bázy tvorí základný priestor a pravidlá programu (spolu s integritnými obmedzeniami) určujú podmienky, kladené na riešenie problému. Na základe spĺňania týchto podmienok sa vyberajú riešenia zo základného priestoru. Každé získané riešenie je automaticky minimálne a podopreté. To znamená, neobsahuje redundanciu. Ak je potrebné niektoré stabilné modely odfiltrovať z nejakých dôvodov, možno tak učiniť pomocou dodatočných integritných obmedzení.

Možnými aplikáciami *SLP* programovania sú:

- programovanie s obmedzeniami,

- plánovanie (pozri [Lif 99]),[9]
- kombinatorické problémy, grafárske problémy (pozri napríklad DeReS),
- diagnostikovanie (`dlv`),
- verifikácia distribuovaných systémov (`smodels`).

Základné *SLP* programy sú vlastne výrokovologickými programami. Keďže *SLP* programovanie nachádza stabilné modely, môže vzniknúť otázka, či modely ľubovoľných výrokovologických teórií (množín výrokovologických formúl) nebudú rovnako užitočným reprezentantom riešení problémov (s výpočtovým mechanizmom preverovania splniteľnosti). Dá sa však ukázať, že z hľadiska reprezentácie poznatkov sú *SLP* programy silnejším nástrojom než ľubovoľné výrokovologické teórie. Po prvé, stabilné modely sú minimálne a podopreté.

Po druhé, Niemelä dokázal, že problém výrokovologickej splniteľnosti možno *modulárne* redukovať na problém nájdenia stabilného modelu, neexistuje však opačná modulárna redukcia. Samozrejme, vzhľadom na *NP*-úplnosť oboch problémov, existuje polynomická redukcia v opačnom smere. Nie je však modulárna.

Intuitívne, modulárne zobrazenie má tú vlastnosť, že dodanie nejakého atómu k programu vyžaduje iba lokálnu zmenu tohto zobrazenia, ktorá nezasiahne zvyšok. Očividne, modulárnosť je z hľadiska reprezentácie poznatkov dôležitá vlastnosť.

Definícia 6.56 Nech F je zobrazenie z výrokovologických programov do výrokovologických teórií. Hovoríme, že F je *modulárne*, ak pre každý program P a každú množinu atómov S platí, že $P \cup S$ má stabilný model práve vtedy, keď množina $S \cup F(P)$ je splniteľná. ■

Za dôležité ciele pre ďalší výskum a pre uplatnenie *SLP*-programovania možno považovať: Systematické štúdium metodológie vývoja *SLP* programov (príkladom tohto druhu výskumu je aj rozširovanie vyjadrovacích schopností pre písanie *SLP* programov, pozri [Sms 99]). Ďalej, prehlbovanie vhľadu do výpočtovej povahy stabilných modelov a vývoj rýchlejších algoritmov. Pôvodná metóda výpočtu stabilných modelov bude v kapitole 7.1.

6.4 Reprezentácia znalostí a sémantiky negácie

V tejto časti si ukážeme, ako možno niektoré sémantiky negácie využiť pri reprezentácii poznatkov. Použijeme príklady, na ktorých sa zvykne testovať vyjadrovacia schopnosť reprezentačných formalizmov a demonštrujeme na

[9] Informácie o využití rôznych systémov *SLP* na problémy plánovania sú na `http://www.cs.utexas.edu/users/esra/experiments/experiments.html`.

nich, že sémantická charakterizácia pomocou sémantík negácie dáva korektné riešenia. Záujemcu o podrobnejšiu prezentáciu týchto problémov odkazujeme na [BGe 94].

Nasledujúci program ukazuje, že hierarchické siete možno reprezentovať stratifikovanými programami. (Nás to, po zvládnutí časti 4.2 vôbec neprekvapuje.)

Príklad 6.57 Pripomeňme si príklad 4.139 z časti 4.2, trochu ho zmodifikujeme. Nech program P obsahuje nasledujúce pravidlá a fakty:

$$\begin{array}{lll}
r1 & lieta(X) & \leftarrow \; vták(X), \neg ab_1(X), \\
r2 & vták(X) & \leftarrow \; tučniak(X), \\
r3 & ab_1(X) & \leftarrow \; tučniak(X), \\
r4 & urob_strechu_pre(X) & \leftarrow \; lieta(X), \\
f1 & vták(a) & \leftarrow \; , \\
f2 & tučniak(b) & \leftarrow \; .
\end{array}$$

Jeho stratifikácia je (napríklad) $P_1 = \{f1, f2, r3, r2\}, P_2 = \{r1, r4\}$.

Potom je $M_{P_1} = \{vták(a), tučniak(b), vták(b), ab_1(b)\}$ a $M_{P_2} = M_{P_1} \cup \{lieta(a), urob\_strechu\_pre(a)\}$. Samozrejme, $M_P = M_{P_2}$.

Odpovede na otázky môžeme nájsť v M_P. Napríklad, odpoveďou na $\leftarrow lieta(X)$ je *áno*; $X = a$, odpoveďou na $\leftarrow lieta(b)$ je *nie* atď. ■

Druhý príklad je známy pod menom Yale Shooting Problem. Stal sa chýrnym preto, lebo s jeho pomocou demonštrovali Hanks a McDermott [HmD 87] nedostatočnosť niektorých formalizmov ako prostriedku na reprezentáciu akcií a ich výsledkov.

Príklad 6.58 V pôvodnej formulácii sa zavádzajú tieto konštanty typu A (akcie):

$$\{load, wait, shoot\}.$$

Konštatnty typu F (fluenty, t.j. výroky, ktorých pravdivosť sa mení v závislosti na vykonaných akciách) sú:

$$\{alive, loaded\}.$$

Používajú sa dva predikáty, *holds* a *ab*. Atóm tvaru $holds(F, S)$ znamená, že fluenta F platí v situácii S. Atóm tvaru $ab(R, A, F, S)$ vyjadruje, že je abnormálne vzhľadom na pravidlo R, ak po akcii A v situácii S platí fluenta F. Používa sa aj funkčný symbol res, pričom $res(A, S)$ reprezentuje situáciu, ktorá vznikne po akcii A v situácii S. Nasleduje logický program, ktorý reprezentuje Yale Shooting Problem:

$$\begin{array}{lll}
r1 & holds(F, res(A, S)) & \leftarrow \; holds(F, S), \neg ab(r1, A, F, S), \\
r2 & holds(loaded, res(load, S)) & \leftarrow \; , \\
r3 & ab(r1, shoot, alive, S) & \leftarrow \; holds(loaded, S), \\
r4 & holds(alive, s_0) & \leftarrow \; .
\end{array}$$

Program nie je stratifikovaný: *holds* negatívne závisí na *ab* (podľa r1) a *ab* závisí na *holds* (podľa r3). Závislostný graf základných atómov je však acyklický: Cyklus môžu spôsobiť iba ($r1$) a ($r3$). Podľa ($r3$) sa však môžeme dostať od ľubovoľnej inštancie $ab(r1, shoot, alive, S)$ k zodpovedajúcej inštancii $holds(loaded, S)$, od nej sa podľa ($r1$) dostaneme iba k inštanciám tvaru $ab(r1, A, loaded, S)$. Teda, program je lokálne stratifikovaný. Vieme, že lokálne stratifikovaný program má jediný stabilný model (zhodný s jeho perfektným modelom). Tento stabilný model je nekonečný (vzhľadom na to, že funkčný symbol *res* umožňuje tvoriť stále nové inštancie situácií), obsahuje však tieto atómy:

$$\begin{gathered}\{holds(alive, s_0),\\ holds(alive, res(shoot, s_0)),\\ holds(loaded, res(load, s_0)),\\ holds(alive, res(load, s_0)),\\ ab(r1, shoot, alive, res(load, s_0))\}.\end{gathered}$$

Odpovedanie na otázky môžeme špecifikovať na základe platnosti v stabilnom modeli. Napríklad odpoveď na otázku

$$\leftarrow \neg holds(alive, res(shoot, res(load, s_0)))$$

je *áno*, podobne aj na otázku

$$\leftarrow \neg holds(alive, res(shoot, res(wait, res(load, s_0)))).$$

Hanks a McDermott zistili, že práve táto otázka robí problémy niektorým formalizmom, napríklad aj normálnym defaultom. Vsunutie akcie *wait* dokáže spôsobiť neintuitívny dôsledok o možnej platnosti fluenty *alive* v stave

$$res(shoot, res(wait, res(load, s_0))).$$

V prípade defaultovej formalizácie v niektorej extenzii nemusí platiť

$$ab(r1, shoot, alive, res(wait, res(load, s_0))).$$

Sémantika stabilných modelov dokáže tento problém riešiť korektne preto, lebo konštruuje iba podoprеté modely. Akcia *wait* nedovoľuje odvodiť nijaký nový dôsledok o platnosti nejakej ďalšej fluenty na základe uvedeného programu a stabilnej sémantiky. ■

Oba uvedené príklady ukázali, že sémantiky negácie môžu slúžiť ako nástroje na reprezentáciu niektorých klasických príkladov, analyzovaných v teórii nemonotónneho odvodzovania. Nasledujúca sekcia však prinesie fundamentálnejší pohľad: Dá sa ukázať, že pomocou týchto sémantík možno presne charakterizovať najznámejšie formalizácie nemonotónneho odvodzovania.

6.5 Sémantika nemonotónnej inferencie

Jedným z dôležitých prínosov umelej inteligencie je štúdium problematiky nemonotónneho odvodzovania. Výskum ľudskej schopnosti usudzovať sa vďaka tomu posunul smerom, ktorý sa zdá byť plodným. Navyše, vznikla tak nová paradigma pre výpočtové modelovanie inteligencie.

Historicky prvé a dodnes najvýznamnejšie formalizácie nemonotónneho usudzovania sú defaultové teórie, predpoklad uzavretého sveta, cirkumskripcia a autoepistemická logika. S prvými dvoma sme sa už zoznámili v kapitolách 3 a 4. Základným informáciám o cirkumskripcii a autoepistemickej logike sú venované dodatky J a I.

Výskum nemonotónneho odvodzovania na jednej strane a výskum negácie v logickom programovaní na strane druhej prebiehal spočiatku izolovane, nezávisle na sebe. Koncom osemdesiatych rokov sa však objavili zreteľné súvislosti medzi týmito oblasťami.

Przymusinski v [Prz 89b] vyzdvihol, že pre stratifikované logické programy je sémantika perfektných modelov ekvivalentná so všetkými štyrmi uvedenými formalizmami (resp. ich stratifikovanými verziami). Pre defaultové teórie to dokázali [BiF 87], pre CWA [GPP 89], pre cirkumspkripciu [Prz 89a, Lif 88], pre autoepistemickú logiku [Gel 87].

Teda, za prítomnosti stratifikácie (t.j. vhodnej hierarchie definícií v báze znalostí) základné formalizácie nemonotónneho odvodzovania splývajú. Toto zistenie o korešpondencii pôvodne nezávislých smerov výskumu je určite zaujímavé. Stratifikáciu pritom možno považovať za veľmi prirodzenú formu organizácie (nie iba báz znalostí). Navyše, spomínané zistenie poukazuje na to, že efektívne výpočtové metódy, navrhnuté pre logické programovanie a deduktívne dátabázy možno použiť ako inferenčné nástroje pre tradične skúmané formy nemonotónneho odvodzovania. Stratifikované logické programy môžu slúžiť ako vhodný výpočtový nástroj pre všetky uvedené formalizmy.

Teraz si iba stručne zrekapitulujeme spomínané výsledky o ekvivalencii:

Tvrdenie 6.59 ([Gel 87]) *Ak stratifikovaný logický program P transformujeme v stratifikovanú autoepistemickú teóriu tak, že nahradíme každé $\neg p$ pomocou $\neg Lp$, platí:*

Pre každý stratifikovaný logický program P existuje jediná autoepistemická expanzia autoepistemickej teórie, získanej uvedenou transformáciou.

Tvrdenie 6.60 ([Lif 88]) *Majme stratifikovaný logický program P. Nech $Pred$ je množina predikátov z P a*

$$\preceq \, = \, \{(p,q) \in Pred \times Pred : v\ z\acute{a}vislostnom\ grafe\ G_P\ vedie\ hrana\ od\ p\ do\ q\ \}.$$

Každý stratifikovaný logický program má presne jeden herbrandovský model, ktorý spĺňa cirkumskripčnú formulu $CIRC(DB; \{q : p \preceq q\}; p)$.[10]

[10]Ide o cirkumskripciu po bodoch a s priroritami, pozri dodatok J.

Tvrdenie 6.61 ([BiF 87]) *Nech P je stratifikovaný logický program. M je jeho defaultový model práve vtedy, keď je jeho perfektný model.*

Tvrdenie 6.62 ([GPP 89]) *Nech P je logický program stratifikovaný do vrstiev $(P_1, \ldots, P_n)$. Potom jeho model M je perfektným modelom P práve vtedy, keď je modelom množiny formúl, získaných pomocou ICWA, t.j. CWA, iterovaného po vrstvách.*

Dôsledok 6.63 *Defaultová logika, cirkumskripcia po bodoch a s prioritami, stratifikovaná autoepistemická logika, iterovaný predpoklad uzavretého sveta, perfektné modely majú ten istý význam pre stratifikované logické programy.*

Napokon, už iba stručná charakterizácia vzťahu dobre fundovanej sémantiky a klasických nemonotónnych formalizmov (výsledok dosiahol Przymusinski [Prz 91]):

Tvrdenie 6.64 *Pre ľubovoľný program je dobre fundovaná sémantika ekvivalentná trojhodnotovým rozšíreniam cirkumskripcie, defaultovej teórie, autoepistemickej logiky a CWA.*

6.6 Komentáre

Rozšírenia logického programovania V súvislosti s výskumom v oblasti negácie boli navrhnuté rôzne rozšírenia logického programovania. Doposiaľ sme rozoznávali definitné logické programy (bez negácie) a všeobecné logické programy (s negáciou ako konečným zlyhaním). Disjunktívne programy majú v hlave disjunkciu, pozri príklad 6.55. Ďalej sa študovali: Rozšírené logické programy [GeL 90], ktoré obsahovali aj klasickú negáciu. Generalizované logické programy s negáciou ako konečným zlyhaním v hlave pravidiel [APP 98]. Dynamické logické programy (postupnosti generalizovaných logických programov), ktoré majú reprezentovať evolúciu báz poznatkov [APP 98]. Dynamické programy nadväzujú na revidujúce programy (revision programs) [MaT 94]. Logické programy s novou spojkou, nazvanou epistemická disjunkcia [GeL 91]. Epistemické špecifikácie okrem klasickej negácie a epistemickej disjunkcie používajú aj modálne operátory K (viem, že) a M (možno predpokladať, že) [Gel 94]. Výborný prehľad o rôznych rozšíreniach logického programovania a ich aplikácii na reprezentáciu poznatkov možno nájsť v [BGe 94].

Sémantika falzifikácie Čitateľovi som ešte dlžný ohliadnutie do tejto kapitoly, ktoré by uzavrelo námet, že sémantiky negácie sú relevantné pre charakterizáciu falzifikácie (a tým aj hypotetického usudzovania). Dôležitý argument v prospech tohto tvrdenia dávajú výsledky z časti 6.5. Nasledujúce poznámky sú veľmi voľné námety na tému relevantnosti sémantík negácie pre pochopenie falzifikácie.

Príklad logických programov a ich sémantík nás môže inšpirovať pre analogické konštrukcie na ľubovoľných bázach poznatkov. Vráťme sa k predstave stratifikovaných teórií. V takýchto teóriách proces falzifikácie môžeme tiež rozvrstviť. Vždy, keď chceme falzifikovať hypotézu z vrstvy i, stačí nám analyzovať vety z tejto vrstvy a model tejto vrstvy. Podobne pre metaforu lokálnej stratifikácie.

Sémantika stabilných modelov nás inšpiruje hľadať alternatívne množiny hypotéz a alternatívne pristupovať aj k procesu falzifikácie. Možno chcieť uskutočniť falzifikáciu v každej alternatívnej množine hypotéz, možno byť prísnejší a falzifikovať hypotézu už na základe vykonanej falzifikácie v jedinej alternatíve. Podstatné však je, že falzifikáciu možno (inšpirujúc sa vzorom stabilných modelov) robiť vzhľadom na nejakú množinu atómov, stabilných pre danú bázu poznatkov. Sémantika dobre fundovaných modelov by nás viedla k procesu falzifikácie s rozlišovaním troch pravdivostných hodnôt a s dynamickou stratifikáciou v duchu definície 6.50.

Venovali sme sa problémom s definovaním uspokojujúcej sémantiky negácie v logických programoch. Sémanticky sme špecifikovali stratifikované a lokálne stratifikované logické programy, zaviedli sme pojem perfektného modelu. Potom sme definovali stabilné modely a dobre fundované modely. Všimli sme si ich výpočtové aspekty a experimentálne implementácie. Zaujala nás paradigma stabilného logického programovania. Ukázali sme, ako možno niektoré príklady hypotetického usudzovania charakterizovať pomocou zavedených sémantík. Na záver sme informovali o výsledkoch, ukazujúcich istú konvergenciu sémantík negácie v logických programoch a tradičných formalizácií nemonotónneho odvodzovania. Tým sme uzavreli oblúk o relevantnosti sémantík negácie pre pochopenie falzifikácie hypotéz (a pre pochopenie hypotetického usudzovania).

Kapitola 7

Dynamika poznania a usudzovania

Pre hypotetické usudzovanie je charakteristická dynamika: Formulujú sa nové hypotézy, zisťujú sa nové fakty. Nové fakty a hypotézy falzifikujú niektoré prv prijaté presvedčenia (domnienky), a tým vynucujú revízie. Nemonotónnosť je iba vonkajším prejavom dynamiky poznania a usudzovania.

V tejto kapitole sa najprv budme venovať revíziám. Ich úloha je z hľadiska dynamiky poznania kľúčová. Potom bude na rade obzretie sa dozadu: máme absolvované niektoré príklady hypotetického usudzovania, venovali sme sa nemonotónnosti v logickom programovaní a jej sémantickým charakterizáciám. Je teda čas uvažovať o nejakom jednotiacom pohľade na témy a konštrukcie, s ktorými sme sa doposiaľ stretli. Veľmi stručne poreferujeme o niektorých návrhoch na jednotiaci pohľad. Nakoniec si jeden takýto pohľad predstavíme podrobne.

7.1 Revízie

Na viacerých miestach sme už narážali na súvislosť hypotetického odvodzovania s revíziami, ale podrobnejšiu pozornosť sme vlastnému problému revízií nevenovali.

Úlohou tejto časti je jednak sformulovať kritériá, ktoré by mal spĺňať proces revízií bázy znalostí a jednak priblížiť niektoré konkrétne prístupy k vlastnému procesu revízií (a ich princípy, teoretické rekonštrukcie).

Videli sme opakovane, že revízie sú neodmysliteľnou črtou hypotetického usudzovania. V tejto kapitole sa problému budeme venovať podrobnejšie.

Pre modelovanie inteligencie je dôležitá dynamika aspoň v dvoch zmysloch slova:

- prostredie je dynamické, preto je nutná adaptácia bázy znalostí na meniace sa prostredie,
- samotný obsah bázy znalostí nebýva obvykle dokonalý, úplne adekvátny; treba ho postupne upresňovať, vylepšovať.[1]

Nutnosť a dôležitosť revízií pre výpočtové modelovanie inteligencie je zrejmým dôsledkom týchto pozorovaní.

Príklad 3.34 nám ukázal, že revízie sa stávajú zaujímavým problémom až vďaka komplikovaným vzťahom medzi vetami v báze znalostí. Odstránenie jednej vety musí byť niekedy sprevádzané odstránením ďalších viet, z ktorých možno odstraňovanú vetu odvodiť. Dodanie nejakej vety môže spôsobiť nežiadúce dôsledky, takže aj pridávanie nových viet do bázy znalostí treba kontrolovať. Príklad 3.34 ukázal, že pri modifikácii možno dosiahnuť požadovaný efekt viacerými spôsobmi. Môžeme teda zhrnúť: zaujímavým problémom na revíziách je to, že sa musia týkať aj implicitných poznatkov.

Pri štúdiu revízií prevládajú dva typy prístupov:

- abstraktná špecifikácia podmienok, za ktorých sú revízie rozumné,
- operacionálny prístup, zameraný na procesy revízie a ich implementáciu.

Treba iba dodať, že aj operacionálny prístup a implementované systémy sú cenné z hľadiska poznania princípov. Samozrejme, oba prístupy nie sú konfliktné, skôr sú komplementárne a ich kooperácia prospešná.

7.1.1 Postuláty racionálnosti

Alchourrón, Gärdenfors a Makinson študovali revízie tak, že formulovali množiny postulátov, ktorých cieľom bolo charakterizovať, kedy je modifikácia

[1] Uvedené dva prípady sa zvyknú odlišovať aj terminologicky a študujú sa vlastnosti, ktoré ich odlišujú. My sa týmto problémom venovať nebudeme.

množiny presvedčení racionálna. Na ich výskumy nadviazali mnohí autori. Dobrým prehľadom výsledkov je [GRo 95]. Postuláty tohto typu sa zvyknú nazývať AGM-postulátmi (podľa prvých písmen priezvísk priekopníkov predkladaného prístupu)[2].

Definícia 7.1 Nech K je množina viet prvorádového jazyka. Budeme ju nazývať *množinou presvedčení*, ak je uzavretá vzhľadom na dedukciu, t.j. $Cn_{FOL}(K) = K$. ■

Teda, neberieme do úvahy iba nejakú množinu explicitne zaznamenaných viet (bázu poznatkov), ale aj všetky jej deduktívne dôsledky.

Definícia 7.2 (Vloženie) Ak K je množina presvedčení a α je veta jazyka J, potom $K^+_\alpha = Cn_{FOL}(K \cup \{\alpha\})$. ■

Definovanú operáciu môžeme nazvať *vložením*, v AGM-terminológii: expansion. V definícii 7.2 sme vlastne zaviedli funkciu $+$, ktorá dostane argumenty K a α a vyprodukuje novú množinu presvedčení. Očividne ide o také pridanie nejakej vety do množiny presvedčení, pri ktorom si nerobíme žiadne problémy s konzistentnosťou.

Nižšie uvedená množina postulátov charakterizuje starostlivejšie (opatrnejšie) pridanie vety α do množiny presvedčení K. Budeme charakterizovať funkciu, ktorú nazveme *revíziou*. Doposiaľ sme termín revízia používali neformálne (a aj naďalej ho budeme používať i neformálne) ako zastrešenie rôznych možných zmien presvedčení. Aj práve zavádzaný formálny pojem má túto „univerzálnu“ vlastnosť. Ide o také pridania viet do množiny presvedčení, ktoré si môžu vynútiť odstránenie nejakých iných viet z množiny presvedčení. Na označenie funkcie revízie budeme používať symbol $*$. Argumentami tejto funkcie bude množina presvedčení K a veta α. Hodnotou funkcie bude revidovaná množina presvedčení K^*_α, ktorá spĺňa nasledujúce postuláty.[3] Najprv uvedieme postuláty, potom okomentujeme ich obsah:

1. $Cn_{FOL}(K^*_\alpha) = K^*_\alpha$,
2. $\alpha \in K^*_\alpha$,
3. $K^*_\alpha \subseteq K^+_\alpha$,
4. ak $\neg\alpha \notin K$, potom $K^+_\alpha \subseteq K^*_\alpha$,
5. ak $K_\perp$ je nekonzistentná množina presvedčení, potom $K^*_\alpha = K_\perp$, ak $\models \neg\alpha$,
6. ak $\models \alpha \equiv \beta$, potom $K^*_\alpha = K^*_\beta$,

[2]Kontrolná otázka: koľko slov, ktoré nezačínajú na „p“, je v tejto zátvorke?

[3]Samozrejme, takýchto funkcií môže byť viac. Prijímame predpoklad, že sme schopní vybrať z nich jedinú. Rôzne stratégie výberu funkcie revízie uvádzajú [GRo 95].

7. $K^*_{\alpha\wedge\beta} \subseteq (K^*_\alpha)^+_\beta$,

8. ak $\neg\beta \notin K^*_\alpha$, potom $(K^*_\alpha)^+_\beta \subseteq K^*_{\alpha\wedge\beta}$.

Intuitívny zmysel uvedených postulátov je tento:

1. Revidovaná množina presvedčení K^*_α je uzavretá vzhľadom na dedukciu.

2. Pridávané presvedčenie patrí do revidovanej množiny presvedčení.

3. Množina presvedčení, ktorá vznikne mechanickým pridaním nového presvedčenia k pôvodnej množine presvedčení, je nadmnožinou množiny presvedčení K^*_α.

4. Ak negácia pridaného presvedčenia α nepatrila do pôvodnej množiny presvedčení, (mechanické) vloženie α a opatrné pridanie α (revízia) sú rovnocenné.

5. Revízie sú obvykle sprevádzané kontrolou konzistentnosti: K^*_α je konzistentná až na tú výnimku, keď pridávaná veta α je logicky nemožná ($\neg\alpha$ je vždy platná). Keď vložíme tautologicky nepravdivú vetu, výsledkom bude nekonzistentná množina viet.

6. Pridanie dvoch ekvivalentných presvedčení dáva rovnaký výsledok.

7. Pridanie konjuncie dáva množinu presvedčení, ktorá je podmnožinou presvedčení, získaných pridaním jedného konjunktu a mechanickým vložením druhého konjunktu.

8. Ak negácia druhého konjunktu nepatrí do množiny presvedčení získaných pridaním prvého konjunktu, potom pridanie konjunkcie je rovné pridaniu jedného jej konjunktu a mechanickému vloženiu druhého konjunktu.

Ďalšia množina postulátov charakterizuje *odstránenie* (AGM-termín: contraction) vety α z množiny presvedčení K. Realizuje ho každá funkcia $-$, ktorej argumentami sú K a α, pričom hodnotou je množina presvedčení K^-_α, ktorá vyhovuje nasledujúcim postulátom:

1. $Cn_{FOL}(K^-_\alpha) = K^-_\alpha$,

2. $K^-_\alpha \subseteq K$,

3. ak $\alpha \notin K$, potom $K^-_\alpha = K$,

4. ak α nie je vždy platnou prvorádovou formulou, potom $\alpha \notin K^-_\alpha$,

5. $K \subseteq (K^-_\alpha)^+_\alpha$,

6. ak $\models \alpha \equiv \beta$, potom $K^-_\alpha = K^-_\beta$,

7. $K^-_\alpha \cap K^-_\beta \subseteq K^-_{\alpha\wedge\beta}$,

8. ak $\alpha \notin K^-_{\alpha\wedge\beta}$, potom $K^-_{\alpha\wedge\beta} \subseteq K^-_\alpha$.

Po komentároch k funkcii revízie za zmienku stojí azda iba postulát 5: ak $\alpha \notin K$, potom odstránenie α nemení K, ale po mechanickom vložení α dostaneme rozsiahlejšiu množinu presvedčení. Inklúzia $(K^-_\alpha)^+_\alpha \subseteq K$ platí, ak $\alpha \in K$.

Nakoniec ukážeme, ako vzájomne zadefinovať oba zavedené operátory. Intuitívne, odstrániť presvedčenie β z K je to isté ako ponechať v K iba to, čo nie je s β zlučiteľné (rozšírenie K o $\neg\beta$ je s β nezlučiteľné). Revidovať K pomocou β je to isté ako odstrániť $\neg\beta$ z K a potom mechanicky vložiť β.

$$\begin{aligned} K^-_\beta &= K \cap K^*_{\neg\beta} \\ K^*_\beta &= (K^-_{\neg\beta})^+_\beta \end{aligned}$$

Uvedené postuláty, samozrejme, nedefinujú selekčný mechanizmus, potrebný na realizáciu modifikácií, splňujúcich postuláty racionality. Jedným zo študovaných mechanizmov je *epistemické opevnenie* (epistemic entrenchment).[4] Epistemické opevnenie je nejaká relácia na vetách, ktorá je reflexívna a tranzitívna. Táto relácia umožňuje porovnávať vety. Hlavná myšlienka je, že pri selekcii kandidátov na odstraňovanie viet z bázy znalostí sa skôr odstraňujú tie vety, ktorých epistemické opevnenie (inými slovami váha, epistemická cena) je nižšie. Postuláty, ktoré charakterizujú epistemické opevnenie, zavádzať nebudeme. Podrobnosti v [GRo 95]. Na problémy, súvisiace s vypočítateľnosťou epistemického opevnenia upozornil Hájek [Haj 93].

Abstraktný výskum problémov revízie, smerujúci k formulácii určitých postulátov racionálnosti, je stále v popredí záujmu. Na svetovej megakonferencii o umelej inteligencii, IJCAI'99, boli prezentované tri práce s týmto zameraním [Del 99, Ker 99, CPa 99].

Teraz prejdeme k alternatívnemu prístupu. Dôraz bude na procesoch, procedúrach revízie. Samozrejme, v pozadí budú nejaké princípy. Základnou myšlienkou je, že revízia bázy poznatkov musí využívať *zdôvodnenia* týchto poznatkov.

AGM-postuláty chápu množiny presvedčení ako deduktívne uzavreté množiny viet. Nie sú schopné však zaznamenať a rešpektovať argumentačné závislosti medzi vetami.

Príklad 7.3 ([WiB 93]) Nech $P = \{$

$$\begin{aligned} p &\leftarrow \neg r, \\ q &\leftarrow \neg p\}. \end{aligned}$$

[4] Terminologické problémy ma príliš nezaťažujú. Radšej zavediem termín cirkumskripcia, ako by som mal čitateľovi slovenského textu spôsobiť problémy s prechodom k anglickým textom. Samotný termín „entrenchment“ (alebo „magické množiny“) ukazuje, že v dnešnom svete nie je tvorba terminológie škrupulózna. Na druhej strane, sú s tým nejaké problémy. Napríklad taký „entrenčment“ nie je bohviečo.

Stabilný model tohto programu je $S = \{p\}$. Môžeme preto prijať, že zodpovedajúca množina presvedčení je $K = Cn_{FOL}(S)$. Predpokladajme, že chceme revidovať P tak, aby v ňom platilo aj r. Ak by sme revíziu programu P chceli špecifikovať pomocou revízie modelu S (či množiny presvedčení K), dostali by sme $S' = \{p, r\}$ a $K' = Cn_{FOL}(S')$. Je však zrejmé, že S' nezachytáva význam programu $P' = P \cup \{r \leftarrow\}$. Stabilným modelom P' je $\{r, q\}$.

Teda, adekvátna špecifikácia revízie vyžaduje, aby sme rešpektovali závislosti medzi vetami: napríklad, pravdivosť p závisí na nepravdivosti q. Inými slovami, nepravdivosť q je argumentom v prospech pravdivosti p. Začneme sa teda vážne zaujímať o argumenty, zdôvodnenia. ■

Vo [WiB 93] je sformulovaný postulát v štýle AGM, ktorý je zameraný na rešpektovanie argumentačných závislostí pri revíziách.

Odteraz sa budeme usilovať revízie chápať tak, aby brali do úvahy aj argumenty v prospech alebo neprospech revidovaných presvedčení.

Predpokladajme, že sme odstránili nejakú vetu p. Ak p slúžila ako dôvod na akceptovanie vety q a iné platné dôvody na akceptovanie q už nie sú, musíme odstrániť q. Analogicky, po odstránení p možno akceptovať vetu q, ak medzi argumentami svedčiacimi v jej neprospech je p (samozrejme, pokiaľ sú argumenty v prospech q platné). Podobne, pridanie novej vety môže zdôvodniť pridanie alebo odmietnutie ďalších viet. Budeme analyzovať základné črty systémov na manipuláciu s dôvodmi (spravovanie zdôvodnení, udržiavanie zdôvodení v korektnom stave).[5] Najprv sa budeme venovať klasickému systému J. Doyla [Doy 79] Truth Maintenance System (TMS). Tento systém ako prvý prišiel s myšlienkou spravovania zdôvodnení.

7.1.2 Spravovanie zdôvodnení

Riešič problémov Predstavme si systém, ktorý rieši nejaké problémy. Metóda, ktorú používa, je prehľadávanie nejakého priestoru možných riešení.[6] Takýto systém nazvime riešičom problémov. Riešič problémov bude pre nás čierna skrinka. Jediné, čo nás bude na ňom zaujímať, je to, že odovzdáva systému TMS nejaké výrokovologické formuly. Tieto výrokovologické formuly sú jednak riešenia, ktoré získal, jednak ich zdôvodnenia (vety, ktorými riešenia podoprel).

Naivný riešič problémov môže ťažkopádne prehľadávať celý priestor možných riešení. Uveďme si príklad, ktorý to ilustruje, pochádza od deKleera [dKl 86]. Nech riešič rieši nasledujúcu úlohu:

[5] Reason Maintenance Systems alebo Truth Maintenace Systems alebo Belief Maintenace Systems. Priekopníckymi systémami z tohto koša boli TMS a ATMS. Ich vplyv na výskum problematiky revízií v inteligentných systémoch a na výskum hypotetického usudzovania vôbec bol a zostáva výrazný.

[6] ⟨ prehľadávanie : dodatok G ⟩

Príklad 7.4 Sú dané premenné x, y, z, nadobúdajú hodnoty z dvojprvkovej množiny $\{0, 1\}$. Sú dané tri, výpočtovo veľmi náročné, funkcie e_1, e_2, e_3, pričom $a = e_1(x)$, $b = e_2(y)$ a $c = e_3(z)$. Na a, b, c sú uložené tieto podmienky: $a \neq b$, $b \neq c$.

Obvyklý nedôvtipný prístup vyzerá takto: slepo sa vyberú hodnoty pre x, y, z, vypočítajú sa $e_1(x)$, $e_2(y)$ a $e_3(z)$. Testuje sa, či sú splnené podmienky $a \neq b$, $b \neq c$. V prípade, že nie sú, nasleduje backtracking. Po návrate k poslednému miestu vo výpočte, kde sa volila nejaká možnosť (v našom prípade nejaká hodnota buď pre x alebo y alebo z) sa zvolí alternatívna možnosť. Tento prístup má závažné vady:

1. niektoré výpočty sa zbytočne opakujú,

2. neracionálny backtracking – po zlyhaní sa výpočet vracia do posledného predchádzajúceho bodu, v ktorom boli alternatívne voľby možností; príčina zlyhania však môže byť niekde hlbšie v histórii výpočtu (nie je napríklad vylúčené, že posledným vetvením bola voľba hodnoty pre z, ale backtracking je spustený preto, že nebola splnená podmienka $a \neq b$, ktorá na z nezávisí).

■

Základné úlohy TMS Vráťme sa k príkladu 7.4, predstavme si, že si pamätáme dôvody, na ktorých je opretý fakt, že $a = b$ (povedzme, že nimi sú: $x = 0$, $y = 1$). Rozumný cieľ je vrátiť sa priamo k nim.

Riešič problémov môže klásť otázky systému TMS a vďaka jeho odpovediam zefektívniť svoje výpočty. Pri backtrackingu môže ísť priamo k príčinám zlyhaní. Jeho backtracking je riadený závislosťami. Okrem toho, TMS si pamätá prv uskutočnené odvodenia riešiča. Preto slúži pre riešič aj ako pamäť, na požiadanie mu poskytuje zaznamenané odvodenia.

Teda – úlohou TMS bolo spolupracovať s riešičom problémov a zaznamenávať „presvedčenia“, výsledky riešiča a dôvody, argumenty, na ktorých riešič problémov zakladá svoje presvedčenia. Jeho dôležitou povinnosťou bolo udržiavať všetky presvedčenia a argumenty v súlade (v koherencii).

Doyle sa explicitne vyhýbal nejakej zmienke o pravdivosti. Jednoducho sa zaujímal o to, aby presvedčenia boli podopreté nejakými argumentami, aby systém dokázal udržiavať svoje presvedčenia a argumenty v súlade. Na základe aktuálneho stavu argumentov identifikuje aktuálny stav presvedčení systému. Ak sa závery programu alebo nové argumenty dostanú do rozporu s tým, čo systém prijal, nastane revízia presvedčení.

Celá mašinéria TMS vychádza z kritiky konvenčného prístupu k usudzovaniu. Doyle kritizuje monotónnosť tradičného prístupu, jeho atomickosť (na jednotlivé vety, presvedčenia sa nazerá ako keby boli izolované) a nedostatok autoreflexie. Proti tomuto typu prístupu sa rozhoduje akceptovať nemonotónne dôvody pre nejaké presvedčenia: presvedčenie môže závisieť na tom, že neexistuje dostatok dôvodov pre presvedčenie o opaku. Na presvedčenia sa nazerá v spojitosti s dôvodmi, argumentami pre ich platnosť. Každá myšlienka je neoddeliteľná od argumentov v jej prospech.

TMS je prvým úspešným a často citovaným systémom, ktorého cieľom bolo udržiavať bázu presvedčení v koherentnom stave. Architektúra a správanie systému boli založené na dôvodoch, argumentoch v prospech platnosti presvedčení.

Stavba a funkcie TMS Formálnejšie,[7] základnými údajovými štruktúrami systému sú uzly a dôvody. Uzly zodpovedajú vetám. (Odteraz budeme hovoriť iba o vetách.) Dôvody v prospech nejakej vety sú reprezentované ako dvojice množín viet $(PRE, PROTI)$. Ku každej vete môže existovať viac takých dvojíc, rôzne argumenty môžu podopierať jedno tvrdenie.

Každá veta, ktorú systém pozná, môže byť v jednom z dvoch stavov: IN alebo OUT. Tie vety, ktoré sú v stave IN, tvoria množinu aktuálnych presvedčení systému. Patria do nej všetky vety, v prospech ktorých hovorí aspoň jeden aktuálne platný dôvod, argument. V OUT sú tie vety, ktoré aktuálne nepodporuje žiadny platný dôvod.

Presnejšie: veta P, zdôvodnená nejakou množinou M dvojíc tvaru $(PRE, PROTI)$ je v stave IN, ak existuje medzi jeho zdôvodneniami v M aspoň jedna dvojica $(PRE, PROTI)$ taká, že všetky vety v množine PRE sú platné (sú v stave IN) a žiadna veta z množiny $PROTI$ nie je platná (nie je v stave IN).

Aby pri preverovaní platnosti nevznikol nekonečný regres, musia existovať základné vety, presvedčenia. Budeme ich nazývať *základnými predpokladmi.* Tieto základné predpoklady sú dvoch druhov. Po prvé, sú to „axiómy“. Ich zdôvodnením je dvojica prázdnych množín. Po druhé, sú to nemonotónne predpoklady, podopreté (zdôvodnené) prázdnou množinou PRE a neprázdnou množinou $PROTI$. Prijímame ich dovtedy, kým sa do stavu IN nedostatne nejaká veta z množiny $PROTI$. V tej chvíli je daný nemonotónny predpoklad falzifikovaný.

Elementárnym príkladom takéhoto nemonotónneho predpokladu je veta P s dvojicou $(\emptyset, \{\neg P\})$ ako dôvodmi. Teda P platí, pokiaľ nie je známe, že by platilo $\neg P$.

Je očividné, že usudzovanie s využitím takýchto predpokladov je nemonotónne. Neskoršia platnosť $\neg P$ ruší dôvod pre platnosť P.

Treba ešte poznamenať, že zaznamenané dôvody sú v rámci systému trvalé. TMS ich neprestáva registrovať ani vtedy, keď nie sú platnými argumentami v prospech platnosti nejakého tvrdenia P. Tieto dôvody sa však po prípadnej zmene v stave aktuálnych presvedčení opäť môžu stať argumentami v prospech platnosti P.

Základnými pojmami implementácie sú uzly a zdôvodnenia. Uzly zodpovedajú potenciálnym presvedčeniam. Okrem viet (prvkov dátabázy) Doyle

[7] Tento atribút treba brať s istou rezervou, jasný výklad nie je Doylovou silnou stránkou.

rozlišuje aj pravidlá a procedúry ako možné entity, ktorým zodpovedajú uzly a v ktoré možno veriť na základe súčasného stavu presvedčení a argumentov. Operácie s pravidlami a procedúrami nie sú popísané, preto budeme ďalej predpokladať, že každý uzol zodpovedá nejakej vete. Zopakujeme dôležitý fakt: S uzlom môže byť spriahnutých viac zdôvodnení, teda viac dvojíc množín typu $(PRE, PROTI)$.

Systém robí tieto základné akcie:

- vytvorí nový uzol,
- vloží nové zdôvodnenie pre určitý uzol,
- označí uzol ako kontradikciu.[8]

Vytvorenie nového uzlu je triviálna akcia, bez závažnejších následkov. Po zvyšných dvoch akciách spúšťa TMS jeden z procesov:

- Truth Maintenace (TM),
- závislosťami riadený backtracking (ďalej iba DDB, podľa Dependency Directed Backtracking).

Činnosť TM sa naštartuje vtedy, keď sa k nejakému uzlu pridá nové zdôvodnenie. Úlohou TM je vypočítať aktuálny stav presvedčení na základe tejto modifikácie. Proces vyzerá takto:

1. Ak nové zdôvodnenie nie je platné alebo je platné, ale príslušný uzol už bol IN, netreba urobiť nič (okrem aktualizácií v údajových štruktúrach, ktoré systém obhospodaruje – v podstate ide o zoznamy konzekventov a antecedentov a ich tranzitívne uzávery).

2. Ak zdôvodnenie uzlu p je platné a p bol pôvodne OUT: p sa dostáva do stavu IN. Okrem toho treba ešte ďalej propagovať dôsledky zmeny jeho stavu: Ak niektoré uzly boli platné (IN) na základe toho, že p bol OUT a neexistuje iný dôvod pre ich platnosť, dostávajú sa do stavu OUT. Tranzitívne sa dôsledky tejto zmeny šíria dovtedy, kým sa to dá.

3. Podobne sa ošetrí symetrický prípad: pri propagácii niektorý z uzlov bol pôvodne OUT a teraz vďaka revízii „jeho argumentu“ sa dostane do IN.

Navyše, TM musí zabezpečiť, aby argumenty boli necyklické.

DDB je naštartovaný, keď sa identifikuje nejaká nekonzistentnosť. Výpočet nájde všetky platné (akceptované) základné predpoklady, ktoré sú za

[8] Doyle uvažuje aj o štvorhodnotovej pravdivostnej množine, ktorú už poznáme [Bel 76]. Pripúšťa súčasnú platnosť viet P aj $\neg P$, obe môžu byť súčasne v stave IN. Možno teda usudzovať, že na kontrolu kontradikcií sú určené špecializované procedúry, ktorých cieľom nemusí byť identifikovať všetky dvojice $P, \neg P$. Kontradikcie skôr súvisia s integritnými obmedzeniami. Pozri ďalej „nogood sets“.

túto nekonzistentnosť (nogood set, odteraz *zlá množina*) zodpovedné. Cieľom procedúry potom je dodať nové predpoklady tak, aby existujúcu kontradikciu odstránili – ukončia platnosť zlej množiny (podrobnejšie sa k problému dostaneme neskôr v tejto kapitole).

7.1.3 Racionálna rekonštrukcia TMS

Pôvodný Doylov popis TMS bol hmlistý, navyše čisto procedurálny, bez jasnej sémantiky. Teda bez možnosti spoľahlivo kontrolovať, či TMS počíta to, čo má počítať.

O precízny výklad TMS prostriedkami výrokovologického programovania sa úspešne pokúsil Elkan v [Elk 90]. Podobné prístupy možno nájsť v [PiC 89, GiM 90], pozri aj časť 7.1.5.

Uzly sú výrokové premenné. Zdôvodnenia sú klauzy výrokovej logiky tvaru $c \leftarrow \alpha \wedge \beta$, kde c je atóm, α je konjunkcia atómov, β je konjunkcia negatívnych literálov.

Definícia 7.5 *Zdôvodnenie* $c \leftarrow \alpha \wedge \beta$ *podporuje* uzol c vzhľadom na interpretáciu M práve vtedy, keď $\alpha \wedge \beta$ je pravdivá v M. ■

Necyklickosť zdôvodnení podporujúcich uzol c je formalizovaná nasledujúcou definíciou:

Definícia 7.6 Model M množiny zdôvodnení J je *uzemnený* (grounded), ak ho možno písať v tvare $M = \{c_1, \ldots, c_n\}$, kde pre každé c_j existuje zdôvodnenie $c_j \leftarrow \alpha \wedge \beta$, ktoré ho podporuje, pričom $\alpha \subseteq \{c_1, \ldots, c_{j-1}\}$. ■

Je očividné, že na množinu zdôvodnení možno nazerať ako na logický program. Stabilný model takéhoto logického programu je zhodný s uzemneným modelom zdôvodnení.

Tvrdenie 7.7 ([Elk 90]) *Model množiny zdôvodnení je stabilný práve vtedy, keď je uzemnený.*

Teda, množina uzlov, ktoré sú IN, je sémanticky charakterizovaná stabilným modelom aktuálnej množiny zdôvodnení.

Elkan presne popísal aj DDB. Tento backtracking sa inicializuje, keď systém identifikuje neprípustnú množinu predpokladov. Vieme už, že takéto množiny Doyle nazval zlými množinami: *Zlá množina* je nejaká množina literálov, ktoré nesmú súčasne platiť (byť IN).

Ak $\{b_1, \ldots, b_n\}$ je zlá množina, potom všetky modely musia splňať $\neg(b_1 \wedge \cdots \wedge b_n)$ a súčasne byť uzemnené. (Čitateľ iste vidí súvislosť s integritnými obmedzeniami.)

Stačí uvažovať jednoprvkové zlé množiny: Ak $\{b_1, \ldots, b_n\}$ je zlá množina, potom možno prijať zdôvodnenie $c \leftarrow b_1 \wedge \cdots \wedge b_n$, kde c je nový uzol (nevyskytuje sa v žiadnom inom zdôvodnení). Novou zlou množinou sa stáva $\{c\}$. Základnú myšlienku DDB vystihuje pojem nogood stratégie (nebudeme používať termín „zlá stratégia“, lebo ide o celkom dobrú stratégiu):

Definícia 7.8 *Nogood stratégia* je algoritmus, ktorého vstupom je zlá množina $\{c\}$ a množina zdôvodnení J. Jeho výstupom je množina zdôvodnení Δ, ktorá môže obsahovať nové uzly, pričom žiadny uzemnený model $J \wedge \Delta$ nespĺňa c. ■

DDB je potom taká nogood stratégia, ktorá prehľadáva smerom dozadu cez antecedenty zdôvodnení dovtedy, kým nenájde základné negatívne predpoklady. Sú to negatívne literály v antecedentoch zdôvodnení. Keďže nemôžu byť v hlave ďalšieho zdôvodnenia, hľadanie sa môže zastaviť.

Čitateľ iste postrehol, že DDB abdukuje: hľadá vysvetlenie toho, prečo sa objavila zlá množina. Nájdeným vysvetlením sú nejaké základné predpoklady. Čo je ešte zaujímavejšie, DDB sa neuspokojuje s vysvetlením, s diagnostikovaním príčiny výskytu nekonzistentnosti. Jeho cieľom je túto príčinu odstrániť, stav bázy presvedčení napraviť.

Robí to nasledovne: Predpokladajme, že množina $\{c\}$ je zlá. Úlohou DDB je vytvoriť nové zdôvodnenia, ktoré falzifikujú nejakú podmnožinu základných negatívnych predpokladov tak, aby c už nebolo odvoditeľné.

Príklad 7.9 Nech $J = \{$

$$\begin{array}{rcl} p & \leftarrow & \neg q, \\ q & \leftarrow & \neg p, \\ r & \leftarrow & \neg p, \neg s\}. \end{array}$$

Táto množina zdôvodnení má dva uzemnené modely: $M_1 = \{p\}$ a $M_2 = \{q, r\}$. $M = \{p, q, r\}$ nie je uzemnený: Žiadna klauza nepodporuje q (ako možná podpora prichádza do úvahy iba $q \leftarrow \neg p$, ale $\neg p$ nie je pravdivé v M). Podobne r nie je podopreté v M.

TMS neuvažuje alternatívne, preto zvolí jeden z modelov M_1 alebo M_2 ako aktuálny stav presvedčení (IN). Nech to je M_2. Predpokladajme, že po nejakom čase je $\{r\}$ deklarovaná ako zlá množina.[9] Vtedy algoritmus DDB nájde základné predpoklady, ktoré sú príčinou splnenia zlej množiny $\{r\}$, sú to $\{\neg p, \neg s\}$. Potom DDB vytvorí nové zdôvodnenie. Môže ním byť napríklad $s \leftarrow \neg p$. Keďže $\neg p$ je pravdivé v M_2, prestáva platiť $\neg s$ a nemožno už odvodiť r. Uzemneným modelom rozšírenej množiny zdôvodnení sa miesto M_2 stane $\{q, s\}$.

Tento príklad nie je iba jednoduchou ukážkou činnosti DDB, ale aj ilustráciou jeho nie celkom adekvátneho správania – TMS miesto revidovania M_2 mohol „prepnúť“ na M_1. ■

[9] Príklad naznačuje, že počas činnosti systému možno dodávať nielen nové uzly a nové zdôvodnenia, ale aj nové integritné obmedzenia. Teda aj konzistentnosť môže mať dynamický status.

Ukážeme si dokonca, že existuje triviálna nogood stratégia, ktorá je v rozumnom zmysle slova korektná a ktorá nevyžaduje použitie DDB. Najprv definícia.

Definícia 7.10 Nogood stratégia, ktorá zobrazuje vstup $\{c\}$ a J do výstupu $J \cup \Delta$, je *korektná*, ak

(i) každý uzemnený model J, ktorý spĺňa $\neg c$, možno rozšíriť na uzemnený model $J \cup \Delta$,

(ii) každý uzemnený model $J \cup \Delta$, ak ho zúžime iba na uzly vyskytujúce sa v J, je uzemneným modelom J spĺňajujúcim $\neg c$.

■

Lema 7.11 ([Elk 90]) *Nech J je množina zdôvodnení a $\{c\}$ je zlá množina. Nech p a q sa nevyskytujú v J.*

Nogood stratégia, ktorá zobrazuje J a $\{c\}$ na $J \cup \Delta$, kde $\Delta = \{$

$$\begin{array}{rcl} p & \leftarrow & \neg c, \\ p & \leftarrow & \neg q, \\ q & \leftarrow & p\}, \end{array}$$

je korektná.

Idea dôkazu: neexistuje uzemnený model množiny zdôvodnení $J \cup \Delta$, ktorý nespĺňa $\neg c$. ■

Do objavenia sa Elkanovej práce nebol publikovaný algoritmus, ktorý by implementoval nemonotónnny TMS a bolo by dokázané, že ten algoritmus pracuje korektne a pracuje v polynomickom čase. Elkanov výsledok v tomto smere je negatívny, v rozpore s tradičnými nezáväznými komentármi.

Tvrdenie 7.12 *Problém, či množina zdôvodnení má uzemnený model, je NP-úplný.*

Elkanovu sémantickú a zložitostnú charakterizáciu procesu TM i analýzu procesu DDB treba hodnotiť veľmi pozitívne (napriek zisteným negatívnym výpočtovým vlastnostiam). Podarilo sa mu demonštrovať, že systémy a koncepcie, založené na intuícii a procedurálnom prístupe, neumožňujú dostatočne jasné, transparetné a presvedčivé analýzy a charakterizácie ich vlastností a činnosti.

7.1.4 Spravovanie predpokladov

Rozdiel medzi TMS a ATMS Aj ATMS spolupracuje s nejakým riešičom problémov. Jeho ciele sú podobné ako ciele TMS. Sú tu však niektoré významné rozdiely. TMS neumožňuje prácu s viacerými alternatívnymi kandidátmi na riešenie. Videli sme to aj v príklade 7.9. S využitím Elkanovej rekonštrukcie TMS môžeme povedať, že TMS v každej chvíli považuje jediný stabilný model za sémantickú špecifikáciu prijatých presvedčení. V ATMS sa množina prijatých presvedčení nazýva *kontextom*, preto môžeme povedať, že TMS pracuje v každom čase iba s jediným kontextom. Po modifikácii treba v TMS kontext prepočítať.

Na rozdiel od toho, ATMS pracuje s viacerými kontextami súčasne. Hlavná myšlienka ATMS je, že paralelne udržiava alternatívne množiny presvedčení. Každú z týchto alternatív charakterizuje pomocou základných predpokladov, na ktorých sú všetky presvedčenia v danej množine založené. Spomeňme si na DDB: hľadá základné predpoklady, ktoré zapríčiňujú nekonzistentnosť. ATMS má však tieto základné predpoklady priamo reprezentované.[10] Ak je nejaká množina predpokladov spolu s danými zdôvodneniami nekonzistentná, rozčlení ju na dve množiny konzistentných predpokladov.

TMS sa zvykne charakterizovať ako systém založený na zdôvodneniach (justification-based). Na druhej strane, ATMS má už v názve, že je založený na predpokladoch. Popri manipulácii so zdôvodneniami manipuluje aj so základnými predpokladmi, t.j. množinami viet, ktoré jednoznačne určujú nejaký kontext. Všetky vety, platné v nejakom kontexte, možno odvodiť zo základných predpokladov pomocou zdôvodnení.

ATMS dokáže:

- súbežne hľadať viaceré riešenia,
- pomôcť pri porovnávaní viacerých rovnako vieryhodných riešení,
- nezávisle a alternatívne zvažovať riešenia, ktoré sú vzájomne nekonzistentné (teda, v istom zmysle slova umožňuje pracovať s nekonzistentnými predstavami),
- prepínať stavy: dočasne zmeniť pracovnú hypotézu a skúmať dôsledky tejto zmeny a neskôr sa prípadne vrátiť do pôvodného stavu.

Stavba ATMS ATMS pracuje s nasledujúcimi základnými pojmami:

- uzly (možno ich stotožniť s výrokovými symbolmi),
- zdôvodnenia (na rozdiel od TMS sú to hornovské klauzy, negatívne literály sa nepoužívajú),
- predpoklady (špeciálny typ uzlov)[11]

[10] Pozorný čitateľ si možno spomenul na kompiláciu jedných problémov na iné problémy podľa [CDS 95], spomínali sme ju v časti 4.3. Priame reprezentovanie základných predpokladov v nejakej údajovej štruktúre ATMS možno v základných črtách považovať za takúto kompiláciu. Žiaľ, zdá sa, že ide o kompiláciu do priestoru exponenciálne väčšieho. Z týchto dôvodov treba brať poznámky De Kleera o lepšej výpočtovej efektívnosti ATMS v porovnaní s TMS s rezervou. Presné zložitostné výsledky mi nie sú známe.

[11] Čitateľ pravdepodobne tuší súvis ATMS s abdukciou. Aj vyčlenenie predpokladov (v časti o TMS sme používali termín *základné predpoklady*) ako špeciálneho typu uzlov pripomína vyčlenenie abdukovateľných predikátov spomedzi množiny predikátov.

- prostredia (množiny predpokladov).

ATMS odvodzuje monotónne. Explicitná reprezentácia množín presvedčení má aj tú výhodu, že ich môžeme priamo porovnávať, čo podporuje alternatívne uvažovanie a jeho aplikáciu na konkrétnu problémovú doménu.

Aby sme sa mohli dostať ďalej, musíme sa podrobnejšie venovať prostrediam. Nejaký uzol *p platí v prostredí E* práve vtedy, keď je výrokovo-logicky odvoditeľný z $E \cup J$, kde J je množina zdôvodnení. To znamená, $p \in Cn_{VL}(E \cup J)$. Prostredie E je nekonzistentné, ak $E \cup J$ je nekonzistentná množina kláuz.

ATMS umožňuje udržiavať vzájomne nekonzistentné alternatívne množiny presvedčení a paralelne s nimi pracovať. Usiluje sa však udržiavať jednotlivé alternatívy v konzistentnom stave. Na tento cieľ sa používajú kontexty.

Kontext tvoria predpoklady *konzistentného prostredia* spolu so všetkými uzlami, ktoré z nich možno odvodiť.

Ciele a činnosť ATMS Pojem kontextu je centrálny – úlohou ATMS je efektívne určiť kontexty, ak je daná množina predpokladov a zdôvodnení.

ATMS poskytuje pre riešič problémov dve základné služby. Oznamuje, kedy sa kontext stáva nekonzistentným a odpovedá na otázku, či nejaký uzol platí v nejakom kontexte. Veľmi dôležitý koncept, umožňujúci efektívny výpočet týchto dvoch úloh, je *návestie.*

Ku každému uzlu p prislúcha nejaké návestie. Je to množina konzistentných prostredí $\{E_1, \cdots, E_n\}$ takých, že pre každé i platí $p \in Cn_{VL}(E_i \cup J)$.

ATMS zaručuje, že každé návestie N_p každého uzlu p má tieto dôležité vlastnosti:

- všetky jeho prostredia sú konzistentné,
- je korektné: pre každé prostredie E z návestia N_p platí $p \in Cn_{VL}(E \cup J)$,
- je úplné: ak pre nejaké prostredie E platí $p \in Cn_{VL}(E \cup J)$, potom E je nadmnožina nejakého protredia z návestia N_p,
- je minimálne: pre žiadne prostredie z návestia N_p neexistuje iné prostredie E' z návestia N_p také, že $E' \subset E$.

Odpoveď, či uzol platí v nejakom kontexte, dáva priamo návestie tohto uzlu. Pri testovaní konzistentnosti prostredí sa stačí venovať tým prostrediam, ktoré prislúchajú k uzlom, čo boli dotknuté práve vykonanou modifikáciou.

Návestia a kontexty vlastne dovoľujú revíziu realizovať pomocou prepínania do predvypočítaných kontextov.

Údajové štruktúry, ktoré ATMS využíva (deKleer ich nazýva ATMS-uzly)[12] sú trojice (údaj, návestie, zdôvodnenie). Údaj je výroková premenná.

[12]Treba ich odlíšiť od uzlov, o ktorých sme hovorili doteraz ako o reprezentantoch výrokových premenných.

Podrobnejšie, ATMS-uzol je štruktúra tvaru

$$(p, \{\{A_{1,1}, \cdots, A_{1,n_1}\}, \ldots, \{A_{m,1}, \cdots, A_{m,n_m}\}\},$$
$$\{\{x_{1,1}, \cdots, x_{1,n_1}\}, \ldots, \{x_{k,1}, \cdots, x_{k,n_k}\}\}),$$

kde

- p je údaj,
- $\{\{A_{1,1}, \cdots, A_{1,n_1}\}, \ldots, \{A_{m,1}, \cdots, A_{m,n_m}\}\}$ je návestie (množina konzistentných prostredí $\{A_{i,1}, \cdots, A_{i,n_i}\}$, v ktorých p platí),
- $\{\{x_{1,1}, \cdots, x_{1,n_1}\}, \ldots, \{x_{k,1}, \cdots, x_{k,n_k}\}\}$ je množina zdôvodnení.

ATMS-uzly chápe deKleer v logickej reprezentácii ako implikácie tvaru

$$\begin{aligned} (A_{1,1} \wedge \cdots \wedge A_{1,n_1}) \vee \cdots \vee (A_{m,1} \wedge \cdots \wedge A_{m,n_m}) &\rightarrow n, \\ (x_{1,1} \wedge \cdots \wedge x_{1,n_1}) \vee \cdots \vee (x_{k,1} \wedge \cdots \wedge x_{k,n_k}) &\rightarrow n. \end{aligned}$$

Rekonštrukcia činnosti ATMS Zrekapitulujme, že kľúčovými úlohami ATMS sú:

- identifikácia nekonzistentných kontextov,
- zisťovanie, či nejaký uzol (výroková premenná) platí v nejakom kontexte.

Prvá úloha nás nebude zaujímať – ide o všeobecný problém testovania konzistentnosti. Druhú úlohu chápeme ako úlohu nájsť také predpoklady, z ktorých vyplýva daná výroková premenná.

Podľa [ReK 87] činnosť ATMS možno charakterizovať ako výpočet primitívnych implikant.

Definícia 7.13 *Primitívna implikanta* množiny klauz C je klauza c taká, že $C \models c$ a pre žiadne c' také, že $c' \subset c$ neplatí $C \models c'$. ■

Charakterizujme teda činnosť ATMS: Nech J je množina zdôvodnení, p nech je otázka (uzol, výrokový symbol). Odpoveďami na túto otázku sú množiny predpokladov

$$\{A_1, \ldots, A_k : (\exists c)\ c = (A_1 \wedge \cdots \wedge A_k \rightarrow p), k \geq 0,$$
$$c \text{ je primitívna implikanta výrokového symbolu } p\}.$$

Ekvivalentne:

$$\{A_1, \ldots, A_k : \{A_1 \ldots A_k\} \text{ je minimálna}$$
$$\text{podpora pre } p \text{ vzhľadom na } J\},$$

pričom S je *minimálna podpora* pre p vzhľadom na J práve vtedy, keď $p \notin Cn_{VL}(J)$, $p \in Cn_{VL}(S \cup J)$ a neexistuje $S' \subset S$ také, že $p \in Cn_{VL}(S' \cup J)$.

Intuitívne, hľadáme minimálne kontexty, v ktorých platí uzol p. Inými slovami, hľadáme abduktívne vysvetlenie atómu p.

7.1.5 Komentáre

Pimentel a Cuadrado [PiC 89] kombinovali spravovanie množiny presvedčení, založené na zdôvodneniach so spravovaním, založeným na predpokladoch. Tak dosiahli schopnosť nemonotónneho usudzovania i schopnosť spravovať súbežne viaceré kontexty. Prístup založili na dobre vybudovanej sémantike, na stabilných modeloch (podobne ako [Elk 90]). Dôležité je, že s každým literálom asociovali množinu všetkých kontextov, v ktorých je literál pravdivý. Táto množina tvorí status (belief status) literálu. Približne jedno desaťročie pred zrodom idey SLP (stable logic programming) [MaT 99, Nie 98, Lif 99], sa rozhodujú považovať existenciu viacerých stabilných modelov za výhodu a klauzy ponímať ako obmedzenia, kladené na rozumné presvedčenia. Ak sú presvedčenia špecifikované neúplne, je prirodzené, že ich možno vyjadriť pomocou viacerých stabilných modelov. Na týchto základoch postavili teóriu dôkazu a implementáciu systému, využívajúcu štruktúru nazvanú kompresovateľný sémantický strom, reprezentujúcu množinu interpretácií.

Giordano a Martelli [GiM 90] charakterizovali TMS pomocou zovšeobecnených stabilných modelov. Tie zaviedli ako sémantickú charakterizáciu všeobecných logických programov doplnených o integritné obmedzenia. Používanie integritných obmedzení im umožnilo kontrolovať nekonzistentnosti a odstraňovať ich pomocou DDB.

Witteveen a Brewka [WiB 93] založili svoju teoretickú analýzu spravovania dôvodov na trojhodnotových dobre fundovaných modeloch. Tie dovoľujú skonštruovať jediný model množiny presvedčení. Podarilo sa im charakterizovať DDB tak, že sa vyhli ľubovoľnostiam, ktorými obvyklé DDB trpí (narazili sme na ne v časti 7.1.3). Popísali skeptickú revíziu presvedčení, zvládnuteľnú z výpočtového hľadiska: jej zložitosť je $\mathcal{O}(n^3)$.

Zaviedli sme niektoré postuláty, charakterizujúce korektné revízie. Potom sme sa venovali systémom TMS a ATMS, klasickým prístupom k implementácii procesu revízie presvedčení. Venovali sme sa predovšetkým základom, na ktorých sú založené: manipuláciám so zdôvodneniami a manipuláciám so základnými predpokladmi. V súvislosti s obidvoma systémami sme sa oboznámili aj s ich formálnymi rekonštrukciami, ktoré ukázali, ako možno dosiahnuť hlbší a presnejší vhľad do problému.

7.2 Jednotiace pohľady

Doposiaľ sme sa stretli s mnohými pohľadmi na hypotetické usudzovanie. Venovali sme sa rôznym formalizáciám tohto typu usudzovania. Hrozí, že kvôli stromom neuvidíme les, i keď sme sa nezriedka dotýkali súvislostí nemonotónneho usudzovania a revízií i súvislostí rôznych druhov hypotetického usudzovania. Teraz je naším cieľom pozrieť sa na les. Žiaľ, vieme o ňom oveľa menej ako o stromoch. Možno základný problém je nájsť miesto, odkiaľ by sme mali dobrý výhľad.

Už dve desaťročia sa v umelej inteligencii venuje veľká pozornosť nemonotónnemu usudzovaniu. Tento druh usudzovania sa považuje za kľúč pre pochopenie zdravého rozumu i za základ adekvátnej reprezentácie poznatkov pre výpočtové modely inteligencie. Z času na čas sa však ozve hlas, že dôležité je pozrieť sa hlbšie a spýtať sa, prejavom čoho je nemonotónnosť usudzovania. Jeden takýto pohľad sme mohli nájsť v kapitole 3. Najpregnantnejšie vyjadril potrebu nazrieť poza nemonotónnosť van Benthem [vBe 87]: nemonotónnosť je skôr symptómom, než esenciou hypotetického usudzovania.[13]

Plán pre túto časť textu je nasledovný: Začneme (časť 7.2.1) stručnými informáciami o skúmaní vzťahov jednotlivých typov hypotetického usudzovania k revíziám, k sémantikám logického programovania i o skúmaní vzťahov medzi nimi navzájom.

Potom, v časti 7.2.2, prejdeme k informáciám o výskumoch, ktoré si za cieľ postavili práve vytvorenie jednotiacej teórie hypotetického (nemonotónneho) usudzovania. Táto časť obsahuje vlastne iba veľmi stručný sprievod po (vybranej) literatúre. Cieľom je poskytnúť záujemcom o hlbšie štúdium prvé smerníky a ostatným základný dojem.

Časť 7.3 je venovaná pôvodným výsledkom, podávajúcim jednotiaci pohľad na hypotetické usudzovanie.

7.2.1 Niektoré vzťahy

Videli sme už, že sémantiky logického programovania umožňujú precízne vyjadriť správanie systémov na spravovanie dôvodov a systémov na spravovanie základných predpokladov. Možno to považovať za dobrý signál o sile, flexibilnosti a aplikovateľnosti sémantík negácie. Máme dobré dôvody pokladať tieto sémantiky za slušné formálne nástroje, vhodné na pochopenie hypotetického usudzovania a reprezentácie poznatkov.

Od systémov na spravovanie dôvodov a presvedčení sa dostávame k abdukcii: Výpočtu abdukcie pomocou TMS sa venovali [KaM 90a, KaM 90b,

[13] Van Benthem síce hovoril o neštandardnom usudzovaní, ale iba z hľadiska toho, čo sa štandardne študovalo v logike. Ak sa na vec pozrieme z hľadiska common sensu, hypotetické usudzovanie je ten štandardný prípad.

SaI 91]. Hovorili sme už o tom, že činnosť ATMS spočíva v abdukovaní. Po pochopení súvislostí medzi TMS a sémantikami negácie preto neprekvapí, že sú známe pokusy aplikovať sémantiky negácie na charakterizáciu abdukcie a hypotetického usudzovania vôbec [InS 96, Den 94, GiM 90, KaM 90b, KKT 93, EGL 97, Dun 91]. Navyše, súčasné práce o abdukcii, revíziách atď. spravidla nemožno pochopiť bez znalosti niektorej zo sémantík negácie.

Načrtli sme už, že abdukciu možno poňať všeobecnejšie – tak, že revízia tvorí jej integrálnu súčasť. Teraz tomuto motívu budeme venovať trochu pozornosti.

Abdukcia a revízie Doposiaľ sme abdukciu redukovali na dedukciu pozorovaní z vygenerovaných hypotéz a z teórie, ktorá je v pozadí. Pritom sa predpokladala konzistentnosť pôvodnej teórie a vygenerovaných hypotéz. (Samozrejme, konzistentnosť vysvetlených pozorovaní s hypotézami a s teóriou v pozadí je zrejmá z definície.)

Často však pri vysvetľovaní ide o objasnenie, prečo sú pozorovania v konflikte s našimi presvedčeniami, teóriami. Je teda primerané považovať za kľúč k abdukcii priame začlenenenie revízií do koncepcie abdukcie. Takýto prístup možno nájsť napríklad v [Bou 95], odkiaľ pochádzajú nasledujúce tri postuláty pre abdukciu:

Nech β je pozorovanie a α je (abduktívne) vysvetlenie. Množina presvedčení K hrá rolu teórie v pozadí.

(i) $\alpha \in K$ práve vtedy, keď $\beta \in K$, podobne $\neg\alpha \in K$ práve vtedy, keď $\neg\beta \in K$

(ii) $\beta \in K^*_\alpha$

(iii) $\beta \in (K^-_\beta)^*_\alpha$

Intuitívny zmysel:

(i) Pozorovanie a jeho vysvetlenie buď súčasne platia alebo súčasne neplatia v množine presvedčení.

(ii) Ak do množiny presvedčení vložíme vysvetlenie α, pozorovanie β do nej tiež patrí.

(iii) Ak z množiny presvedčení odstránime dané pozorovanie β, ale vložíme do nej vysvetlenie α, v takto revidovanej množine presvedčení naďalej platí β.

Zavedeniu revízií do abdukcie sa venovali aj [Ali 00, Buc 97].

Ďalšie súvislosti Samozrejme, podobným spôsobom možno zovšeobecniť aj iné typy hypotetického odvodzovania.

Už sme spomenuli praktickú potrebu kombinovať indukciu a revízie. De Raedt a Bruynooghe ukázali v [dRB 92], že indukcia (inkrementálne učenie pojmov) je špeciálnym prípadom revízií presvedčení. Nadviazali pritom na výskumy Shapira, ktorý využil súvislosti medzi revíziami a indukciou pri riešení problému identifikovania chýb v logických programoch (Model Inference System [Sha 83]).

Aj vzťah defaultov a revízií sme už zdôrazňovali. Morris v [Moi 88] ukázal, že problém anomálnych extenzií defaultových teórií možno vyriešiť, ak sa preformuluje do slovníka TMS. Zdôvodnenia možno vyjadriť ako defaultové pravidlá (čitateľa, ktorý pochopil súvislosť logických programov a zdôvodnení a ktorý vidí aj súvislosti medzi defaultovými teóriami a logickými programami, to neprekvapuje a vie si predstaviť aj návrh formalizácie). Po takejto formalizácii možno chápať defaultové teórie ako sémantickú charakterizáciu TMS. Defaultové usudzovanie ponímajú ako revízie presvedčení napríklad [ZZC 98] alebo [Bou 94].

Sú známe aj pokusy všeobecným spôsobom charakterizovať nemonotónne (hypotetické) usudzovanie a ukázať, že revízia presvedčení a nemonotónne usudzovanie sú prepojené, že sú to „dve strany jednej mince“ [MaG 91]. Rôznymi spôsobmi túto všeobecnú tézu podporujú aj ďalšie výskumy ako napríklad [Boc 98].

7.2.2 Integrácie

Tu si spomenieme niektoré pokusy o jednotiaci pohľad na hypotetické usudzovanie, ktorých ambície sú univerzálnejšie – ide im o viac, než o nachádzanie súvislostí medzi špeciálnymi prípadmi hypotetického usudzovania.

Najprv sa budeme venovať pokusom o všeobecnú charakterizáciu hypotetického usudzovania pomocou postulátov. Ich cieľom je vyjadriť podmienky, za ktorých je toto usudzovanie racionálne (tento štýl už poznáme: definícia kumulatívnosti Cn, kapitola 3, postuláty racionálnosti revízií, táto kapitola).

Azda prvý použil tento prístup Gabbay [Gab 85] s cieľom položiť teoretické základy pre štúdium nemonotónneho usudzovania.

Pod teoretickými základmi rozumel skúmanie tohto problému: Majme nejakú implementáciu nejakého hypotetického usudzovania. Môžeme pozorovať, ako sa táto implementácia správa: z akých predpokladov odvodzuje aké závery. Prvá otázka je, či toto správanie možno opísať nejakými pravidlami (teda, či správanie implementácie je dôsledné, konzistentné). Ďalšia otázka je, či môžeme nad toto správanie nadhliadnuť a charakterizovať nájdené pravidlá nejakými všeobecnejšími formálnymi vlastnosťami, ktoré budú pre nás zárukou rozumného, spoľahlivého, dôveryhodného správania. S týmto cieľom na očiach možno nadviazať buď na Tarského tradíciu a študovať vlastnosti operátora Cn alebo na tradíciu Gentzenovu a študovať vlastnosti odvodzovacích pravidiel (už sme o tom hovorili v kapitole 3). Po Gabbayovi touto cestou išli napríklad [Mak 89, Woj 88, KLM 90, GeP 92, LeM 92, GMa 94, Geo 96, Geo 97, Boc 99]. Z ďalších výskumov spomeňme [Fla 98], ktorý sa venoval aproximácii vzťahov odvodenia. Pomocou takýchto abstraktných vlastností možno charakterizovať rôzne príklady hypotetického usudzovania. Už sme si uviedli (v časti 4.1), že defaultové usudzovanie (na rozdiel od ostatných základných formalizácií nemonotónneho usudzovania) nie je kumulatívne. Sémantiky negácie v logickom programovaní klasifikoval z hľadiska spĺňania postulátov tohto typu [Dix 95a, Dix 95b].

Štúdiom abstraktných vlastností hypotetického usudzovania možno dosiahnuť napríklad výsledky tohto typu: Charakterizuje sa odvodenie s nejakými rozumnými vlastnosťami. Potom sa študujú buď teoreticko-modelové prístupy (napríklad založené na preferovaných modeloch) alebo prístupy založené na odvodzovacích pravidlách a zisťuje sa, či majú požadované vlastnosti. Ak nemajú, definujú sa také podtriedy, ktoré rozumné vlastnosti majú. Porovnávajú sa rôzne abstraktne definované relácie vyplývania alebo operácie odvodenia. Zisťuje sa, ktoré konkrétne formalizácie majú ktoré z definovaných abstraktných vlastností.

V istom zmysle slova prístup, založený na postulátoch, poskytuje metateoretickú charakterizáciu hypotetického usudzovania: charakterizujú sa triedy teórií, ktoré spĺňajú sformulované postuláty.

Ďalší druh prístupov spočíva v tom, že sa v nejakom jazyku vytvorí nejaký formalizmus a ukáže sa o ňom, že je natoľko silný, že jeho špeciálnym prípadom sú iné formalizácie nemonotónneho (hypotetického) usudzovania.

Jedným z príkladov takéhoto silného formalizmu je rámec, ktorý modeluje argumentáciu [BDK 97]. Je daná báza poznatkov. Ďalej, sú dané nejaké pravidlá, ktoré zavádzajú argumenty. Na týchto pravidlách je definovaná relácia preferencie. Argumenty a protiargumenty sa vzájomne atakujú. Dá sa ukázať, že pomocou takého rámca možno vyjadriť tradične študované príklady nemonotónneho usudzovania.

Lifschitz [Lif 94] vytvoril logiku minimálnej viery a negácie ako zlyhania. Ide vlastne o modálnu logiku s dvoma modálnymi operátormi. (S modálnymi operátormi sme sa prvýkrát stretli v kapitole 2.) Minimálnu vieru i negáciu ako zlyhanie definuje pomocou modálnych operátorov. Ukázal, že špeciálnym prípadom jeho logiky sú niektoré formy defaultovej logiky, cirkumskripcie, logického programovania.

Przymusinski [Prz 97] zaviedol autoepistemickú logiku viery a poznania, ktorej špeciálnym prípadom sú základné formalizmy ako autoepistemická logika, cirkumskripcia, predpoklad uzavretého sveta a základné sémantiky negácie v logickom programovaní.

Przymusinski zosilnil autoepistemickú logiku, ktorá pracuje s jedným modálnym operátorom poznania K („viem, že ...“), o ďalší modálny operátor viery B („verím, že ...“, „pripúšťam, že ...“). Toto rozšírenie jazyka je východiskom pre rozlíšenie dvoch postojov. Umožňuje formalizovať usudzovanie o vetách, o ktorých vieme, že sú pravdivé, aj o ktorých predpokladáme, že by mohli byť pravdivé a preto im veríme, prijímame ich ako hypotézy. V nasledujúcej časti sa k Przymusinského logike ešte vrátime.

V tejto časti sme poinformovali o výskumoch vzájomných vzťahov revízií, abdukcie, logického programovania, defaultových teórií, indukcie. Potom sme pozornosť venovali metateoretickej charakterizácii operátora nemonotónneho (hypotetického) odvodenia pomocou jeho vítaných, rozumných vlastností. Nakoniec sme spomenuli niektoré formalizmy značnej sily – ich špeciálnymi prípadmi sú rozhodujúce formalizácie nemonotónnsho usudzovania.

7.3 Dynamické Kripkeho štruktúry

V tejto časti opíšeme pomerne silnú sémantickú štruktúru. Na obvyklých Kripkeho štruktúrach dovolíme nejaké transformácie a operácie. Naším cieľom je ukázať, že táto sémantická konštrukcia dovoľuje charakterizovať hypotetické usudzovanie.

Predstavme si nemonotónny operátor Cn_{nm} a dve množiny viet A, B, pre ktoré platí $A \subset B$ a $Cn_{nm}(A) \not\subseteq Cn_{nm}(B)$. To znamená, že existujú vety ϕ, ψ také, že $\phi \in B \setminus A$ a $\psi \in Cn_{nm}(A) \setminus Cn_{nm}(B)$ (môžeme to vidieť tak, že ϕ patrí medzi nové vety, rozširujúce A a ψ medzi tie, ktoré treba odstrániť z $Cn_{nm}(A)$ kvôli vloženiam do A).

Práve opísanú situáciu môžeme charakterizovať sémanticky pomocou dynamickej Kripkeho štruktúry. Najprv si osviežme pojem Kripkeho štruktúry, ktorý poznáme z kapitoly 2, definícia 2.28. Osviežime si základnú myšlienku, nie formálnu definíciu: Pracujeme s mnohými (všetkými) možnými interpretáciami nejakého jazyka. Voláme ich možnými svetmi. Kripkeho štruktúry dovoľujú vidieť nejaké súvislosti medzi interpretáciami. Z daného možného sveta sa možno dostať do iných možných svetov vďaka relácii dostupnosti. Relácie dostupnosti umožňujú rozlíšiť niečo, čo by sme mohli chápať ako epistemické alternatívy. Ak použijeme príklad z kapitoly 2, relácia dostupnosti smerovala od danej databázy s neúplnou informáciou k jej možným zúplneniam. Pojem pravdivosti, splňovania, modelu možno definovať s využitím relácie dostupnosti. Napríklad, nejaká formula nutne platí vo svete w práve vtedy, keď platí v každom svete dostupnom z w.

Teraz načrtneme, ako možno charakerizovať revízie z prvého odstavca pomocou dynamických Kripkeho štruktúr (DKS). DKS si predstavme ako dvojicu $(\mathcal{F}, \mathcal{K})$. $\mathcal{K}$ je Kripkeho štruktúra, pomocou ktorej je definovaný vzťah $\models_{nm}$. Tento vzťah môžeme chápať ako sémantickú špecifikáciu operátora Cn_{nm}. $\mathcal{F}$ je množina nejakých transformácií možných svetov. Ak $f \in \mathcal{F}$, potom $f : W \longrightarrow W$.

Predstavme si teraz $w \in W$ a $f(w) \in W$ také, že $w \not\models_{nm} \phi$, ale $f(w) \models_{nm} \phi$. To znamená, že f zodpovedá nejakému vloženiu. Podobne, predpokladajme, že $w \models_{nm} \psi$ a $f(w) \not\models_{nm} \psi$. To znamená, f zodpovedá aj nejakému odstráneniu (povedzme, že vynútenému spomínaným vložením).

Základná intuícia spojená s dynamickými Kripkeho štuktúrami je teda nasledovná: DKS pozostáva z dvoch zložiek, z jednej statickej (je ňou obyčajná Kripkeho štruktúra) a z druhej dynamickej (je ňou množina transformácií možných svetov na iné možné svety). Pre nemonotóne usudzovanie sú kľúčové tie situácie, keď sa získavajú nové poznatky a keď kvôli tomuto pridaniu nových poznatkov sme nútení revidovať niektoré poznatky, ktoré sme akceptovali predtým. DKS poskytujú sémantickú charakterizáciu práve týchto situácií. Transformácia jedného možného sveta na iný možný svet predstavuje zmenu v našom poznaní. Môžeme to vidieť tak, že transformácie zodpovedajú generovaniu hypotéz a revíziám (nejaká hypotéza môže byť pravdivá vo svete $f(w)$, ale nie vo w a naopak). Transformácia niekedy „mení" aj reláciu dostupnosti – presnejšie: pre dané svety w_1, w_2, reláciu dostupnosti ρ a transformáciu f môže platiť: $(w_1, w_2) \in \rho$, $(f(w_1), f(w_2)) \notin \rho$ alebo naopak. To znamená – v prípadoch, keď je operátor odvodenia Cn špecifikovaný v závislosti na relácii dostupnosti, táto transformácia môže spôsobiť nemonotónnosť Cn. V istom zmysle slova DKS je konštrukciou vysvetľujúcou nemonotónnosť usudzovania. Z hľadiska DKS: ak nemonotónnosť je symptóm, potom generovanie hypotéz (získavanie nových poznatkov) a revízie vynútené týmto generovaním sú esenciou (neštandardného, hypotetického, na kontexte závislého, vyvíjajúceho sa usudzovania).

Dynamické Kripkeho štruktúry boli zavedené v [Sef 97], kde sa s ich pomocou charakterizovali vloženia a odstránenia viet do/z databázy a predpoklad uzavretého sveta. Teraz si všimneme, ako možno DKS využiť na sémantickú charakterizáciu Przymusinského autoepistemickej logiky poznania a viery [Prz 97] (ďalej AELKB). Táto charakterizácia bola prezentovaná v [Sef 99]. Okolnosť, že špeciálnym prípadom AELKB sú viaceré základné formalizácie nemonotónneho usudzovania, posilňuje predpoklad o tom, že DKS je vhodným nástrojom na unifikujúcu sémantickú charakterizáciu hypotetického usudzovania.

Môžeme prejsť k detailom:

7.3.1 AELKB-štruktúry

Najprv opíšeme AELKB. Predpokladáme výrokovologický jazyk J so spojkami $\neg, \rightarrow, \wedge, \vee$, spočítateľnú množinu výrokových premenných

$$\mathcal{P} = \{p_1, \dots, p_n, \dots\}$$

a dva špeciálne symboly: ϵ (prázdna formula), $\perp$ (*nepravda*). Výrokovologický atóm, literál a formula sú definované ako obvykle.

Teraz rozšírime jazyk J: jazyk J_A obsahuje navyše aj operátory K a B. Každý atóm, literál a formula jazyka J je (objektívny) atóm, literál a formula jazyka J_A. Ak ϕ je formula jazyka J_A, potom $B\phi$ a $K\phi$ sú (subjektívne) atómy a $B\phi$, $K\phi$, $\neg B\phi$ a $\neg K\phi$ sú (subjektívne) literály jazyka J_A. Každý (subjektívny) literál je *formula*. Ak ϕ a ψ sú formuly, potom $\phi \wedge \psi$, $\neg\phi$ sú formuly (budeme tiež hovoriť, že sú J_A-formuly). Formuly, ktoré obsahujú operátory K alebo B sa nazývajú subjektívnymi.

Definícia 7.14 (AELKB-teória [Prz 97]) Pod teóriou v jazyku J_A rozumieme (prípadne nekonečnú) množinu viet tvaru
$\beta_1 \wedge \dots \wedge \beta_k \wedge B\phi_1 \wedge \dots \wedge B\phi_l \wedge K\psi_1 \wedge \dots \wedge K\psi_m \rightarrow \alpha_1 \vee \dots \vee \alpha_n \vee B\chi_1 \vee \dots \vee B\chi_r \vee K\tau_1 \vee \dots \vee K\tau_s$,
kde α_i, β_i sú výrokovologické atómy, ϕ_i, ψ_i, χ_i, τ_i sú ľubovoľné formuly jazyka J_A.

Teóriu v jazyku J_A budeme volať aj *AELKB-teóriou.* ■

Množinu všetkých atómov (objektívnych aj subjektívnych) jazyka J_A označme ako $\mathcal{P}_{J_A}$. *Interpretáciou* jazyka J_A je ľubovoľná podmnožina $\mathcal{P}_{J_A}$. Prijímame, že ohodnotenie J_A-formuly v interpretácii $\mathcal{I}$ možno definovať presne ako dvojhodnotové výrokovologické ohodnotenie:

Definícia 7.15 Nech $\mathcal{I}$ je interpretácia:

- ak ϕ je atóm (objektívny alebo subjektívny), potom $val_{\mathcal{I}}(\phi) = t$ práve vtedy, keď $\phi \in \mathcal{I}$,
- ak ϕ je literál $\neg\psi$, potom $val_{\mathcal{I}}(\phi) = t$ práve vtedy, keď $\psi \notin \mathcal{I}$,
- v ostatných prípadoch sa $val_{\mathcal{I}}(\phi)$ počíta podľa pravidiel vyhodnocovania formúl výrokovej logiky.

Ak X je množina formúl, potom $val_{\mathcal{I}}(X) = t$ práve vtedy, keď pre každé $\phi \in X$ platí $val_{\mathcal{I}}(\phi) = t$. Vtedy hovoríme, že $\mathcal{I}$ je *modelom* X (X je *splnená* v $\mathcal{I}$). ■

Použijeme našu známu konvenciu: Ak $\mathcal{I}$ je interpretácia, budeme ju označovať aj ako $\mathcal{I} \cup \mathcal{N}$, kde $\mathcal{N} = \{\neg\phi : \phi \in \mathcal{P}_{J_A} \setminus \mathcal{I}\}$.

Definícia 7.16 Nech sa interpretácie $\mathcal{I}$ a $\mathcal{J}$ zhodujú na subjektívnych literáloch.

$\mathcal{I} \prec \mathcal{J}$ práve vtedy, keď pre každý objektívny atóm α platí: ak $\alpha \in \mathcal{I}$, potom $\alpha \in \mathcal{J}$.

Nech Σ je nejaká množina interpretácií a $\mathcal{I} \in \Sigma$. Budeme hovoriť, že $\mathcal{I}$ je *minimálna* v Σ práve vtedy, keď žiadna interpretácia $\mathcal{J} \in \Sigma$ rôzna od $\mathcal{I}$ nespĺňa podmienku $\mathcal{J} \prec \mathcal{I}$.

Ak formula ϕ je pravdivá vo všetkých minimálnych modeloch nejakej AELKB-teórie T, potom hovoríme, že ϕ *minimálne vyplýva* z T (značenie: $T \models_{min} \phi$). ■

Pravdivostné hodnoty subjektívnych atómov sú nezávislé na pravdivostných hodnotách formúl, ktoré sú argumentami operátorov K alebo B. Intuícia, ktorá je za tým:

Predpokladajme, že špecifikujeme nejaký výpočtový model inteligencie. Model, prejavujúci sa nejakým inteligentným správaním, budeme v tejto chvíli nazývať *agentom.* Tento termín budeme používať iba preto, že by sa nám ťažko pracovalo so spojeniami ako „výpočtový model inteligencie vie, že“ a podobnými. Predpokladajme, že tento agent má schopnosť introspekcie: používa formuly tvaru $K\phi$ a $B\phi$, vie s nimi manipulovať a vyhodnocovať ich pravdivostné hodnoty. Je rozumné očakávať, že agent pokladá $K\phi$ ($B\phi$) za pravdivé vtedy, keď *vie*, že ϕ (alebo *verí*, že ϕ) nezávisle na tom, či ϕ je pravdivé alebo nepravdivé.[14] To, čo agent pozná alebo to, čomu verí, budeme reprezentovať pomocou množín subjektívnych atómov.

Definícia 7.17 Nech $\mathcal{I}$ je interpretácia výrokových premených a S nech je množina subjektívnych atómov. Definujeme funkciu val, ktorá priraďuje nejakú hodnotu z množiny $\{f, t\}$ každej dvojici $(\mathcal{I}, S)$ a každej J_A formule:

- ak ϕ je objektívny atóm, potom $val_{\mathcal{I}}^{S}(\phi) = t$ práve vtedy, keď $\phi \in \mathcal{I}$,
- ak ϕ je subjektívny atóm, potom $val_{\mathcal{I}}^{S}(\phi) = t$ práve vtedy, keď $\phi \in S$,
- ak ϕ je $\neg\psi$, potom $val_{\mathcal{I}}^{S}(\phi) = t$ práve vtedy, keď $val_{\mathcal{I}}^{S}(\psi) = f$,
- ak ϕ je $\psi \wedge \tau$, potom $val_{\mathcal{I}}^{S}(\phi) = t$ práve vtedy, keď aj $val_{\mathcal{I}}^{S}(\psi) = t$, aj $val_{\mathcal{I}}^{S}(\tau) = t$.

Zafixujme nejaké S. Definujme: $val^{S}(\tau) = t$ práve vtedy, keď pre každú interpretáciu $\mathcal{I}$ je $val_{\mathcal{I}}^{S}(\tau) = t$. Nech ďalej X je množina J_A-formúl. Potom $val_{\mathcal{I}}^{S}(X) = t$ práve vtedy, keď $val_{\mathcal{I}}^{S}(\phi) = t$ pre každé $\phi \in X$. Hovoríme, že $\mathcal{I}$ je *modelom* X (X je *splnené* v $\mathcal{I}$). ■

Pravdivostné hodnoty subjektívnych formúl sme definovali pomocou nejakej množiny subjektívnych literálov, ktoré sú evidenciou o poznatkoch alebo vierach (nejakého predpokladaného agenta). Bude však rozumné pripustiť iba také množiny, ktoré majú rozumné vlastnosti (aby sme ich mohli považovať za poznatky a viery, presvedčenia rozumného agenta).

Definícia 7.18 ($Cn_A(T)$ [Prz 97]) Nech T je AELKB-teória, $Cn_A(T)$ je najmenšia množina formúl, ktorá obsahuje teóriu T, všetky inštancie nasledujúcich axióm:

$$\neg B\bot,$$
$$B(\phi \to \psi) \to (B\phi \to B\psi),$$

je uzavretá vzhľadom na výrokovologické odvodzovanie Cn_{VL} a vzhľadom na pravidlo $\dfrac{\phi}{B\phi}$. ■

$Cn_A(T)$ považujeme za množinu všetkých formúl, ktoré sú odvoditeľné z danej AELKB-teórie T. Každá množina $Cn_A(T)$ predstavuje rozumnú množinu presvedčení, založených na T.

„Introspektívny obsah“ nejakej AELKB-teórie T budeme teraz definovať ako AELKB-teóriu T^*, ktorú nazývame *statickou autoepistemickou expanziou* (skratka SAE).

[14] Samozrejme, čo to znamená, že niekto niečo vie, je zložitá filozofická otázka. Nás trápiť nebude – nájdeme všetko v nejakom zozname (množine).

Definícia 7.19 (Statická autoepistemická expanzia) Teória T^* sa nazýva *statickou autoepistemickou expanziou* AELKB-teórie T práve vtedy, keď $T^* = Cn_A(T \cup \{K\phi : T^* \models \phi\} \cup \{\neg K\phi : T^* \not\models \phi\} \cup \{B\phi : T^* \models_{min} \phi\})$. ■

Rozlišujeme teda tri úrovne logickej charakterizácie AELKB-teórií:

- dvojhodnotové modely (a Cn_{VL}-odvodenie),
- Cn_A-odvodenie,
- statické autoepistemické expanzie.

Teraz je naším cieľom charakterizovať SAE AELKB-teórií pomocou Kripkeho štruktúr. Autoepistemickú logiku s jedným operátorom K charakterizoval pomocou Kripkeho štruktúr Moore [Moo 85] konštruktér tejto logiky. AELB-teórie, fragment AELKB-teórií obsahujúci iba B-operátor, charakterizovali pomocou Kripkeho štruktúr [BDP 97].

Najprv zavedieme všeobecný pojem Kripkeho štruktúry. Neskôr budeme pracovať s jeho špecializáciami.

Definícia 7.20 *Kripkeho štruktúra* je trojica (W, R, m), kde W je množina možných svetov, R je množina relácií. Pre $\rho \in R$ je $\rho \subseteq W \times W$. Funkcia m každému možnému svetu priradí nejakú interpretáciu (jazyka J_A). ■

Definícia 7.21 *Úplný $S5$-rámec* je Kripkeho štruktúra (W, ρ) taká, že $\rho = W \times W$.[15] ■

V úplnom $S5$-rámci je každý možný svet dostupný z každého možného sveta, teda úplný $S5$-rámec je jednoznačne determinovaný množinou možných svetov W.

Definícia 7.22 Množina J_A-formúl S je *stabilná* práve vtedy, keď

- $S = Cn_{VL}(S)$,
- ak $\phi \in S$, potom $K\phi \in S$,
- ak $\phi \notin S$, potom $\neg K\phi \in S$.

■

Teraz sme pripravení na konštrukciu vhodnej Kripkeho štruktúry, umožňujúcej charakterizáciu stabilných autoepistemických expanzií AELKB-teórií. Možnými svetmi budú úplné $S5$-rámce a jedna z relácií dostupnosti smeruje k minimálnym modelom.

---

[15] Kvôli jednoduchosti píšeme ρ miesto $\{\rho\}$ a možné svety stotožníme s interpretáciami. (Formálne, funkcia m je identita, ale jej explicitné zaznamenávanie vypúšťame.)

Nakoniec terminologická poznámka - pomocou $S5$ sa označuje jedna z klasických modálnych logík. Podrobnosti napríklad v [Luk 90] alebo v [MaT 93].

Odteraz predpokladáme, že J_A má iba konečný počet výrokových premenných a predpokladáme *konečné* množiny *konečných* intepretácií.

Budeme definovať trojhodnotové ohodnotenie objektívnych formúl. Motivujeme to nasledovne: Každá konzistentná SAE ľubovoľnej AELKB-teórie T obsahuje presne jeden z komplementárnych literálov $K\phi, \neg K\phi$ pre každú J_A-formulu ϕ. (Ideálne introspektívny agent vie o každej formuli ϕ, či $K\phi$ alebo $\neg K\phi$). Chystáme sa definovať zobrazenie m, ktoré každému možnému svetu w priradí nejakú interpretáciu m_w. Táto interpretácia m_w by mala zaručiť, aby pre každú formulu ϕ platilo buď $K\phi$ alebo $\neg K\phi$. Niekedy je však prirodzené prijať aj $m_w(\neg K\phi) = t$ aj $m_w(\neg K\neg\phi) = t$. (Náš agent nie je vševediaci, je „iba“ ideálne introspektívny.) Musí platiť $m_w(\phi) \neq t$ a aj $m_w(\neg\phi) \neq t$. Preto zavedieme tretiu pravdivostnú hodnotu u (nedefinovaný, neznámy). Pre subjektívne formuly budeme používať dvojhodnotové ohodnotenia.

Definícia 7.23 (AELKB-štruktúra) Nech Int je množina všetkých interpretácií nejakej AELKB-teórie v jazyku J_A.

AELKB-štruktúra je trojica (W, R, m), kde $W = \mathcal{P}(Int)$ je množina všetkých podmnožín množiny Int, $R = \{\rho_1, \rho_2\}$, pričom

$$\begin{aligned}\rho_1 &= \{(w, w') : w \neq w' \wedge (\exists \mathcal{I} \in Int)\ w = w' \cup \{\mathcal{I}\}\},\\ \rho_2 &= \{(w, w') : w' = \{\mathcal{I} : \mathcal{I} \text{ je minimálny vo } w\}\}.\end{aligned}$$

Napokon, m definujeme takto:[16]

- pre každú objektívnu formulu ϕ: $m_w(\phi) = t$, ak $(\forall \mathcal{I} \in w)\ val_{\mathcal{I}}(\phi) = t$, $m_w(\phi) = f$, ak $(\forall \mathcal{I} \in w)\ val_{\mathcal{I}}(\phi) = f$, inak $m_w(\phi) = u$,
- $m_w(K\phi) = t$ práve vtedy, keď $m_w(\phi) = t$,
- $m_w(\neg K\phi) = t$ práve vtedy, keď $m_w(\phi) \neq t$,
- $m_w(B\phi) = t$ práve vtedy, keď $\forall w'\ ((w, w') \in \rho_2 \rightarrow m_{w'}(\phi) = t)$, inak $m_w(B\phi) = f$,
- ak ϕ a ψ sú J_A-formuly, potom $m_w(\phi \wedge \psi) = min\{m_w(\phi), m_w(\psi)\}$, pričom na pravdivostných hodnotách je definované toto usporiadanie: $f < u < t$,
- ak ϕ je J_A-formula, potom $m_w(\neg\phi) =$
  - t, ak $m_w(\phi) = f$,
  - u, ak $m_w(\phi) = u$,
  - f, ak $m_w(\phi) = t$.

[16] m priradí interpretáciu $m(w)$ ku každému možnému svetu w. Aplikáciu interpretácie $m(w)$ na formulu ϕ označíme ako $m_w(\phi)$.

Ak T je AELKB-teória, potom $m_w(T) = t$ práve vtedy, keď

$$(\forall \phi \in T)\ m_w(\phi) = t.$$

■

Reláciu ρ_1 možno využívať pri výpočtoch, pozri [Sef 99]. Tu sa tejto otázke venovať nebudeme.

Nech T je AELKB-teória a w je množina modelov. Množinu všetkých modelov teórie T označíme ako $Mod(T)$. Množinu všetkých formúl pravdivých v každom modeli z možného sveta w označíme ako $Th(w)$. Zrejme platí, že $Th(w) = Cn_A(Th(w))$, $T = Th(Mod(T))$ a $w = Mod(Th(w))$.

Teraz vyslovíme tvrdenie, ktoré poukáže na dôležitosť množín formúl

$$\{\phi : m_{w'}(\phi) = t\}.$$

Tvrdenie vlastne konštatuje adekvátnosť zobrazenia m: takéto množiny sa za istých predpokladov zhodujú so statickými autoepistemickými expanziami.

Tvrdenie 7.24 *Nech T je AELKB-teória, $\mathcal{K} = (W, \{\rho_1, \rho_2\}, m)$ je AELKB-štruktúra a $w_T \in W$ je množina všetkých modelov T. Nech $w_\perp$ je prázdna množina interpretácií.*

Pre každý možný svet $w' \in W$ taký, že $w_\perp \subseteq w' \subseteq w_T$ platí, že množina formúl $T^* = \{\phi : m_{w'}(\phi) = t\}$ je stabilná autoepistemická expanzia teórie T.

Náčrt dôkazu : Najprv dokážeme, že $T^* = \{\phi : m_{w'}(\phi) = t\} \subseteq Cn_A(T \cup \{K\phi : T^* \models \phi\} \cup \{\neg K\phi : T^* \not\models \phi\} \cup \{B\phi : T^* \models_{min} \phi\})$.

Nech $m_{w'}(\phi) = t$. Ak ϕ je tvaru $K\psi$, potom $m_{w'}(\psi) = t$. To znamená, že každý model množiny formúl T^* je modelom formuly ψ. Teda $\phi \in \{K\tau : T^* \models \tau\}$. Podobne pre $\phi = \neg K\psi$ a pre $\phi = B\psi$. To, že T^* je uzavretá vzhľadom na Cn_A, je zrejmé. Konečne, každá podmnožina w_T spĺňa T, čiže aj w' spĺňa T. Teda, $T \subseteq T^*$. Očividne, pre každú objektívnu formulu ϕ takú, že $m_{w'}(\phi) = t$ platí, že $\phi \in Cn_A(T \cup \{K\phi : T^* \models \phi\} \cup \{\neg K\phi : T^* \not\models \phi\} \cup \{B\phi : T^* \models_{min} \phi\})$. To znamená, že T^* je podmnožinou nejakej SAE.

Naopak, predpokladajme $\phi \in Cn_A(T \cup \{K\phi : T^* \models \phi\} \cup \{\neg K\phi : T^* \not\models \phi\} \cup \{B\ \phi : T^* \models_{min} \phi\})$. Priamočiaro možno ukázať, že $m_{w'}(\phi) = t$. ■

Obsah tohto tvrdenia nie je hlboký. Pripravme sa na dôležitejší postreh: Nech w_T je množina všetkých modelov AELKB-teórie T. Všimneme si dve množiny formúl: $S = \{\phi : m_{w_T}(\phi) = t\}$ a $Cn_A(T)$. Nasledujúci príklad ukáže, že $S \setminus Cn_A(T) \neq \emptyset$ pre nejaké AELKB-teórie.

Príklad 7.25 ([Prz 97]) Nech T je $\{B\neg b \wedge B\neg f \rightarrow r, \neg Kb \wedge \neg Kf \rightarrow d\}$. Niektorými prvkami z w_T sú

$$\nu_1 = \{B\neg b, B\neg f, \neg Kb, \neg Kf, r, d, b, f\},$$
$$\nu_2 = \{\neg B\neg b, \neg B\neg f, Kb, Kf, \neg r, \neg d, \neg b, \neg f\}.$$

Teda, $m_{w_T}(\neg Kb) = t$ a $m_{w_T}(\neg Kf) = t$, ale $\neg Kb \notin Cn_A(T)$ a $\neg Kf \notin Cn_A(T)$.

Nech w'_T je množina všetkých minimálnych modelov T. Ak $\nu \in w'_T$, potom určite $\neg b \in \nu$, $\neg f \in \nu$. Teda, $m_{w_T}(B\neg b) = t = m_{w_T}(B\neg f)$, ale $B\neg b, B\neg f \notin Cn_A(T)$. ■

Neprázdne $S \setminus Cn_A(T)$ môže obsahovať literály dvoch tvarov: $\neg K\phi$ alebo $B\phi$. Intuitívne, funkcia m_w generuje dva druhy hypotéz (viet, čo nepatria medzi Cn_A-dôsledky teórie T): vety o viere a vety o nepoznaní.

Vieme (na základe tvrdenia 7.24), že množiny formúl pravdivých pri ohodnotení m_{w_T} určujú nejakú SAE teórie T. Náš príklad však ukázal, že nie každá z týchto formúl je pravdivá v každom modeli z možného sveta w_T. Naším cieľom teraz bude nájsť taký možný svet w, ktorý determinuje SAE (množina všetkých viet pravdivých vo w sa bude presne zhodovať s nejakou SAE; inými slovami: w je množinou všetkých modelov nejakej SAE danej AELKB-teórie T).

Naším nasledujúcim cieľom je prezentovať konštruktívnejšiu metódu charakterizácie SAE. Vyriešime tak zatiaľ otvorenú kľúčovú otázku – ktorý možný svet determinuje SAE.

Tvrdenie 7.26 *Nech $W = \mathcal{P}(Int)$. Potom zobrazenie $\Phi : W \longrightarrow W$ definované ako $\Phi(w) = Mod(\{\phi : m_w(\phi) = t\})$ je monotónne.*

Dôkaz: Ak $w \subseteq w'$, potom $\{\phi : m_w(\phi) = t\} \supseteq \{\phi : m_{w'}(\phi) = t\}$, teda $Mod(\{\phi : m_w(\phi) = t\}) \subseteq Mod(\{\phi : m_{w'}(\phi) = t\})$. ■

Z monotónnosti Φ vyplýva, že má najmenší a najväčší pevný bod, pozri napríklad [Llo 87]. Už môžeme charakterizovať SAE.

Tvrdenie 7.27 *Nech T je AELKB-teória, $\mathcal{K} = (W, \{\rho_1, \rho_2\}, m)$ je AELKB-štruktúra a $w_T \in W$ je množina všetkých modelov T.*

Potom pre každý možný svet $w \in W$, kde $w_\perp \subseteq w \subseteq w_T$, platí: ak $\Phi(w) = w$, potom $Cn_A(Th(w))$ je statická autoepistemická expanzia teórie T (budeme hovoriť, že w determinuje túto SAE).

Definovaný operátor poskytuje základ pre konštrukciu (výpočet) statickej autoepistemickej expanzie. Naivný výpočet by prebiehal cez postupné prehľadávanie celého priestoru, ohraničeného zhora svetom w_T a testovaním podmienky $\Phi(w) = w$.

Základná myšlienka jemnejšej metódy spočíva vo výbere niektorých[17] formúl z množiny $S \setminus Cn_A(T)$, v ich vložení do T a následnom verifikovaní, či toto vloženie vedie k SAE teórie T.

Príklad 7.28 Vráťme sa k príkladu 7.25:

$$\neg Kb, \neg Kf, B\neg b, B\neg f \in S \setminus Cn_A(T).$$

[17] Niekedy výber celej množiny $S \setminus Cn_A(T)$ vedie k nekonzistentnej SAE.

Ak $T' = T \cup \{\neg Kb, \neg Kf, B\neg b, B\neg f\}$ a $w_{T'}$ je množina všetkých modelov teórie T', potom $w_{T'}$ je pevný bod operátora Φ, čiže $Cn_A(Th(w_{T'}))$ je SAE teórie T. ■

V tomto príklade sme našli SAE na prvý pokus. Samozrejme, vo všeobecnosti treba postupne pridávať ďalšie vloženia. Uvedený výpočet, spočívajúci v postupnom vkladaní nejakých formúl do T a v následnom testovaní, či množina všetkých modelov rozšírenej teórie je pevným bodom operátora Φ nás motivuje, aby sme väčšiu pozornosť venovali vloženiam do AELKB-teórí. Navyše, sémantická charakterizácia týchto vložení je zaujímavá sama o sebe: práve vloženia vytvárajú situácie, v ktorých sa prejavuje nemonotónnosť AELKB-teórií (a všetkých formalizmov na reprezentáciu poznatkov).

7.3.2 Dynamické AELKB-štruktúry

V tejto časti podáme charakterizáciu vložení do AELKB-teórií pomocou dynamických Kripkeho štruktúr.

Pokračujme v námetoch z príkladu 7.25:

Príklad 7.29 Nech T je opäť $\{B\neg b \wedge B\neg f \to r, \neg Kb \wedge \neg Kf \to d\}$.

Vložme do T formulu $b \vee f$. Potom $T' = T \cup \{b \vee f\}$. Ak w_T je množina všetkých modelov T a $w_{T'}$ všetkých modelov T', potom $w_{T'}$ neobsahuje modely z w_T, ktoré obsahujú aj $\neg b$ aj $\neg f$. To znamená, že množina minimálnych modelov T' a zodpovedajúce B-dôsledky (množina formúl tvaru $B\phi$, ktoré sú dôsledkami T') sa tiež menia.

Túto zmenu možno špecifikovať pomocou nejakej transformácie. Ak w je možný svet, potom $F_{b\vee f}(w)$ je možný svet $w' = \{m \in w : m(b \vee f) = t\}$, t.j. $F_{b\vee f}(w)$ je svet, pozostávajúci zo všetkých modelov z w, ktoré spĺňajú $b \vee f$ (ak taký model neexistuje, potom $F_{b\vee f}(w) = w_\perp$). ■

Videli sme, ako možno ku každej formule, ktorú vkladáme do AELKB-teórie, priradiť transformáciu, zobrazujúcu jeden možný svet (množinu všetkých modelov pôvodnej teórie) na iný možný svet (množinu všetkých modelov rozšírenej teórie). Túto myšlienku si teraz sformalizujeme: zadefinujeme vhodnú algebraickú štruktúru, *monoid.* Prvky tohto monoidu budú špecifikovať potrebné transformácie možných svetov.

Základné intuície: U, množinu všetkých vložení do AELKB-teórie T, budeme reprezentovať množinou formúl (vloženie formuly u reprezentujeme touto formulou, nepatrne to však budeme musieť skomplikovať). Keď po vložení formuly u nasleduje vloženie formuly v, môžeme konkatenáciu týchto dvoch vložení reprezentovať vložením formuly $u \wedge v$. Môžeme si predstaviť aj vloženie prázdnej formuly, ktoré nič nemení. Týmto vlastnostiam presne zodpovedá pojem monoidu.

Definícia 7.30 *Monoid* je trojica $(M, \circ, e)$, kde M je množina, $\circ : M \times M \longrightarrow M$ je asociatívna operácia, $e \in M$ a pre každé $x \in M$ platí $e \circ x = x = x \circ e$ (e nazývame jednotkou monoidu). ■

Teraz potrebujeme zadefinovať taký monoid, aby sme dokázali ku každému jeho prvku priradiť nejaké transformácie na možných svetoch. Nech w je možný svet a $f_u(w) = w'$ pre nejaké u. Aby sme zobrazenie f_u definovali rozumne, ostáva nám jediný problém – potrebujeme dosiahnuť, aby platilo: ak $u \equiv v$, potom $f_u(w) = f_v(w)$ pre každé $w \in W$.

Definícia 7.31 Nech u je J_A-formula a $[u]_\equiv = \{x \in J_A : x \equiv u\}$.

Ak σ je funkcia, ktorá priradí ku každej triede $[u]_\equiv$ presne jednu formulu ϕ z tejto triedy, ϕ budeme nazývať *reprezentantom* danej triedy. ■

Definícia 7.32 (i-monoid) Nech $U = \{\phi : \exists [u]_\equiv \ \sigma([u]_\equiv) = \phi\}$ je množina všetkých reprezentantov. Definujeme *i-monoid* $\mathcal{U}$ takto: pre $u, v \in U$ bude $u \circ v = \sigma([u \wedge v]_\equiv)$. ■

Očividne, operácia $\circ$ je asociatívna a prázdna formula ϵ plní úlohu jednotky monoidu, $u \circ \epsilon = u = \epsilon \circ u$ pre každé $u \in U$. Na základe konvencie môžeme ϵ považovať za reprezentanta triedy všetkých tautológií.

Definícia 7.33 (Dynamická AELKB-štruktúra) Dvojicu $(\mathcal{U}, \mathcal{K})$, kde

$$\mathcal{K} = (W, \{\rho_1, \rho_2\}, m)$$

je AELKB-štruktúra a $\mathcal{U}$ je i-monoid, budeme volať *dynamická AELKB-štruktúra.*

Akciu monoidu $\mathcal{U}$ na W definujeme nasledovne: pre $u \in U$ je $f_u(w) = w' = Mod(Cn_A(Th(w) \cup \{u\}))$. ■

Samozrejme, w' je (jedinou) hodnotou $f_u(w)$:

Fakt 7.34 *Nech T je AELKB-teória, $\mathcal{K} = (W, \{\rho_1, \rho_2\}, m)$ je AELKB-štruktúra. Nech $w \in W$ je množina všetkých modelov T a $T' = T \cup \{u\}$.*

Potom vo W existuje presne jeden možný svet w' taký, že

$$\begin{aligned} w' &= Mod(Cn_A(T')), \\ f_u(w) &= w'. \end{aligned}$$

Fakt 7.35 *Nech je daná dynamická AELKB-štruktúra. Platí:*

- $f_\epsilon(w) = w$,
- $f_{u \circ v}(w) = f_u(f_v(w))$,
- *ak* $f_{u \circ v}(w) = w'$, *potom* $Th(w') = Cn_A(Th(w) \cup \{u \wedge v\}) = Cn_A(Th(w) \cup \{u\} \cup \{v\})$.

Nakoniec načrtneme procedúru výpočtu SAE založenú na vloženiach. Nech T je AELKB-teória a $\mathcal{K}$ je zodpovedajúca dynamická AELKB-štruktúra. Nech $w_T \in W$ je množina všetkých modelov T:

- vyber hypotézu h z $S \setminus Cn_A(T)$,
- vypočítaj $f_h(w_T) = w'$,
- ak w' je pevným bodom Φ, potom vráť vypočítanú SAE (a hľadaj ďalšie SAE), v opačnom prípade vyber hypotézu h' z $S' \setminus Cn_A(T')$, kde $S' = \{\phi : m_{w'}(\phi) = t\}$ a $T' = Cn_A(T \cup \{h\})$,[18] a pokračuj vo výpočte (rekurzívnom).

Poznámka 7.36 Predpokladáme backtracking – niekedy sa môže ukázať užitočným revidovať niektoré predchádzajúce výbery. Predčasný výber formúl tvaru $\neg K\phi$ môže napríklad viesť k priamej konštrukcii nekonzistentej SAE. ■

V nasledujúcej časti ukážeme, ako možno zosilniť konštrukciu dynamickej Kripkeho štruktúry. Doposiaľ sme si všímali iba transformácie možných svetov. Teraz si načrtneme, ako možno rekonštruovať reláciu dostupnosti.

7.3.3 Revízie AELB-teórií

DKS možno použiť aj na špecifikáciu a výpočet revízií. Ďalej predpokladáme iba AELB-teórie (nebudeme používať operátor K).

Príklad 7.37 ([APP 96]) Nech $T = \{B\neg b \to r\}$. Množina všetkých modelov T je $w =$

$$\{\{B\neg b, r, b\}, \{B\neg b, r, \neg b\}, \{\neg B\neg b, r, b\},$$
$$\{\neg B\neg b, r, \neg b\}, \{\neg B\neg b, \neg r, b\}, \{\neg B\neg b, \neg r, \neg b\}\}.$$

Nech $u = \{\neg r\}$. Platí, že $f_u(w) = w' = \{\{\neg B\neg b, \neg r, b\}, \{\neg B\neg b, \neg r, \neg b\}\}$ je množina všetkých modelov teórie $T' = T \cup \{\neg r\}$. Množina minimálnych modelov je $w_{min} = \{\{\neg B\neg b, \neg r, \neg b\}\}$. Teda, T'^*, jediná SAE teórie T' je nekonzistentná: $T' \models_{min} \neg b$, preto $B\neg b \in T'^*$, a teda $T'^* \models \neg r \wedge r$. ■

Videli sme, že v konexte neúplnej informácie má sémantika minimálnych modelov nežiadúce dôsledky. V našom príklade hypotéza $B\neg b$ spôsobila nekonzistentnosť teórie T'. Táto hypotéza je prvkom SAE (definovanej štandardným spôsobom). Zdá sa, že potrebujeme modifikovanú ideu rozumnej hypotézy. Náš návrh spočíva v rekonštrukcii danej dynamickej AELKB-štruktúry tak, aby SAE tejto rekonštruovanej štruktúry spĺňala naše intuitívne požiadavky.

[18] Ak $S' \setminus Cn_A(T') = \emptyset$, potom w' je pevným bodom Φ.

Príklad 7.38 Všimnime si príčinu nekonzistennosti pozorovanej v príklade 7.37. Minimálny model $\{\neg B\neg b, \neg r, \neg b\}$ je v určitom zmysle patologický. Obsahuje dvojicu $(\neg B\neg b, \neg b)$ – nazvime ju gangom (podľa [KLo 89]) – s potenciálnym konfliktom mezi $\neg b$ a $\neg B\neg b$ (na jednej strane tvrdíme $\neg b$, na druhej strane tomu neveríme). Túto patológiu možno napraviť pomocou techniky z [Buc 97]. Jej podstatou je modifikácia relácie dostupnosti ρ_2. My ju aplikujeme tak, že odstránime dvojicu (w', w_{min}) z ρ_2 a do ρ_2 vložíme nejakú opravu tejto dvojice. Cieľom opravy je eliminovať nežiadúce dôsledky, ktoré vďaka minimálnemu vyplývaniu dostávame z AELB-teórie.

Hlavnou myšlienkou opravy je nahradiť gang nejakým racionálnejším výberom. Napríklad, môže to byť

$$w_{rat} = \{\neg B\neg b, \neg r, b\}$$

(interpretácia $\{B\neg b, \neg r, \neg b\}$ nie je modelom T').

Dvojicu (w', w_{rat}) vložíme do ρ_2. Po tejto oprave (rekonštrukcii)

$$\rho_2 := (\rho_2 \setminus (w', w_{min})) \cup (w', w_{rat})$$

platí, že $T' \models_{min} b$, z toho $Bb \in T^{'*}$, napokon teda: $T^{'*} \not\models \neg r \wedge r$. ■

Tu sme iba načrtli základnú myšlienku, technické detaily, založené na pojme racionálnej interpretácie, možno nájsť v [Sef 99].

Zosumarizujme to: Nezmenili sme pojem SAE, ale sémantickú štruktúru, zodpovedajúca AELB-teórii. Modifikovaná AELKB-štruktúra mení vzťah minimálneho vyplývania. Preto sa mení množina odvoditeľných hypotéz tvaru $B\phi$. Vyplývanie, špecifikované takouto sémantikou, môžeme volať *dynamickým preferenčným vyplývaním.* Ak niektoré fakty bázy poznatkov protirečia odvoditeľným presvedčeniam, potom dynamicky modifikujeme danú sémantickú špecifikáciu vyplývania, založenú na preferovaní minimálnych modelov.

Uvedený prístup k revíziám AELB-teórií motivuje zovšeobecnenie DKS. DKS možno obohatiť o zobrazenia z relácií dostupnosti na iné relácie dostupnosti. Pojem dynamickej Kripkeho štruktúry možno ďalej modifikovať a zosilňovať. V nasledujúcej časti si ukážeme, ako možno účelne definovať operácie na Kripkeho štruktúrach.

7.3.4 Dynamické logické programovanie

Evolúcia poznania je jedným z kľúčových problémov z hľadiska nemonotónneho usudzovania. Už sme uviedli dôvody, prečo treba považovať nemonotónnosť iba za symptóm vývoja poznania (pribúdania nových poznatkov, generovania hypotéz, vynucovania revízií, falzifikácie starších presvedčení).

Formalizáciu niektorých čŕt vývoja poznania navrhli [APP 98] a nazvali ju dynamickým logickým programovaním. Vo svojej práci nadviazali na predchodcov, ktorými boli hlavne [MaT 94, Prz 97b, APe 96].

Základná myšlienka dynamického logického programovania je táto: Je daný program P. Ten predstavuje počiatočnú bázu poznatkov. Potom je modifikovaný pomocou iného programu U. Ten predstavuje bázu poznatkov, špecifikujúcu modifikácie pôvodnej bázy poznatkov. Výsledkom takejto modifikácie je nový program $P \oplus U$. Túto situáciu možno zovšeobecniť na postupnosť programov $P \oplus U_1 \oplus \cdots \oplus U_n$ (alebo inú komplikovanejšiu štruktúru). Modifikujúci program môže predstavovať poznatky o zmenách v prostredí, ktoré nastali časom, môže znamenať revízie presvedčení alebo aj zmenu hľadiska.

Cieľom dynamického logického programovania je reprezentácia dynamicky sa meniaceho poznania, dynamických báz poznatkov. Programy, ktoré sa používajú v dynamickom logickom programovaní, sa nazývajú zovšeobecnenými logickými programami a môžu mať v hlavách aj negáciu (ako konečné zlyhanie). V [APP 98] sa používa termín defaultová negácia. Tento termín sa opiera o výsledky z [BiF 87]. Sémantiky negácie ako konečného zlyhania za určitých podmienok možno charakterizovať pomocou defaultov. Intuícia, ktorá nám pomôže: $\neg A$ prijímame ako default. To znamená, prijímame ju dovtedy, kým nezistíme, že platí A. Keď platí A, vlastne sme vyvrátili stráž normálneho defaultového pravidla $: \neg A / \neg A$.

Pravidlá modifikujúceho programu špecifikujú nejaké vloženia do (a odstránenia z) pôvodného programu. Defaultová negácia v hlave pravidla slúži na špecifkáciu odstránení.

V [APP 98] sa analyzovala možnosť špecifikovať $P \oplus U$ vychádzajúc z modifikácií interpretácií. Prednosťou takéhoto prístupu je, že dovoľuje modifikáciu špecifikovať sémanticky. Dovoľuje tak abstrahovať od náhodilostí, závislých na nepodstatných syntaktických rozdieloch. Pod *modifikáciou interpretácií* rozumieme to, že modifikovaný program (teóriu) zamýšľame špecifikovať na základe modifikovanej sémantickej štruktúry. Môžeme to symbolicky vyjadriť ako

$$Mod(P') = Update_U(Mod(P)).$$

$Mod(X)$ je množina všetkých relevantných (napríklad stabilných) modelov programu X. $Update_U(M)$ je modifikácia triedy modelov M, determinovaná programom U. Preto, P' je modifikáciou P pomocou U práve vtedy, keď množina všetkých stabilných modelov P' sa zhoduje s množinou modifikovaných stabilných modelov programu P.

Podľa [APP 98] modifikovaný program $P \oplus U$ nezávisí iba na sémantike programov P a U, ale aj na ich syntaxi. Dokladá to príklad (podobnosť na príklad 7.3 je očividná):

Príklad 7.39 ([APP 98]) Predstavme si program P:

$$nevinný \leftarrow \neg\, preukázaná\_vina.$$

Nech význam P reprezentuje množina všetkých jeho stabilných modelov

$$Mod(P) = \{\{nevinný\}\}.$$

Jediným stabilným modelom P je $S = \{nevinný\}$.

Nech P modifikujeme pomocou

$$U = \{preukázaná\_vina \leftarrow\}.$$

Potom množina $S' = S \cup \{preukázaná\_vina\}$ by mala byť stabilným modelom modifikovaného programu, ak by sme za sémantickú špecifikáciu modifikácií považovali modifikácie interpretácií.

Samozrejme, S' nepredstavuje zamýšľaný význam modifikovaného programu. ■

Prístup z [APP 98] opíšeme stručne a neformálne. Po prvé, rozšíri sa množina výrokových premenných. Ku každej výrokovej premennej z P i U sa zavedú nové výrokové premenné: $\hat{J} = \{A^-, A_P, A_P^-, A_U, A_U^- : A \in J\} \cup J$. Nové premenné umožnia zaznamenávanie vplyvu P a U na modifikovaný program $P \oplus U$.

Po druhé, ku každej klauze z P a U sa zavedú nejaké klauzy v modifikovanom slovníku, okrem toho sa zavádzajú nové klauzy pre každú výrokovú premennú. Zovšeobecnený program $P \oplus U$ je výsledkom syntaktickej transformácie. Pozostáva z nasledujúcich tried kláuz v rozšírenom jazyku $\hat{J}$:

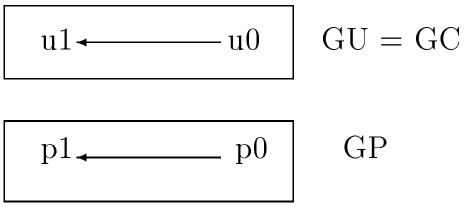
- Pre každú klauzu $L \leftarrow L_1, \ldots, L_n$ z P (alebo U) sa zavádza do $P \oplus U$ nová klauza s hlavou L_P (alebo L_U) a negatívne literály L_i v tele pôvodnej klauzy sa nahradia novými symbolmi L_i^-.

- Pre každý objektívny atóm $A \in J$ sa zavádza:

  - dve nové klauzy $A \leftarrow A_U$ a $A^- \leftarrow A_U^-$ (vplyv programu U na program $P \oplus U$ je rozhodujúci),
  - dve nové klauzy $A \leftarrow A_P, \neg A_U^-$ a $A^- \leftarrow A_P^-, \neg A_U$ (vplyv programu P na program $P \oplus U$ je možný vtedy, keď sa nedostávame do konfliktu s U),
  - dve nové klauzy $A^- \leftarrow \neg A_P, \neg A_U$ a $\neg A \leftarrow A^-$ (pravidlá, zavádzajúce defaultové predpoklady).

Takouto transformáciou sa zachytáva dominantný vplyv programu U na program $P \oplus U$, definuje sa, čo z P ostáva v platnosti a pravidlá pre defaulty umožnia prijať negácie tam, kde nemáme dôvod veriť v opak.

Naším cieľom je: Skonštruovať (čisto) sémantickú štruktúru, ktorá umožní špecifikovať modifikovaný program (bez používania syntaktických transformácií, bez používania nových výrokových premenných, bez zavádzania nových kláuz). Zadefinujeme Kripkeho štruktúru, asociovanú so zovšeobecneným logickým programom a nejakú operáciu na Kripkeho štruktúrach. Táto operácia nám bude slúžiť ako sémantická špecifikácia modifikovaného programu. Podrobnosti v [Sef 00].



Obrázok 7.1: Vrchol $p0$ reprezentuje interpretáciu $\{\neg\ preuk\acute{a}zan\acute{a}\_vina\}$, $p1 = \{\neg\ preuk\acute{a}zan\acute{a}\_vina, innocent\}$, $u0 = \emptyset$, $u1 = \{preuk\acute{a}zan\acute{a}\_vina\}$. Hrany $(p0, p1)$ a $(u0, u1)$ reprezentujú závislosti medzi literálmi (druhý člen dvojice je zdôvodnený prvým členom).

Príklad 7.40 Modifikácia interpretácií zlyháva na dynamických logických programoch preto, lebo nie je schopná zaznamenať závislosti medzi literálmi. V istom zmysle slova je literál *nevinný* na základe programu P závislý na literáli $\neg preuk\acute{a}zan\acute{a}\_vina$ (je zdôvodnený týmto literálom).

Toto zdôvodnenie je však narušené (anulované) modifikujúcim programom U (faktom $preuk\acute{a}zan\acute{a}\_vina \leftarrow$). Aby sme mohli vytvoriť adekvátnu sémantickú charakterizáciu modifikácií zovšeobecnených logických programov, musíme v nej rešpektovať závislosti medzi literálmi.

Neprekvapuje, že navrhovaná sémantická charakterizácia využije Kripkeho štruktúry: zdôvodnenia sú reprezentované pomocou relácie dostupnosti medzi interpretáciami.

Na obrázku 7.1 je ilustrovaná možnosť reprezentovať význam logického programu Kripkeho štruktúrou. Grafy GP a GU znázorňujú relevantné časti Kripkeho štruktúr asociovaných s programami P a U. Vrcholy predstavujú interpretácie (množiny literálov). Pokiaľ ide o reláciu dostupnosti: 3-interpretácia M je dostupná z 3-interpretácie M', ak telo nejakého pravidla daného programu je splnené v M' a telo spolu s hlavou je splnené v M.

Sémantická charakterizácia modifikovaného programu (graf GC) sa konštruuje nad grafom GU (zachovávame dominantný vplyv programu U). Niektoré časti grafu GP možno – vo všeobecnosti – pripojiť ku GU, v tomto príklade to však nie je možné. ■

Podrobnejšie: Predpokladajme konečnú množinu výrokovologických premenných J. Množinu $\{\neg\ A : A \in J\}$ budeme označovať ako $\mathcal{D}$ (defaulty, predpoklady). Množinu J_{not} definujeme ako $J \cup \mathcal{D}$. Prvky z J_{not} nazývame *literálmi*.

Definícia 7.41 *Zovšeobecnená klauza* je formula c tvaru $L \leftarrow L_1, \ldots, L_k$, kde L, L_i sú literály. L budeme označovať aj ako $hlava(c)$ a konjunkciu $L_1, \ldots, L_k$ ako $telo(c)$. Množinu zovšeobecnených kláuz voláme *zovšeobecnený logický program.* ■

Budeme používať skratky: „klauza“ miesto „zovšeobecnená klauza“ a „program“ miesto „zovšebecnený logický program“.

Definícia 7.42 *Konfliktnými literálmi* budeme nazývať A a $\neg\ A$ pre každé $A \in J$. Množina literálov je *konzistentná*, ak neobsahuje dvojicu konfliktných literálov. ■

Budeme si všímať 3-interpretácie (podľa definície 3.7 z kapitoly 3), konzistentné podmnožiny J_{not}. Ak sa v 3-interpretácii $\mathcal{I}$ vyskytuje atóm A, považujeme ho za pravdivý v $\mathcal{I}$ (a $\neg A$ za nepravdivý), ak sa v nej vyskytuje negatívny literál $\neg B$, považujeme ho za pravdivý v $\mathcal{I}$ a B za nepravdivý. 2-interpetáciu budeme volať aj *totálnou interpretáciou*. Je to taká 3-interpretácia $\mathcal{I}$, že pre každé $A \in J$ buď $A \in \mathcal{I}$ alebo $\neg\ A \in \mathcal{I}$.

Zaujímajú nás iba výrokové premenné, ktoré sa vyskytujú v programoch. J^P bude označovať množinu všetkých výrokových premenných v programe P. 3-interpretácia *programu* P je konzistentná podmnožina množiny J^P_{not}. Množinu všetkých 3-interpretácií programu P označíme ako Int_P. Nekonzistentnú množinu literálov (nemožnú interpretáciu) označíme ako $w_\perp$.

Splnenie a model literálu, klauzy a programu v 3-interpretácii definujeme ako obvykle.

Výrokovologické zovšeobecnené programy môžeme chápať ako hornovské programy: každý literál $\neg\ A$ môžeme považovať za novú výrokovú premennú (ak $\neg\ A \in J$, treba ho premenovať). Najmenší model hornovského programu H označíme ako $Least(H)$.

Prv než zadefinujeme pojem Kripkeho štruktúry asociovanej s programom P, ešte jeden príklad, komplikovanejší ako bol príklad 7.40:

Príklad 7.43 ([Nie 98]) Nech P je

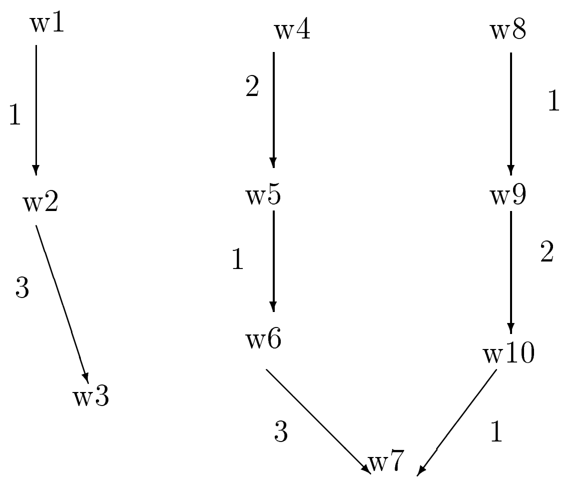
$$\begin{aligned} \{p &\leftarrow \neg\ q, r, \\ q &\leftarrow \neg\ p, \\ r &\leftarrow \neg\ s, \\ s &\leftarrow \neg\ p\}. \end{aligned}$$

Kripkeho štruktúru $\mathcal{K}_P$ môžeme v tejto chvíli analyzovať iba intuitívne. Predstavme si $\mathcal{K}_P$ ako graf. Jeho fragment je na obrázku 7.2.

Vrcholy sú 3-interpretácie. Budeme rozlišovať tri druhy hrán – ρ_1, ρ_2, ρ_3.

Hrana $(w1, w2)$ je príkladom ρ_1-hrany, pričom $w1 = \{\neg\ p\}$ a $w_2 = \{\neg\ p, q, s\}$. V P sú dve klauzy s telom splneným vo $w1$. Dôsledky týchto kláuz pripojíme k $w1$. Možný svet $w2$ je výsledkom tejto operácie. Okrem toho pripustíme aj $(\{\neg p\}, \{\neg p, q\}) \in \rho_1$ atď.

Relácia ρ_2 umožňuje dopĺňať 3-interpretácie negatívnymi defaultovými predpokladmi, ak sa takýto negatívny literál neskôr použije v „odvodení“ pomocou nejakej ρ_1-hrany. V našom príklade, $(w4, w5) \in \rho_2$, kde $w4 = \{r\}$,



Obrázok 7.2: Fragment $\mathcal{K}_P$. Hrana označená ako i je ρ_i-hrana. Možný svet $w1$ je $\{\neg\, p\}$, $w2 = \{\neg\, p, q, s\}$, $w3 = \{\neg\, p, q, s, \neg\, r\}$, $w4 = \{r\}$, $w5 = \{r, \neg\, q\}$, $w6 = \{r, \neg\, q, p\}$, $w7 = \{r, \neg\, q, p, \neg\, s\}$, $w8 = \{\neg\, s\}$, $w9 = \{\neg\, s, r\}$, $w10 = \{\neg\, s, r, \neg\, q\}$.

$w_5 = \{r, \neg\, q\}$. Splnené sú nasledujúce podmienky: $w5$ je model tela klauzy $p \leftarrow \neg\, q, r$, nemáme žiadnu evidenciu proti $\neg\, q$ ($w4$ neobsahuje q) a $(w5, w6) \in \rho_1$, t.j. $w5$ sa používa ako zdôvodnenie p a $w6 = \{r, \neg\, q, p\}$.

Napokon motivujeme ρ_3. Neexistuje totálna interpretácia u, pre ktorú platí $(w_6, u) \in \rho_1$. To znamená, žiadna klauza nie je aplikovateľná na $w_6 = \{r, \neg\, q, p\}$ (okrem $p \leftarrow \neg\, q, r$, tá však nezmení možný svet $w6$). To znamená, že P neumožňuje potvrdiť pravdivosť s (ak predpokadáme $w6$). Preto sa rozhodujeme prijať defaultový predpoklad, že s nie je pravdivé (vzhľadom na P). Hrana z $\{r, \neg\, q, p\}$ do $\{r, \neg\, q, p, \neg\, s\}$ reprezentuje zúplnenie $\{r, \neg\, q, p\}$ pomocou $\neg\, s$.

Okomentovať treba, prečo nedovoľujeme zúplňovanie kedykoľvek (tým by sa umožnilo splynutie hrán typu ρ_2 a ρ_3). Jediným dôvodom je, aby graf rástol čo najmenej. Spomalenie tohto rastu potrebujeme kvôli potenciálnej kompilácii (v štýle [CDS 95]) výpočtu stabilných modelov, ktoré nám práve ilustrované Kripkeho štruktúry dovolia. Ešte sa k tomu vrátime. ∎

Zhrňme to: Hrany ρ_1 zodpovedajú aplikáciám kláuz na 3-interpretácie. Klauza c je aplikovateľná na 3-interpretáciu w, ak $w \models \mathit{telo}(c)$. Vo všeobecnosti, pre každé $c \in P$: ak w je modelom $\mathit{telo}(c)$, potom $\mathit{hlava}(c) \in w'$ pre w' také, že $w \subseteq w'$ a $(w, w') \in \rho_1$. Intuitívne, (w, w') reprezentuje krok vo výpočte zdola nahor.

Ak $(w, w') \in \rho_2$, potom $w' = w \cup u$ pre nejaké u také, že $\emptyset \neq u \subseteq \mathcal{D}$ a existuje $(w', w'') \in \rho_1$. To znamená, ρ_2 reprezentuje predpokladanie defaultových negácií, používaných pri výpočte podľa ρ_1.

Napokon, ak atóm A nie je vypočítaný zdola nahor, predpokladáme, že platí $\neg A$. Relácia ρ_3 reprezentuje zúplnenie tých 3-interpretácií, ktoré nemožno zmeniť nejakými klauzami z P.

Sme pripravení definovať Kripkeho štruktúru $\mathcal{K}_P$ asociovanú s P.

Definícia 7.44 Nech P je program. Kripkeho štruktúra $\mathcal{K}_P$ asociovaná s P je dvojica (W, ρ), kde:

- $W = Int_P \cup \{w_\perp\}$, W je množina možných svetov, Int_P je množina všetkých 3-interpretácií programu P,
- ρ je binárna relácia na $W \times W$ (relácia dostupnosti) a pozostáva z troch relácií: $\rho = \rho_1 \cup \rho_2 \cup \rho_3$,
  1. relácia ρ_1 obsahuje množinu všetkých dvojíc (w, w') takých, že $w' = w \cup \{\mathit{hlava}(c_i) : i = 1, \ldots, k\}$, kde $c_1, \ldots, c_k \in P$, pričom $w \models \mathit{telo}(c_i)$,
  2. ak $(w_2, w') \in \rho_1$ a $\emptyset \neq w_2 \setminus w_1 \subseteq \mathcal{D}$ pre $w_1 \in W$ také, že $w_1 \subseteq w_2$, potom $(w_1, w_2) \in \rho_2$,
  3. ak w nie je totálna interpretácia a pre žiadne $u \neq w$ neexistuje hrana $(w, u) \in \rho_1 \cup \rho_2$, potom $(w, w') \in \rho_3$, kde $w' = w \cup \{\neg A : A \notin w\}$.

■

Pri konštrukcii hrany ρ_1 nepredpokladáme, že nutne pre všetky $c \in P$ také, že $w \models \mathit{telo}(c)$, platí $\mathit{hlava}(c) \in w'$, ak $(w, w') \in \rho_1$.

Samozrejme, na Kripkeho štruktúru $\mathcal{K}_P$ možno nazerať ako na graf:

Definícia 7.45 *ρ-cesta* je postupnosť σ hrán

$$(w_0, w_1), (w_1, w_2), \ldots, (w_{n-1}, w_n) \in \mathcal{K}_P$$

takých, že $(wi, w_{i+1}) \in \rho$. Niekedy budme používať skratku

$$\langle w_0, w_1, w_2, \ldots, w_{n-1}, w_n \rangle.$$

Budeme hovoriť, že σ je *zakorenená* vo w_0 (aj w_0-zakorenená).

Ak v $\mathcal{K}_P$ neexistuje hrana (w_n, w) taká, že $w \neq w_n$, budeme hovoriť, že σ *terminuje* vo w_n (tiež: w_n je *terminálny vrchol* grafu $\mathcal{K}_P$). ■

Podobne možno definovať ρ_1-cestu alebo $\rho_1 \cup \rho_2$-cestu.

Kripkeho štruktúra $\mathcal{K}_P$ asociovaná s programom P umožňuje špecifikovať a počítať stabilné modely programu P. Videli sme, že Kripkeho štruktúry sú vhodné na zaznamenávanie zdôvodnení. Ak sa zameriame na zdôvodnenia, ktoré nie sú cirkulárne (do kruhu) a ktoré vedú k totálnej intrerpretácii, dostaneme stabilné modely.

Podobne ako v TMS prijímame dva druhy základných predpokladov – fakty a defaultové negácie (podmnožiny $\mathcal{D}$, v TMS [Doy 79] sa volajú nemonotónnymi predpokladmi: ak nie je evidencia proti nim, prijímame $\neg A$ pre nejaký atóm A).

Príklad 7.46 Vráťme sa k príkladu 7.43 (a k obrázku 7.2).

P neobsahuje žiadny fakt, preto v $\mathcal{K}_P$ neexistuje žiadna $\emptyset$-zakorenená cesta. Teda ako relevantné cesty môžeme brať iba tie, čo sú zakorenené v nejakom $w \subseteq \mathcal{D}$. Na obrázku je $\{\neg s\}$-zakorenená ρ-cesta, ktorá terminuje v stabilnom modeli $\{\neg s, r, \neg q, p\}$ a jedna $\{\neg p\}$-zakorenená ρ-cesta, ktorá terminuje v ďalšom stabilnom modeli $\{p, \neg q, \neg s, r\}$. Oba modely sú všetkými stabilnými modelmi programu P. ■

Môžeme už vysloviť kritériá, umožňujúce identifikovať a počítať stabilné modely pomocou $\mathcal{K}_P$.

Definícia 7.47 Nech P je program, σ je acyklická ρ-cesta

$$\langle w_0, w_1, w_2, \ldots, w_{n-1}, w_n\rangle$$

v $\mathcal{K}_P$. Ak

- buď $w_0 = \emptyset$,
- alebo $\emptyset \neq w_0 \subseteq \mathcal{D}$,

hovoríme, že σ je *správne zakorenená*. ■

Tvrdenie 7.48 *Nech P je program, $\mathcal{K}_P$ je Kripkeho štruktúra asociovaná s P, $\sigma = (w_0, w_1), (w_1, w_2), \ldots, (w_{n-1}, w_n)$ je acyklická ρ-cesta v $\mathcal{K}_P$, ktorá terminuje v totálnej interpretácii w_n.*

Ak je σ správne zakorenená, potom w_n je stabilný model programu P.

Dôkaz:
Ak P je zovšeobecnený logický program, môžeme ho považovať za definitný logický program (každý literál $\neg A$ za novú výrokovú premennú) s integritnými obmedzeniami tvaru $\leftarrow A, \neg A$ pre každú premennú $A \in J^P$. Podľa [APP 98]: w_n je stabilným modelom P práve vtedy, keď $w_n = \mathit{Least}(P \cup w_n^-)$, kde $w_n^- = \{\neg A : \neg A \in w_n\}$.

Nech P' je $P \cup \{\neg A \leftarrow : \neg A \in w_n^-\}$. Považujme teda P' za definitný program (s integritnými obmedzeniami). Integritné obmedzenia sú zrejme splnené vo w_n a $w_n = \mathit{Least}(P') = \mathit{Least}(P \cup w_n^-)$. ■

Tvrdenie 7.49 *Nech S je stabilný model programu P a $\mathcal{K}_P$ je Kripkeho štruktúra asociovaná s P.*

Potom v $\mathcal{K}_P$ existuje správne zakorenená a acyklická ρ-cesta

$$\sigma = (w_0, w_1), (w_1, w_2), \ldots, (w_{n-1}, w_n),$$

ktorá terminuje v totálnej interpretácii $w_n = S$.

Idea dôkazu: Opäť môžeme využiť, že S je stabilným modelom P práve vtedy, keď $S = Least(P \cup S^-)$. V $\mathcal{K}_P$ určite existuje (na základe konštrukcie) cesta od faktov alebo defaultových negatívnych predpokladov k $S = w_n = Least(P \cup S^-)$. ■

Do Kripkeho štruktúr, asociovaných s programami možno kompilovať problém hľadania stabilných modelov (a problémy, späté so stabilnými modelmi). Otvorený problém je, či by sa pri vhodnej abstrakcii od nerelevantných hrán v Kripkeho štruktúre dala očakávať kompilácia do priestoru polynomicky väčšieho a zvládnuteľné počítanie problémov súvisiacich so stabilnými modelmi nad týmto priestorom.

Videli sme teda, ako možno skonštruovať Kripkeho štruktúru, asociovanú s daným programom a ako v nej možno identifikovať jeho stabilné modely. Je čas prikročiť k definovaniu operácie na Kripkeho štruktúrach.

Naším cieľom je skonštruovať Kripkeho štruktúru $\mathcal{K}_{P \oplus U}$ nad dvoma Kripkeho štruktúrami – $\mathcal{K}_P$ (pripomeňme, že špecifikuje sémantiku pôvodného programu P) a $\mathcal{K}_U$ (špecifikuje sémantiku modifikujúceho programu U). Ako sémantickú špecifikáciu modifikovaného programu zamýšľame používať štruktúru $\mathcal{K}_{P \oplus U}$.

Hlavná myšlienka konštrukcie je taká, že sa oprieme o graf $\mathcal{K}_U$, k nemu pripojíme tie časti grafu $\mathcal{K}_P$, ktoré nie sú v konflikte s $\mathcal{K}_U$ a prípadne dodáme ešte ρ_3 hrany tam, kde je možné zúplnenie. Najprv si treba uvedomiť, ku ktorým vrcholom grafu $\mathcal{K}_U$ môžeme pripájať cesty z $\mathcal{K}_P$. Tieto vrcholy budeme volať premosteniami.

Definícia 7.50 Nech $\mathcal{K}_U$ je Kripkeho štruktúra asociovaná s modifikujúcim programom U.

Premostenia grafu $\mathcal{K}_U$ sú

(i) všetky vrcholy, ktoré terminujú nejakú ρ-cestu,

(ii) všetky vrcholy w také, že existuje ρ_1-hrana $e = (u, w)$ pre nejaké u (budeme hovoriť, že w je *cieľom* e) a ρ_2- alebo ρ_3-hrana $e' = (w, v)$ pre nejaké v (w je *zdrojom* e'),

(iii) $\emptyset$ alebo $w \subseteq \mathcal{D}$, ak nie sú zdrojom nejakej ρ_1-hrany v $\mathcal{K}_U$.

■

Zaujímavý je bod (*ii*) definície: miesto defaultových negatívnych predpokladov môžeme prípadne prijať atómy, zdôvodnené programom P.

Definícia 7.51 Cestu $\sigma = \langle w_0, \ldots, w_n \rangle$ *možno pripojiť* k vrcholu w práve vtedy, keď $w_0 \subseteq w$. ■

Definícia 7.52 Nech $\sigma = \langle u_0, \ldots, u_n \rangle$ je ρ-cesta a w je premostenie grafu $\mathcal{K}_U$.

Potom *pripoj* σ k w je nasledujúca čiastočná operácia: ak σ možno pripojiť k w, potom $(w, u_1 \cup w), \ldots, (u_{n-1} \cup w, u_n \cup w)$ je ρ-cesta grafu $\mathcal{K}_{P \oplus U}$. Ak pre nejaké i platí, že $w \cup u_{i+1}$ je nekonzistentná množina, nahradíme ju vo dvojici $(w \cup u_i, w \cup u_{i+1})$ svetom $w_\perp$ a zvyšok cesty odstránime. ■

Definícia 7.53 ($\mathcal{K}_{P \oplus U}$) Nech P a U sú neprázdne programy a $\mathcal{K}_P$, $\mathcal{K}_U$ sú Kripkeho štruktúry asociované s P, U.

Konštruujme $\mathcal{K}_{P \oplus U}$, Kripkeho štruktúru špecifikujúcu modifikáciu programu P programom U nasledovne:

1. každá ρ_1-hrana a ρ_2-hrana z $\mathcal{K}_U$ je ρ_1-hrana (ρ_2-hrana) grafu $\mathcal{K}_{P \oplus U}$,
2. pre každé premostenie w z $\mathcal{K}_U$ a každú $\rho_1 \cup \rho_2$-cestu
$$\sigma = (u_0, u_1), \ldots, (u_{n-1}, u_n)$$
z $\mathcal{K}_P$: pripoj σ k w,
3. vytvor v $\mathcal{K}_{P \oplus U}$ nové ρ_3-hrany všade tam, kde to je možné.

■

Teraz načrtneme, ako možno $\mathcal{K}_{P \oplus U}$ využiť na sémantickú špecifikáciu (nejakého) modifikovaného programu $P \oplus U$. Nejakého preto, lebo pripustíme, aby vo všeobecnosti existovalo viac modifikovaných programov $P \oplus U$, ak je daný P a U.

Videli sme, že niektoré cesty v Kripkeho štruktúre, asociovanej s nejakým programom, umožňujú identifikovať stabilné modely daného programu, pozri tvrdenie 7.48. Naša úloha je teraz opačná: pre daný graf $\mathcal{K}_{P \oplus U}$ máme špecifikovať vhodný program (alebo množinu programov). Na základe analógie nás budú zaujímať správne zakorenené ρ-cesty terminujúce v totálnych interpretáciách z $\mathcal{K}_{P \oplus U}$. Budeme ich používať ako základňu pre špecifikáciu $P \oplus U$ a budeme chcieť, aby identifikovali stabilné modely programu $P \oplus U$.

Najprv zavedieme niekoľko definícií. Zjednodušíme si tak opis $\mathcal{K}_{P \oplus U}$.

Definícia 7.54 Nech je daná Kripkeho štruktúra $\mathcal{K}_{P \oplus U}$. Nech v $\mathcal{K}_{P \oplus U}$ je správne zakorenená ρ-cesta σ, ktorá terminuje v totálnej interpretácii w.

Budeme hovoriť, že σ je *význačná* ρ-cesta a w je *dobrý svet*. ■

Teraz môžeme povedať, že budeme používať význačné ρ-cesty a dobré svety ako nástroj na špecifikáciu $P \oplus U$.

Nasledujúca definícia formuluje intuitívne požiadavky na $P \oplus U$. Charakterizuje, čo zachovať z pôvodného programu P, ak je daný model M modifikujúceho programu U. Budeme definovať, čo odmietneme z pôvodného programu (tým aj, čo z neho prijmeme) a čo budú pre nás defaultové negatívne predpoklady. Pochopiteľne, odmietneme také klauzy z P, ktoré sú v konflikte s nejakou klauzou z U, podopretou v modeli M. Odmietneme aj fakty z P, ktoré sú v konflikte s modelom M. Ako defaultové predpoklady prijmeme také negatívne literály $\neg A$, ktoré nie sú podoprené.

Definícia 7.55 Nech M je interpretácia programu U. L a L' nech sú konfliktné literály. Nech P je program.

- $Rejected(M) = \{c \in P : (\exists c' \in U)\ ((hlava(c), hlava(c')$ sú konfliktné literály a $M \models telo(c')\} \cup \{(L \leftarrow) \in P : L' \in M\}$.
- $Residue(M) = U \cup (P \setminus Rejected(M))$.
- $Defaults(M) = \{\neg\ A : (\forall c \in Residue(M))\ (hlava(c) = A \Rightarrow M \not\models telo(c))\}$, kde A je atóm.

■

Definícia 7.56 Majme programy P, U a Kripkeho štruktúru $\mathcal{K}_{P\oplus U}$. Nech w je možný svet z $\mathcal{K}_{P\oplus U}$. Hovoríme, že w spĺňa *podmienku stability*, ak platí $w = Least(Residue(w) \cup Defaults(w))$. ■

Nasledujúce tvrdenie ukazuje, že svet, ktorý spĺňa podmienku stability, je modelom modifikujúceho programu U. To znamená, možno nad ním nadstaviť takú modifikáciu $P \oplus U$, ktorá rešpektuje informáciu z U.

Tvrdenie 7.57 *Nech P, U sú programy a $\mathcal{K}_{P\oplus U}$ je Kripkeho štruktúra, špecifikujúca modifikáciu programu P pomocou U.*

Ak možný svet w z $\mathcal{K}_{P\oplus U}$ spĺňa podmienku stability, potom w je model programu U.

Náčrt dôkazu: Predpokladajme, že klauza $r \in U$ nie je splnená vo w, to znamená $w \models telo(r)$, ale $w \not\models hlava(r)$. Ak $hlava(r) \cup w$ nie je konzistentné, potom $(w, w_\perp) \in \rho_1$ a w nemôže spĺňať podmienku stability. V opačnom prípade w je 3-interpretácia a existuje hrana $(w, w') \in \rho_1$ taká, že $hlava(r) \in w'$. I v tomto prípade w nespĺňa podmienku stability. ■

V $\mathcal{K}_{P\oplus U}$ môžeme ľahko identifikovať svety, ktoré spĺňajú podmienku stability (a sú modelmi U):

Tvrdenie 7.58 *Nech sú dané P a U.*

Potom: w_n je dobrý svet v $\mathcal{K}_{P\oplus U}$ práve vtedy, keď spĺňa podmienku stability.

Náčrt dôkazu:
$\Rightarrow$
Nech $\sigma = \langle w_0, w_1, \ldots, w_n \rangle$ je význačná ρ-cesta z $\mathcal{K}_{P \oplus U}$. Predpokladajme, že pri výpočte $\langle w_0, w_1, \ldots, w_n \rangle$ sme použili pravidlo $r \in \mathit{Rejected}(w_n)$. Preto w_n nemôže byť dobrý svet. To znamená, že iba pravidlá z $\mathit{Residue}(w_n)$ boli použité pri konštrukcii ρ_1-hrán.

Nech R je množina negatívnych literálov zavedených buď základnými predpokladmi vo w_0 alebo ρ_2-hranami alebo ρ_1-hranami, t.j. pravidlami, v hlavách ktorých je negatívny literál. Teda $R = \{\neg A \in \mathcal{D} : \neg A$ je zavedený pomocou ρ_2-hrany alebo $\neg A \in w_0$, ak $w_0 \neq \emptyset\} \cup \{\neg A : (\exists r \in \mathit{Residue}(r))\mathit{hlava}(r) = \neg A \wedge w_n \models \mathit{telo}(r)\}$.

Platí, že $R \subseteq Defaults(w_n)$. Predpokladajme opak a ukážeme, že w_n nie je dobrý svet: pre nejaké $\neg A \in R$ existuje pravidlo r také, že $\mathit{hlava}(r)$ je A a $w_n \models \mathit{telo}(r)$, preto $(w_n, w_\perp) \in \rho_1$.

Platí aj, že $Defaults(w_n) \subseteq R$: predpokladajme, že pre nejaké $\neg A \in Defaults(w_n)$ je $\neg A \notin R$, t.j. neexistuje pravidlo, ktoré môže zaviesť A do w_n a ani A ani $\neg A$ nepatria do w_n, preto w_n nie je totálna interpretácia, čiže nie je dobrý svet.
$\Leftarrow$
Keď $w = \mathit{Least}(\mathit{Residue}(w) \cup Defaults(w))$, potom môžeme skonštruovať význačnú cestu terminujúcu vo w. ■

Fakt 7.59 *Dobrý svet je modelom programu U.*

Je načase špecifikovať modifikovaný program $P \oplus U$. Využijeme význačné ρ-cesty a dobré svety. Dobré svety sú modelmi modifikujúceho programu, teda rešpektujú to, čo modifikujúci program vyjadruje, sú jeho (čiastočnou) sémantickou charakterizáciou.

Vo všeobecnosti môžeme každú modifikáciu realizovať rôznymi spôsobmi. Navyše, akceptujeme sémantiku stabilných modelov – keďže programy mávajú viac stabilných modelov, máme zvýšenú voľnosť v realizácii revízie. Napríklad, môžeme postupovať na základe dôverčivej stratégie alebo na základe skeptickej stratégie (medzi týmito krajnosťami sú nejaké medzipolohy). Najjednoduchší spôsob, ktorý sa nám núka na realizáciu dôverčivej stratégie: pre každý dobrý svet w z $\mathcal{K}_{P \oplus U}$ stotožniť $P \oplus U$ s $\mathit{Residue}(w)$. Dostaneme tak toľko modifikácií, koľko máme dobrých svetov. Skeptická verzia modifikácie by bola prienikom všetkých modifikovaných programov.

Iná možnosť, ako realizovať dôverčivú stratégiu je nasledovná: $\mathcal{K}_{P \oplus U}$ špecifikuje množinu $\mathcal{S}$ programov:[19] každá význačná ρ-cesta σ determinuje jeden program Π, pričom dobrý svet w zo σ je jediným stabilným modelom programu Π.

Konštrukcia programu Π: Nech $\sigma = \langle w_0, \ldots, w_n \rangle$. Pre každú hranu $(w_i, w_{i+1}) \in \rho_1 \cup \rho_2$ nech $w_i = \{L_1, \ldots, L_m\}$ a $w_{i+1} \setminus w_i = \{L'_1, \ldots, L'_k\}$. Prijmeme $L'_j \leftarrow L_1, \ldots, L_m$ do Π pre každé $j = 1, \ldots, k$.

[19] Môžeme povedať, že $\mathcal{S}$ je triedou alternatív pre $P \oplus U$.

Fakt 7.60 *Nech* Π *je skonštruované z* $\mathcal{K}_{P\oplus U}$ *nad význačnou* ρ*-cestou* σ*. Potom dobrý svet* w *zo* σ *je jediným stabilným modelom programu* Π.

Reprezentácia modifikácií množinou možností zodpovedá faktu, že v evolúcii poznania sú prirodzene implicitné isté alternatívy.

Samozrejme, existujú sofistikovanejšie možnosti, ako konštruovať $P \oplus U$. Špeciálnu pozornosť si zasluhuje myšlienka parciálnej evaluácie, pozri [Lif 94a]. V skratke ide o toto: akceptujeme program U (každý dobrý svet na význačnej ceste je jeho modelom), pridáme k nemu to, čo získame z P nasledujúcou procedúrou: nech w je daný dobrý svet, nech $c = L \leftarrow L_1, \ldots, L_k$ je pravidlo z P. Ak $w \models \mathit{telo}(c)$, potom do $P \oplus U$ pridáme $L \leftarrow$. Ak pre $G \subset \mathit{telo}(c)$ (na $\mathit{telo}(c)$ možno nazerať ako na množinu) platí $w \models G$, potom do $P \oplus U$ pridáme $L \leftarrow H$, kde $H = \mathit{telo}(c) \setminus G$.

Problém nájsť k danej množine M dobrých svetov (v danej štruktúre $\mathcal{K}_{P\oplus U}$) taký program $P \oplus U$, aby množinou všetkých jeho stabilných modelov bola práve M, je otvorený.

Dynamický logický program, ktorý zodpovedá sérii modifikujúcich programov, tiež možno špecifikovať pomocou Kripkeho štruktúr. Rozšírenie prezentovaného prístupu na konštrukciu štruktúry $\mathcal{K}_{\oplus\{P_s : s\in S\}}$ je priamočiare. Doteraz použité prostriedky stačí rozšíriť o priority, priradené hranám Kripkeho štruktúr $\mathcal{K}_{P_s}$: Ak S je lineárne usporiadaná množina, „novšie“ s sú preferované. Podobne pre iné možnosti usporiadať S.

Dynamické Kripkeho štruktúry umožňujú transformácie možných svetov v Kripkeho štruktúrach. Vďaka týmto transformáciám umožňujú sémanticky charakterizovať modifikácie. S ich pomocou možno charakterizovať veľmi silný formalizmus AELKB-logiky, ktorého špeciálnym prípadom sú mnohé rozhodujúce formalizácie nemonotónneho usudzovania. Zavedenie dynamiky podporuje výpočet stabilných autoepistemických expanzií AELKB-teórií. Popri transformáciách možných svetov je ďalším zdrojom dynamiky možnosť rekonštruovať reláciu dostupnosti v Kripkeho štruktúrach. Využili sme túto možnosť pri charakterizácii revízií AELB-teórií. Ako posledný spôsob vnášania dynamiky do sémantiky možných svetov sme uviedli operácie nad Kripkeho štruktúrami, ktorých výsledkom je opäť (nová) Kripkeho štruktúra. Charakterizovali sme takú operáciu a ukázali sme, ako s jej pomocou možno sémanticky špecifikovať dynamické logické programy.

Kapitola 8

Proti bariére nezvládnuteľnosti

Dostali sme sa do záverečnej kapitoly. Celý text uzavrieme tým, že sa pozrieme ponad jeho horizont. Doposiaľ nás zaujímala predovšetkým sémantická špecifikácia hypotetického usudzovania. Často sme však konštatovali, že máme do činenia s výpočtovo veľmi ťažkými problémami a hľadali sme aj ich zvládnuteľné prípady. Komplikované problémy reprezentácie poznatkov a usudzovania narážajú na bariéru výpočtovej nezvládnuteľnosti. Preto nad solídnymi teoretickými základmi treba hľadať kompromisné riešenia, dovoľujúce konštruovať výpočtovo prijateľné modely. Načrtneme niektoré možnosti.

Videli sme opakovane, že problémy, na ktoré narážame pri úsilí pochopiť a výpočtovo modelovať „počítanie s vetami“, sú mimoriadne zložité. Ich dôsledné formulovanie spravidla vedie k zreteľnému prekročeniu hranice zvládnuteľnosti. Doposiaľ uvádzané zložitostné výsledky sa týkajú prevažne jednotlivých formalizmov. Nie je ťažké odhadnúť, kam by viedla ambícia modelovať komplikované a realistické usudzovanie v dynamicky sa meniacich podmienkach, pri vyvíjajúcom sa poznaní.

Samozrejme, formalizmus, v ktorom problém vyjadrujeme, nie je príčinou toho, že je výpočtovo ťažký. Rovnako samozrejmé je, že bariéru výpočtovej nezvládnuteľnosti potrebujeme nejakým spôsobom zdolať. Inak by sme neboli schopní realizovať predstavy o výpočtovom modelovaní inteligencie. Cieľom je modifikovať problémy tak, aby sa stali zvládnuteľnými.

V tejto kapitole sa budeme venovať (veľmi stručne a iba v základných črtách) rôznym spôsobom, ako zdolať bariéru výpočtovej nezvládnuteľnosti. Ide o veľmi dôležitú a rozsiahlu oblasť, ktorá by si zaslúžila samostatnú monografiu, venovanú praktickej reprezentácii poznatkov. Kľúčovou súčasťou požiadaviek na takúto reprezentáciu je operatívne využiteľné a efektívne implementovateľné usudzovanie.

Treba hľadať kompromis medzi korektnými a úplnými, ale ťažkými riešeniami na jednej strane, a prakticky zvládnuteľnými (na strane druhej). Tu treba pripomenúť skutočne základný význam solídnej teórie hypotetického usudzovania pre takú reprezentáciu poznatkov, ktorá umožňuje aproximatívne, ale prakticky uskutočniteľné usudzovanie. Bez pochopenia princípov nevieme poriadne, čo aproximovať. Navyše, dôležité je vedieť, nakoľko sa aproximácia líši od korektného a úplného riešenia.

Podľa niektorých autorov nemožno vytvoriť adekvátnu teoretickú základňu pre umelú inteligenciu, ak abstrahujeme od problémov limitovaných výpočtových zdrojov [RWe 91]. S tým treba súhlasiť s jedinou výhradou, či upresnením. Adekvátnej teórii, ktorá berie na zreteľ limitované výpočtové zdroje, nechýba obraz o tom, ako je inferencia špecifikovaná a ako sa jej môžeme priblížiť (či musíme vzdialiť) pri limitovaných zdrojoch.

Navrhneme a preberieme niekoľko spôsobov, ako obísť ťažké problémy. Niekoľkokrát sme už videli, že sa to dá ich zjednodušením. Napríklad ich špecializáciou pre verzie v jazykoch s menšími vyjadrovacími schopnosťami. Ďalšou možnosťou je spustiť z nárokov na odvodzovací mechanizmus, predovšetkým z požiadavky, aby bol úplný. Stručne sa tu oboznámime s ďalšími možnosťami. Prv než prejdeme k spôsobom obchádzania ťažkých problémov, zrekapitulujeme si, o ako ťažké problémy ide.

Komplexné usudzovanie Zhrnieme predstavu o výpočtovom modeli počítania s vetami. Pôjde o tak silný model, ktorý by mohol vyjadriť a opísať základné úlohy, súvisiace s hypotetickým usudzovaním. Jednotlivé aspekty tohto modelu samy o sebe sú výpočtovo nezvládnuteľné.

Načrtneme teda, čo všetko by mal vedieť „prefektný“ výpočtový model inteligencie. Tento náčrt presahuje obsah celého predchádzajúceho textu. Pochopiteľne, nezamýšľame ho predstaviť ako realistický návrh uskutočniteľného modelu. Ideálny model počítania s vetami má napríklad tieto schopnosti:

- dedukuje,
- usudzuje s neúplnými poznatkami, generuje hypotézy, usudzuje hypoteticky,
- poradí si s výnimkami, usudzuje aproximatívne, s neučitosťou, s rôznou mierou vieryhodnosti,
- usudzuje alternatívne,
- usudzuje za prítomnosti nekonzistentnosti,
- udržiava konzistentnosť,
- asimiluje nové poznatky,

- reviduje,
- usudzuje v závislosti na kontexte,
- usudzuje v dynamicky sa meniacom prostredí a s dynamicky sa meniacimi, vyvíjajúcimi sa poznatkami,
- usudzuje o časových vzťahoch, o akciách a ich účinkoch, o priestorových vzťahoch, usudzuje v súlade s predpokladmi zdravého rozumu,
- usudzuje o vlastnom usudzovaní a o vlastných poznatkoch, usudzuje o usudzovaní a poznatkoch iných modelov, je schopný introspekcie, autoreflexie, modálneho a epistemického usudzovania, druhorádového usudzovania.

Špecifikácii služieb, schopností a funkcií, ktoré má poskytovať výpočtový model inteligencie,[1] sa venovali – priamo alebo nepriamo – napríklad [Lev 90, Gel 94, BGe 00, HMV 94]. Levesque v [Lev 90] charakterizoval usudzujúceho agenta pomocou modálej logiky s operátorom viery (belief). Túto logiku poňal ako nástroj na špecifikáciu agenta schopného usudzovať. Gelfond v [Gel 94] zavádza pojem epistemickej špecifikácie, založený na dvoch modálnych operátoroch. Pomocou epistemických špecifikácií vyjadruje schopnosti usudzovať i schopnosti introspekcie. Przymusinski v [Prz 97] ukázal, že epistemické špecifikácie sú špeciálnym prípadom jeho logiky AELKB (pozri časť 7.3).

Kombinácii rôznych logických formalizmov s cieľom špecifikovať systémy, schopné komplexného usudzovania, sa venovali napríklad [Eng 96, EnT 95, EnT 96, FiG 96]. Perspektívny cieľ takýchto výskumov je vytvorenie logiky, vhodnej na špecifikáciu systémov, schopných riešiť komplexným, kombinovaným usudzovaním nejaké úlohy.

Už sme opakovane konštatovali, že ide o veľmi bohatý, prepletený balík problémov. Ich štúdium je síce zaujímavé a dôležité, ale tak komplikované, že minimálne v súčasnom stave poskytuje iba čiastkové odpovede. Cenné však môžu byť aj čiastkové riešenia a môžu mať zaujímavé a dôležité aplikácie. Je však prirodzené, že sú tu pokusy presunúť pozornosť k praktickej realizovateľnosti. Podstatnou otázkou sa vtedy stáva, ako aproximovať ideálny model.

Skôr než prejdeme k metódam a konceptom smerujúcim k praktickej realizovateľnosti, zastavíme sa pri oblasti, ktorej sme sa doteraz venovali iba okrajovo. Je to metausudzovanie, autoreflexia. Po jej predstavení naznačíme, ako možno metausudzovanie využiť na zefektívnenie výpočtov.

[1] Pripomeňme si alternatívne označenie: Výpočtový systém, založený na spracovaní poznatkov, na počítaní s vetami. Opäť budeme používať ako skratku aj termín „agent“.

8.1 Metausudzovanie

Schonosť autoreflexie, introspekcie, usudzovania o svojich vlastných poznatkoch a procesoch usudzovania, je určite vítaná vlastnosť výpočtového modelu inteligencie (inteligentného agenta). Keď agent kooperuje s inými agentami, je dôležité aj to, aby mal reprezentované poznatky agentov, s ktorými kooperuje a aby bol schopný analyzovať ich usudzovanie a predvídať ich akcie.

Tu sa nebudeme problému autoreflexie venovať v plnej šírke. Podrobnejší text, súčasť predchádzajúcich verzií tejto práce možno nájsť na tejto adrese: `http://www.ii.fmph.uniba.sk/kri/kri.html`.[2]

Najprv základné pojmy: jazyk, v ktorom sa hovorí o nejakom jazyku J, je *metajazyk*. Jazyk J sa nazýva aj *objektovým* jazykom. Pod metausudzovaním budeme rozumieť usudzovanie v metajazyku. Niekedy je cieľom metausudzovania analyzovať nejaké usudzovanie v objektovom jazyku. Aj metajazyk môže byť objektom opisov v nejakom meta-metajazyku.

Budeme si všímať jednoduché situácie, v ktorých je vhodné zosilniť vyjadrovacie schopnosti tak, aby umožňovali vyjadrovať sa o vlastných znalostiach. Potom sa budeme venovať niektorým technikám zefektívnenia výpočtov. Tieto techniky spočívajú v špecializácii univerzálnejších nástrojov na daný problém.

Ako prvý si uvedieme príklad zosilnenia vyjadrovacích možností pre formulovanie integritných obmedzení. Hlavnou myšlienkou je, že nekonzistentnosť, narušenie integritného obmedzenia sa nezakazuje, ale jej vznik je signálom pre nejaké akcie, ktoré majú vzniknutú situáciu napraviť. S problémom prišli [GaH 93] a riešia ho tak, že zavádzajú popri objektovom jazyku, v ktorom sa vyjadruje vlastná databáza, aj metajazyk, v ktorom riešia prípadné vzniknuté nekonzistennosti. Metajazykom je temporálna logika (logika pracujúca s operátormi, umožňujúcimi vyjadriť časové vzťahy). Minulosť a prítomnosť opisuje ich logika deklaratívne. Budúcnosť imperatívne – ako špecifikáciu akcií, ktoré treba vykonať, aby sa vyriešili problémy, spôsobené narušením integritných obmedzení. Metajazykové vyjadrovacie schopnosti teda zosilňujú schopnosti bežnej databázovej mašinérie. Dôležité je, že ju robia flexibilnejšou pre praktické použitia.

Ďalším z nástrojov, umožňujúcich autoreflexiu, je modálna alebo epistemická logika. Už sme videli, že používanie operátorov K (viem) a B (verím, môžem pripustiť) tiež zosilňuje vyjadrovacie schopnosti jazyka na reprezentáciu poznatkov. Zosilnenie o modálne (epistemické) operátory môže plniť podobnú úlohu ako zosilnenie o metajazykové výrazy. Epistemické logické programy, pozri [BGe 94], sú ďalším z rozšírení logických programov. Pomocou epistemických logických programov môžeme adekvátne vyjadriť také poznatky, ktoré bez ich zavedenia spôsobujú ťažkosti pri reprezentácii.

Prejdeme k metaprogramovaniu. Začneme jednoduchým metaprogramom. Metaprogram je program, ktorého vstupom je nejaký program. Spracúvanie

[2] Postupne tam budú pribúdať ďalšie doplnky (a opravy) k textu tejto knihy.

programu iným programom je typické pre výpočtové modelovanie metausudzovania.[3]

Príklad 8.1 Budeme používať metajazykový predikát *solve*, ktorý nadobúda ako argumenty ciele (otázky adresované nejakému prologovskému programu v pozadí). Ak *Goal* je prologovská otázka, potom $solve(Goal)$ je otázka pre nasledujúci metaprogram.

$$\begin{aligned} solve(true) &\leftarrow \\ solve((A,B)) &\leftarrow solve(A), solve(B) \\ solve(A) &\leftarrow clause(A,B), solve(B) \end{aligned}$$

Tento metaprogram simuluje prologovský výpočet definitných programov. Predikát $clause(A,B)$ je pravdivý vtedy, keď v objektovom programe existuje klauza, ktorej hlava je unifikovateľná s A a jej telo je po tejto unifikácii B. ∎

Uvedený metaprogram nie je veľmi zaujímavý, iba ilustruje ideu metaprogramovania a charakterizuje výpočtovú stratégiu Prologu. Nie je však až taký problém túto stratégiu zmeniť: napríklad klauzou

$$solve((A,B)) \leftarrow solve(B), solve(A).$$

Metaprogram *solve* si trochu skomplikujeme – bude schopný rekonštruovať priebeh výpočtu a dať o ňom informáciu.

Príklad 8.2 Argumentom predikátu $solve(Goal, ProofTree)$ je okrem cieľa *Goal* aj *ProofTree*, ktorý reprezentuje dôkazový strom z výpočtu odpovede na daný cieľ.

$$\begin{aligned} solve(true, true) &\leftarrow \\ solve((A,B),(PrfA, PrfB)) &\leftarrow solve(A, PrfA), solve(B, PrfB) \\ solve(A,(A \longleftarrow Proof)) &\leftarrow clause(A,B), solve(B, Proof) \end{aligned}$$

∎

Postupným zosilňovaním tohto predikátu možno vytvoriť jadro expertného systému, schopné interpretovať bázy znalostí napísané v Prologu, poskytovať vysvetlenia, klásť otázky vtedy, keď to je potrebné (a možné), získať odpovede z databázy, ak odpoveď je už známa a podobne.

Tvorba a používanie metaprogramov dáva práve takéto možnosti – modifikáciu výpočtovej stratégie, vytvorenie prostredia pre komunikáciu s užívateľom, vysvetľovanie a pod. Nižšie uvedené pravidlo sa aktivuje vtedy, keď pre cieľ A predikát $clause(A,B)$ zlyhá.

[3] Jednou z výhod, ktorú prináša modulárne, deklaratívne vyjadrenie bázy znalostí (množiny explicitne reprezentovaných viet E), je práve uľahčenie prístupu tzv. „introspektívnych“ (meta)programov k poznatkom, k ich analýze, flexibilnému využívaniu a ku kontrole inferencie s nimi.

Príklad 8.3 Zárodok metaprogramu, komunikujúceho s užívateľom:

$$solve(A) \leftarrow askbl(A), not\ known(A), ask(A, Odpoveď), rspnd(Odpoveď, A).$$

Predikát *askbl* odlišuje ciele, pre ktoré možno získať riešenie otázkou užívateľovi od ostatných cieľov; ak odpoveď ešte nie je známa, nasleduje otázka užívateľovi; predikát *rspnd* umožní zapamätanie odpovede, aby sa otázka neopakovala ďalší raz. ■

O metaprogramovaní (a metainterpretácii) viac napríklad v [StS 87].

Nasledujúce poznámky sa budú týkať techník, ktoré metajazykový rámec ponúka na zefektívnenie univerzálnych výpočtových metód. Tieto techniky spočívajú v špecializácii všeobecného mechanizmu na práve riešenú inštanciu problému. Môžeme to považovať za prípad metausudzovania: pre daný program a danú inštanciu problému sa postavíme na meta-úroveň a usilujeme sa nájsť špecializáciu daného programu, ktorá je z hľadiska danej inštancie efektívnejšia.

Metaprogram vo všeobecnosti zavádza nové jazykové a výpočtové možnosti, ktoré základný jazyk (v našom príklade Prolog) neposkytuje. Bázu znalostí, ako aj reprezentačné služby, možno písať v takto zosilnenom prostredí. Poskytuje to schopnosť implementovať metausudzovanie, prepínať medzi rôznymi metaprogramami v závislosti na behu výpočtu. Prostredie tohto druhu sa dá budovať inkrementálne, ako séria metaprogramov. Za zosilnenie reprezentácie znalostí sa však platí znižovaním efektívnosti výpočtov, ktorá sa zhoršuje každou vrstvou intepretácie. Jedna z techník, čo v tejto situácii pomáha, je parciálna evaluácia.

Parciálny evaluátor dostáva na vstupe nejaký metaprogram spolu s programom, ktorý je vstupom pre daný metaprogram. Transformuje ich do špecializovaného tvaru, zodpovedajúceho danej úlohe. Tento špecializovaný tvar možno vykonať efektívnejšie.

Príklad ďalšieho metaprogramu možno nájsť v [Bry 90]. Ide o metaprogram *Satchmo*, ktorý realizuje pomocou metapredikátu *new* modifikácie deduktívnej databázy. Použitá myšlienka bola abdukcia: danú požiadavku na modifikáciu deduktívnej databázy možno splniť rôznymi spôsobmi, ak sa požiadavka týka intenzionálnych predikátov, ktoré sú definované pomocou pravidiel. Modifikáciu možno vykonať abduktívne.

Kombinácii abdukcie a indukcie, založenej na metaprogramovaní sa venoval [Chr 00]. Využitím metajazyka v učení a indukcii sa zaoberali [AFr 99].

Ako ďalší príklad na špecializáciu všeobecnejšej výpočtovej mašinérie možno uviesť zmiešané výpočty. Opäť sa uplatňuje autoreflexia a cieľom je efektívnejší výpočet danej inštancie problému. Základné typy výpočtov v deduktívnej databáze sú zhora nadol (od daného cieľa k faktom, ktoré umožňujú odpoveď) alebo zdola nahor (od všetkých faktov postupne k cieľu).

Zmiešané metódy výpočtu zlučujú prednosti oboch základných spôsobov výpočtu a redukujú ich nevýhody.

Jednou z takýchto zmiešaných metód výpočtu je metóda magických množín. Počas výpočtu zhora nadol sa vytvárajú nejaké vzťahy, väzby medzi cieľmi a pravidlami a nejakým spôsobom „tečú informácie“. Na základe týchto väzieb sa konštruuje pre daný cieľ modifikovaná syntaktická verzia pôvodného programu, ktorý sa potom počíta zdola nahor. Nový syntaktický tvar umožňuje zachovať toky informácie zhora nadol a tým redukovať potrebný výpočet. Súčasne umožňuje zachovať prednosti výpočtu zdola nahor, pozri [Ull 88].

Problémy vzťahu výpočtu zhora nadol a zdola nahor sú jednou z dôležitých tém umelej inteligencie. Príklad z deduktívnych databáz môže byť poučný minimálne v jednom smere: ak máme nejakú všeobecnú mašinériu, môžeme jej výkon zvyšovať tak, že skonštruujeme jej špecializáciu na práve riešenú inštanciu problému. Konštrukciu tejto špecializácie možno považovať za príklad metausudzovania.

Stručne sme si teda predstavili metausudzovanie ako súčasť komplexnej výbavy pre reprezentáciu poznatkov a komplikované usudzovanie. Poukázali sme aj na to, že metausudzovanie možno využiť na zefektívnenie výpočtov. Na niekoľkých jednoduchých príkladoch sme ukázali, že vďaka kontrole výpočtov v metajazyku možno všeobecné a výpočtovo náročné mechanizmy špecializovať tak, aby ich aplikácia na danú konkrétnu úlohu bola efektívnejšia. V časti 8.4 sa dostaneme k mimoriadne dôležitému smeru výskumov, ktorý si zaslúži samostatnú pozornosť. Ich hlavným cieľom je tiež zefektívnenie výpočtov vďaka metausudzovaniu.

Ďalšie možnosti zefektívňovania výpočtov spočívajú v obchádzaní ťažkých problémov a ich nahrádzaní (aproximovaní) ľahšími problémami. Začneme oblasťou, ktorá sa ešte prednedávnom považovala za vlastnú doménu výskumu reprezentácie poznatkov. Dnes je už v pozadí.

8.2 Reprezentačné schémy

Je načase vysloviť otázku, ktorú si informovaný čitateľ kládol pravdepodobne už dávnejšie: aký je vzťah tém, analyzovaných v tejto knihe k tradičným témam, k tradičným prostriedkom reprezentácie znalostí ako sú napríklad sémantické siete a rámce.

Pre znalostné systémy je dôležité navrhnúť formalizmy na reprezentáciu znalostí, využívajúce údajové štruktúry, ktoré sú výhodné z hľadiska výpočtovej efektívnosti. Pre tieto formalizmy si zavedieme pojem reprezentačnej schémy.

Pojem reprezentačnej schémy nechceme precizovať nadmieru dôkladne. Bude fungovať skôr pomocne, má pokrývať mnohé formalizmy. Niektoré vznikli ako produkty experimentov, sú založené na intuitívnych východiskách a ich formálny popis je sťažený. Na základe praktických skúseností sa intuitívne odhaduje, že sú výpočtovo zvládnuteľné. Dôležitými prvkami reprezentačných schém sú:

- inferenčné stratégie, ktoré efektívne implementujú inferenčné operácie,
- niektoré predpripravené vysokoúrovňové konštrukty na reprezentáciu znalostí,
- organizácia množiny viet explicitnej dátabázy, uľahčujúca dostup k hľadaným vetám (indexovanie, členenie na segmenty - prípadne s diferencovaním jazykov pre rôzne segmenty).

Reprezentačné schémy však nebývajú dobre špecifikované, často im chýba jasný popis jazyka, sémantiky a inferencie, ako aj dôkladné, presvedčivé odvodenie ich výpočtovej zvládnuteľnosti. Sémantika reprezentačného jazyka nebýva špecifikovaná dostatočným spôsobom, zväčša je charakterizovaná iba pomocou údajových štruktúr, ktoré ju implementujú.

Napríklad sémantické siete sa vyskytujú v rôznych – niekedy ťažko porovnateľných – verziách a implementáciách. Dominuje procedurálny prístup. To, čo sa o nich vie, sú použité údajové štruktúry a procedúry, ktoré s nimi manipulujú. Keď treba údajovú štruktúru mierne predefinovať, ad hoc sa tak urobí. Keď správanie procedúry nie je uspokojujúce, ad hoc sa modifikuje. Samozrejme, modifikácie sa robia aj v deklaratívne vyjadrenej reprezentácii. Podstatný rozdiel je však v tom, že pri procedurálnom prístupe sa komplikuje celkový a jednotiaci pohľad na údajové štruktúry, ako aj manipulácie s nimi, nie je zaručená koherentnosť prístupu (implementácie). Možno nájsť príklady sémantických sietí bez dobre špecifikovaného jazyka a inferenčnej mašinérie.[4] Niekedy nie je jasné, ako sémantická sieť môže byť konštruovaná (a ako už nemôže byť konštruovaná). To, čo z nej odvodíme, niekedy nevyhovuje ani veľmi liberálnym kritériám. Tieto nedostatky sú spôsobené tým, že implementačné rozhodnutia ovplyvňujú vlastnosti jazyka a inferencie nad ním. Inými slovami, jazyk a inferenčný stroj majú ad hoc charakter – závisia často na implementačných náhodilostiach. Často sa voľne hovorí o výpočtovej efektívnosti reprezentačných schém bez toho, že by boli známe nejaké ich zložitostné charakterizácie. (Pravdepodobne niekedy nie sú možné kvôli ich nedostatočnému opisu. Ak sa však urobia, môžu prekvapiť – údajne efektívne výpočty sa ukážu byť nezvládnuteľnými, pozri časť 7.1.3.)

Zhrnieme predstavu o sémantických sieťach z kapitoly 4: Definovali sme ich nejakými orientovanými ohodnotenými grafmi (hypergrafmi). Čo je podstatné, ekvivalentne ich možno vyjadriť v nejakom logickom jazyku s presne definovaným operátorom odvodenia (ako aj s dobre definovanou množinou indexov na formulách). Z tohto hľadiska sa sémantické siete dajú považovať za syntaktické varianty fragmentu jazyka predikátového počtu, obohatené o indexovanie. Inferenčný stroj pritom býva limitovaný v porovnaní s korektnou a úplnou dôkazovou procedúrou predikátovej logiky. Zato však do-

[4] My poznáme sémantické siete pod názvom hierarchické siete. Zoznámili sme sa aj s ich ekvivalentným vyjadrením v jazyku logiky, čím získali jasnú sémantickú charakterizáciu.

voľuje efektívnejšie odvodzovanie.[5] Z tohto hľadiska ich možno vnímať ako špecializované logické jazyky.

Podobne to je s rámcami. Štruktúry typu rámca možno nájsť v dodatku A. Uvedieme tam použitý príklad.

```
NAME:   LIST-EQUAL
IS-A:   (PREDICATE FUNCTION OP BINARY-PREDICATE
        BINARY-FUNCTION BINARY-OP ANYTHING)
GENL:   (SET-EQUAL BAG-EQUAL OSET-EQUAL STRUC-EQUAL)
SPEC:   (LIST-OF-EQ-ENTRIES LIST-OF-ATOMS-EQUAL EQ)
FAST-ALG: (LAMBDA (x y) (EQUAL x y))
RECUR-ALG:(LAMBDA (x y)
            (COND  ((OR (ATOM x) (ATOM y)) (EQ x y))
                   (T (AND
                        (LIST-EQUAL (CAR x) (CAR y))
                        (LIST-EQUAL (CDR x) (CDR y))
                      )
            )
        )
DOMAIN: (LIST LIST)
RANGE:  TRUTH-VALUE
WORTH:  720
```

Uvedený príklad možno v jazyku logiky rekonštruovať pomocou sady formúl s binárnymi predikátmi. Každý predikát zodpovedá menu aspektu (IS-A, GENL atď.). Prvým argumentom bude vždy meno uvedeného konceptu (rámca), t.j. LIST-EQUAL a druhým argumentom bude hodnota príslušného aspektu. Samozrejme, ak by rámce boli štrukturované hlbšie, uvedenú schému logickej reprezentácie by bolo potrebné skomplikovať.

Dôležitým konštruktom je inštancia rámca. Pri takomto rozlišovaní by ďalším argumentom pre prototypový rámec bola premenná, charakterizujúca typ a pre inštancie tohto rámca by sa za túto premennú substituovala príslušná konštanta.

Na rámcoch je dôležité, že vytvárajú zhluky relevantných znalostí. Tieto zhluky sústreďujú všetky poznatky zaujímavé a dôležité z istého hľadiska (tým sa aj implementuje akési hľadisko na poznatky). Jadrom inferencie sa stáva vyhľadávanie, zosilnené o defaulty, dedenie a pripojené procedúry. Pomocou pripojených procedúr sa realizujú niektoré požiadavky ako vyhľadanie hodnoty, pridanie hodnoty, vymazanie hodnoty. Problému logickej rekonštrukcie rámcov sa venovali viacerí autori. Ako príklad spomenieme [Hay 79, GeN 87].

Vyhľadávanie je podporované indexovaním. Indexovanie možno skonštruovať aj na formulách, videli sme to pri hierarchických sieťach, časť 4.2. Problematickými miestami logickej rekonštrukcie rámcov sú defaultové hodnoty a pripojené procedúry. Pri defaultových teóriách sme videli, že dôsledné

[5] Opäť poznamenajme, že výpočtové vlastnosti predikátového počtu nie sú spôsobené jeho jazykom, ale problémami, ktoré treba riešiť.

narábanie s defaultami je konceptuálne i výpočtovo ťažký problém. Naivné rámcové systémy si to neuvedomujú. Pripojené procedúry mávajú ad hoc povahu, nemusia byť transparetné, prístupné sémantickej analýze.

Vo všeobecnosti však možno očakávať, že podiel inferencie pri rámcovej reprezentácii nie je dominantný (pokiaľ, samozrejme, s rámcami neasociujeme pravidlá). Prevláda vyhľadávanie. Problémy, ktoré treba riešiť v súvislosti s rámcovou reprezentáciou, sú asi z výpočtového hľadiska ľahšie ako problémy súvisiace s inferenciou. Treba však istú mieru opatrnosti, keď takéto niečo tvrdíme:

Rámce majú slúžiť ako príklad špecializovanej reprezentačnej schémy, ktorej vyjadrovacia a odvodzovacia sila je limitovaná, ale efektívna. Nasledujúca analýza ukáže, že s ich efektívnosťou to nie je až tak jednoduché.

Ďalší príklad toho, že už nepatrné zosilnenie reprezentačného formalizmu vedie k vážnemu zhoršeniu výpočtových možností tohto prostriedku ukázali [LeB 85]. Definovali jazyk na popis rámcov FL. Jazyk je jednoduchý a je definovaný nasledovnou gramatikou.

```
<type> ::= <atom>
 | (AND <type1> ... <typen>)
 | (ALL <attribute> <type>)
 | (SOME <attribute>)

<attribute> ::= <atom>
| (RESTR <attribute> <type>)
```

Sémantika tohto jazyka je daná funkciou $\mathcal{E}$ (a doménou $\mathcal{D}$) tak, že

- $\mathcal{E}(t) \subseteq \mathcal{D}$ pre ľubovoľný typ t,
- $\mathcal{E}(a) \subseteq \mathcal{D} \times \mathcal{D}$ pre ľubovoľný atribút a.

Pritom

$$\begin{aligned}
\mathcal{E}((AND\, t_1 \ldots t_n)) &= \bigcap_i \mathcal{E}(t_i), \\
\mathcal{E}((ALL\, a\, t)) &= \{x \in \mathcal{D} : < x, y > \in \mathcal{E}(a) \Rightarrow y \in \mathcal{E}(t)\}, \\
\mathcal{E}((SOME\, a)) &= \{x \in \mathcal{D} : \exists y (< x, y > \in \mathcal{E}(a))\}, \\
\mathcal{E}((RESTR\, a\, t)) &= \{< x, y > \in \mathcal{D} \times \mathcal{D} : < x, y > \in \mathcal{E}(a) \wedge y \in \mathcal{E}(t)\}.
\end{aligned}$$

Typ t_1 je špeciálnejší ako typ t_2 práve vtedy, keď pre každé $\mathcal{E}$ a pre každé $\mathcal{D}$ platí $\mathcal{E}(t_1) \subseteq \mathcal{E}(t_2)$. Pri dedení je kľúčovým problémom zistenie, či nejaký typ je pod(nad)typom iného typu. Pozrime sa teda bližšie na tento problém (je známy pod menom problém subsumcie).

Nech $\mathcal{FL}^-$ sa líši od $\mathcal{FL}$ iba tým, že neobsahje operátor $RESTR$.

Existuje $\mathcal{O}(n^2)$ algoritmus rozhodujúci o subsumcii vo $\mathcal{FL}^-$, ale problém subsumcie je co-NP-ťažký vo $\mathcal{FL}$. Patel-Schneider [PaS 89] neskôr pre podobný formalizmus dokázal, že subsumcia v ňom je dokonca nerozhodnuteľná.

Priamym potomkom sémantických sietí a rámcov sú hybridné systémy.

Hybridné systémy Hybridné systémy kombinujú rôzne typy formalizmov. Klasickým z nich je KRYPTON [BFL 85].

V bežných reprezentačných schémach typu sémantických sietí a rámcov bývajú problémy s rozlíšením faktuálnej informácie a definičnej informácie. KRYPTON zavádza ostré rozlíšenie definícií a faktov. Jeho reprezentácia znalostí má dve samostatné časti, TBox a ABox. Každá z nich je napísaná v inom jazyku.

TBox vytvára taxonómiu štrukturovaných termov (typu rámca). Adresované sú mu otázky, týkajúce sa vzťahov medzi termami. Základnými konštruktami jazyka TBoxu sú koncepty. Rozlišujú sa generické a individuálne koncepty. Internú štruktúru konceptov tvoria roly a opisy. Roly zodpovedajú aspektom z rámcov. Opisy dávajú informáciu o tom, ako hodnoty rolí vzájomne interagujú. Samozrejme, na konceptoch je definovaná dedičnosť.

ABox obsahuje teóriu príslušnej aplikačnej oblasti. Adresované sú mu otázky, ktoré sa týkajú aplikačnej oblasti. Jazyk ABoxu je jazyk logického typu, so schopnosťou vyjadriť neúplné poznanie (negáciu, disjunkciu, existenčný kvantifikátor).

Druhý vážny problém – potrebu špecifikovať sémantiku reprezentačných konštruktov inak ako údajovými štruktúrami, použitými pri ich implementácii, rieši KRYPTON funkcionálnym pohľadom na bázu znalostí. Otázka, aké údajové štruktúry systém potrebuje na reprezentáciu znalostí, je podľa tohto prístupu druhoradého významu. Prvoradé otázky sa týkajú špecifikácie toho, čo má systém robiť. V [BFL 85] sa ako základné operácie (ktorými sa tieto funkcie špecifikujú) uvádzajú TELL a ASK. Pomocou týchto operácií sa špecifikuje funkčnosť znalostného systému (nad danou bázou znalostí, napísanou v akomsi reprezentačnom jazyku). TELL špecifikuje modifikácie, vkladanie novej informácie, ASK špecifikuje prehľadávanie (odpovedanie otázok).

Teda, hybridná reprezentácia znalostí pozostáva z dvoch (alebo viacerých) rôznorodých komponentov, pričom každý z nich plní iné úlohy v reprezentácii znalostí.

Logickú rekonštrukciu reprezentačnej schémy typu KRYPTON (a ďalších) poskytujú systémy sortovej dedukcie [Fri 89, Coh 89]. Inšpirácia pre tento prístup pochádza zo zavedenia typovej inferencie do automatických dokazovačov, s ktorou sme sa už stretli [Wal 85]. Systémy poskytujú prepojenie bežnej logickej dedukcie s inferenciou nad sortami. Informácia o sortách sa využíva vtedy, keď sa robia substitúcie, preto sa tento logický systém nazýva substitučným. V podstate ide o mnohosortový logický formalizmus s modifikovanou procedúrou unifikácie a rezolvencie, čo prináša zlepšenú výpočtovú efektívnosť.

Na priekopnícky podnet KRYPTONu nadväzuje celá trieda reprezentačných jazykov. Nazývajú sa terminologickými, vrcholili experimentálnymi implementáciami triedy reprezentačných formalizmov typu KL-ONE [Bra 85]. Teoretické aspekty terminologických formalizmov sú dodnes predmetom intenzívneho výskumu (päť príspevkov na IJCAI'99, [PAI 99], 84-115). Študujú sa na formalizmoch, pre ktoré sa ustálilo pomenovanie logiky deskripcie (description logics).

8.3 Alternatívy inferencie

Intenzívna reprezentácia Pojem „intenzívnej reprezentácie“ (vivid representation), [Lev 86, EBB 89] bol zavedený kvôli výpočtovým ťažkostiam s inferenciami. Motiváciou je okolnosť, že bežné ľudské myslenie je rýchle, používa skratky, ide priamo k riešeniam, zdĺhavé inferencie nie sú bežné.

Hlavný vtip „vivid-prístupu“ spočíva v preferovaní takej reprezentácie, takých znalostí, ktoré maximalizujú nachádzanie odpovedí na dotazy priamo v báze znalostí a minimalizujú potrebné inferencie. Teda ideálom je, aby nájdenie odpovede na otázku bolo na úrovni dátabázového vyhľadávania.

S predstavou intenzívnej reprezentácie sa vynára lákavá možnosť ľubovoľnú reprezentáciu, využívajúcu inferenciu, preložiť do tvaru, kde treba iba vyhľadávať: skonštruovať model, extenziu, stabilný model a tam vyhľadávať. Ak odhliadneme od existencie viacerých extenzií alebo stabilných modelov, je tu jedna zásadná ťažkosť. Už sme sa s ňou zoznámili v kapitole 4 – je to fakt, že nie každá reprezentácia je kompilovateľná do priestoru polynomicky väčšieho než je pôvodná reprezentácia [CDS 95, GKP 95]. Kompilácia umožňuje oddeliť problémy, ktoré sa dajú riešiť off-line, od problémov, ktoré je nutné riešiť on-line. Off-line možno prekompilovať znalosti, nutné pre riešenie problému do tvaru, v ktorom sa na otázky dá odpovedať on-line v polynomickom čase.

Usudzovanie, založené na prípadoch Jedným zo spôsobov minimalizácie odvodzovania je usudzovanie, opierajúce sa o podobnosť na známe prípady. Je výhodné najmä v situáciách, kde nie je známy všeobecný opis, všeobecné pravidlá, ale sú známe nejaké prípady úspešného riešenia problémov daného typu. Vtedy možno analýzou úspešného prípadu a jeho adaptáciou na novú situáciu dosiahnuť úspech. Takto rozmýšľa aj človek – často rieši problémy tak, že si spomína na podobné prípady a využíva ich.

Všeobecná charakterizácia takéhoto typu usudzovania môže vyzerať takto: Je daná databáza prípadov (s objektami heterogénnej štruktúry). Je daný popis aktuálneho problému v danom reprezentačnom jazyku.

S uvedenými dátami pracujú dva mechanizmy, jeden vyberá vhodné prípady, druhý ich adaptuje na danú situáciu.

Mechanizmus na vyhľadávanie relevantných prípadov má za úlohu identifikovať, ktoré prípady sa podobajú na aktuálny problém. Úspešnosť tohto modulu závisí od umnej indexácie prípadov (zložito štrukturovaných objektov) a od vhodnej konceptuálnej štruktúry indexu.

Napokon, je k dispozícii mechanizmus adaptácie a testovania prípadov, ktoré boli vybraté z bázy prípadov ako vhodní kandidáti na riešenie daného problému. Vybratý prípad sa adaptuje tak, aby poskytoval riešenie daného problému. Potom sa testuje, či adaptovaný prípad vyhovuje požiadavkám na riešenie. Usudzovanie na základe prípadov využíva analógiu (pri vyhľadávaní podobných prípadov a pri adaptácii).

Formálna rekonštrukcia usudzovania založeného na prípadoch je prezentovaná v [KKS 95]. Tento typ usudzovania je stále predmetom intenzívneho výskumného záujmu – na IJCAI'99 [PAI 99] bolo prezentovaných šesť prác z tejto oblasti.

Testovanie modelov Myšlienka intenzívnej reprezentácie (a obchádzania inferencie tam, kde je zbytočná) sa využíva aj pri aplikácii techniky testovania modelov na reprezentáciu poznatkov a na usudzovanie. Možno povedať, že testovanie modelov vytvára rámec pre dobré pochopenie myšlienky intenzívnej reprezentácie.

Techniku testovania modelov zaviedli [CES 86] na verifikáciu konečno stavových konkurentných (súbežných) systémov. Prakticky sa používa aj pri testovaní integrovaných obvodov obrovského rozsahu.

Halpern a Vardi [HaV 91] navrhli reprezentovať poznatky (nejakého agenta) nie množinou explicitne reprezentovaných formúl a formúl, ktoré z nich vyplývajú, ale pomocou modelov. Na otázku, či nejaká veta platí, sa odpovedá testovaním, či je splnená v danom modeli. Ak máme danú formulu výrokovej logiky ϕ a nejaký model M, testovanie, či ϕ je v M splnená, zbehne v lineárnom čase. Samozrejme, pri netriviálnych problémoch prebieha testovanie nad nejakou triedou modelov a výpočtová zložitosť sa zhoršuje.

To, či je adekvátne a výhodné aplikovať testovanie modelov alebo odvodzovanie formúl, závisí na danej aplikačnej oblasti. Testovanie modelov preferujeme, ak sa dá pomerne jednoducho určiť model (alebo trieda modelov), zodpovedajúci danej situácii a možno ho presne opísať. Niekedy na opis danej situácie treba príliš silný logický jazyk, skonštruovaním modelu sa dá vyhnúť zbytočne komplikovaným problémom. Keď v danej aplikačnej oblasti dominujú problémy, na riešenie ktorých je zbytočné preverovať vyplývanie, technika testovania modelov poskytuje prostriedky, ktoré nie sú bohatšie než je nevyhnutné.

Na IJCAI'99 boli prezentované tri práce [Ros 99, KoL 99, BaG 99], aplikujúce testovanie modelov na analýzu nemonotónnych logík. Získané výsledky ukazujú, že testovanie modelov je aj v prípade nemonotónnych formalizmov problémom ľahším z výpočtového hľadiska než inferencia. Napriek tomu, táto technika nezaručuje zvládnuteľné výpočty pre ľubovoľný typ problému.

Balansovanie V texte sme videli, ako sa reprezentácia pomocou tabuliek (kapitola 2) alebo grafov (časť 4.2) dá ekvivalentne vyjadriť v logike. Problémy, ktoré sa riešia v týchto prípadoch v logickom formalizme sú rovnako ťažké (ľahké) ako v „intuitívnom“ formalizme tabuliek či grafov. V prvom prípade ide vlastne o testovanie modelu (inštancia databázy je tým modelom), nezaujíma nás, čo vyplýva z každej možnej inštancie databázy. V druhom prípade sme sa stretli s logikou limitovanej sily.

Na druhej strane, je tu aj tendencia preložiť logickú reprezentáciu do intenzívnej reprezentácie, v ktorej sa zaobídeme bez inferencie. Hranice takejto kompilácie pochopili [CDS 95].

Ľahšie a ťažšie problémy sú tu, nezávisia na formalizme. Dôležité je vyberať (reprezentačné) formalizmy tak, aby boli primerané problémom v aplikačnej doméne.

Preformulovanie z tabuliek či grafov do logiky nebolo samoúčelné. Dovoľuje lepšie pochopiť daný reprezentačný formalizmus. Ďalej, vo všeobecnejšom a ľahšie zosilniteľnom formalizme sa možno prirodzene pokúšať o zovšeobecnenia a o analýzu ďalších typov problémov. Tento posun, toto zovšeobecnenie oveľa ľahšie možno urobiť vo formalizme s jasne špecifikovanou syntaxou a sémantikou. Analýzy umelej inteligencie z tejto flexibility a univerzálnosti ťažia.

8.4 Limitované zdroje

Vraciame sa k téme ovplyvňovania efektívnosti výpočtov metausudzovaním. Niekoľko poznámok venujeme mimoriadne dôležitej oblasti výskumu i praktických aplikácií.

Netriviálne výpočtové modely inteligencie musia vyvažovať potrebnú rýchlosť a korektnosť. Na jednej strane, niektoré veci nemá zmysel robiť, ak sa neurobia dostatočne rýchlo. Na druhej strane, rýchla reakcia sa môže fatálne znehodnotiť chybovosťou.

Štúdium inteligentných systémov, ktorých zdroje sú obmedzené, sa stáva čoraz aktuálnejším. Problémy metapoznatkov, metausudzovania, limitovaných zdrojov, limitovanej racionality sú významnými výzvami pre teóriu reprezentácie znalostí a inferencie. Patria medzi kľúčové problémy základov umelej inteligencie.

Najjednoduchší prístup k obmedzeným zdrojom spočíva v limitovaní dĺžky odvodenia. Dĺžka odvodenia je náhodná syntaktická črta, ťažko možno definovať sémantiku.

Ďalšou možnosťou je uskromniť sa a namiesto korektnej a úplnej mašinérie pracovať s odvodzovacím strojom, ktorý nie je korektný a/alebo úplný. Tento prístup má tiež problémy so sémantikou. Je totiž zrejmé, že rezignoval na splnenie sémantickej špecifikácie.

Zaujímavý je výskum aproximácií teórií v súvislosti s technikou kompilácie [SKa 96]. Vychádza sa z reprezentácie vo všeobecnom, nelimitovanom formalizme. Tá sa kompiluje do limitovaného prostredia, ktoré dovoľuje efektívnu inferenciu. Nové je, že v prípade, keď takáto kompilácia nie je možná, systém hľadá najlepšiu aproximáciu pôvodnej reprezentácie. Jednoduchou ilustráciou tejto techniky je aproximácia teórií v jazyku výrokovej logiky hornovskými teóriami.

Nasleduje posledný smer výskumov, o ktorom budeme informovať. Dôležitá východisková myšlienka je, že výpočtovému modelu samotnému treba zveriť rozhodnutie o tom, aké výpočty bude robiť s limitovanými prostriedkami, ktoré má k dispozícii [RWe 91, Zil 95].

Sústredíme sa na pojem flexibilného algoritmu (neviem vhodne preložiť termín *anytime algorithm*). Často sa stáva, že algoritmus skôr než dosiahne úplné riešenie, má k dispozícii celkom slušné riešenie. Na tom je postavená myšlienka flexibilného algoritmu. Ten dovoľuje vypočítať najlepšie možné riešenie problému, ak z časových dôvodov nie je možné, aby výpočet prebiehal do definitívneho ukončenia. Teda, flexibilný algoritmus môže vyvažovať čas, ktorý má k dispozícii a kvalitu riešenia. Ukončenie činnosti je možné buď na základe dohody o pridelenom čase alebo na základe prerušenia. Po prerušení môže algoritmus pokračovať v činnosti (riešenie vylepšiť), ak je to možné a potrebné. Základnú informáciu možno nájsť napríklad v [Gra 96]. Na výpočet revízií použila flexibilný algoritmus [Wil 97]. Na diagnostické usudzovanie, presnejšie na aproximáciu propagácie booleovských obmedzení použili flexibilný algoritmus [VHT 00].

O logickú špecifikáciu usudzovania s obmedzenými zdrojmi sa pokúsili [FGh 99].

8.5 Komentáre

Výpočtové experimenty, relevantné pre teóriu reprezentácie poznatkov a pre hypotetické usudzovanie sa stávajú stále dôležitejšími. V mnohých výskumných centrách sa ako veľmi aktuálna pociťuje potreba vytvoriť pôdu pre praktické implementácie.

Celkom na záver preto upozorním na príznačný fakt – v posledných rokoch narastá počet podujatí, na ktorých sa prezentujú implementácie systémov, schopných usudzovať hypoteticky alebo sa prezentujú analýzy výpočtových aspektov tohto usudzovania. Dôležité je, že sa to deje na pôde, kde sa zdôrazňuje a vyžaduje teoretická fundovanosť a korektnosť predložených implementácií.

V roku 1998 pri príležistosti svetovej konferencie Principles of Knowledge Representation and Reasoning v Trente prebiehal workshop o nemonotónnom usudzovaní. Jedna jeho časť bola venovaná výpočtovým aspektom. Prezentovaných bolo desať prác.

Sú dostupné na `http://saturn.hut.fi/~ini/cnmr98.html`.

V roku 1999 boli na workshope Logic-Based Artificial Intelligence predstavené štyri systémy na výpočet stabilných modelov logických programov (dlv, smodels, DeRes, CCALC), webovské adresy na prvé tri sme už uviedli. Posledný je dostupný na `http://www.cs.utexas.edu/users/mccain/cc`.

Napokon, zatiaľ najväčší workshop sa uskutočnil v apríli 2000 v Breckenridge, Colorado, USA (popri ďalšej svetovej konferencii Principles of Knowledge Representation and Reasoning). V sekcii demonštrácií a opisov implementovaných systémov sa predstavilo štrnásť systémov. So všetkými sa možno zoznámiť na `http://www.uni-koblenz.de/ag-ki/LP`.

Najprv sme si zrekapitulovali, čo všetko by mal vedieť „perfektný“ výpočtový model inteligencie. V tej súvislosti sme stručne spomenuli výskumy, ktorých cieľom je špecifikácia služieb, schopností a funkcií, ktoré má poskytovať výpočtový model inteligencie. Potom sme si všimli problémy metausudzovania a autoreflexie. Od nich sme prešli k metódam a konceptom smerujúcim k praktickej realizovateľnosti výpočtových modelov inteligencie. Najprv sme si všimli niektoré techniky, ktoré využívajú metajazykový rámec na zefektívnenie univerzálnych výpočtových metód. Tieto techniky spočívajú v špecializácii všeobecného mechanizmu na práve riešenú inštanciu problému. Ďalej sme sa venovali tradičným reprezentačným schémam, ktoré sa považovali za prostriedky efektívnej implementácie znalostných systémov. Ukázali sme, že otázka ich efektívnosti je komplikovanejšia ako sa pôvodne predpokladalo. Potom sme analyzovali, ako sa dá vyhnúť zbytočnej inferencii (ten problém vlastne postavili – na naivnej úrovni – reprezentačné schémy). Všimli sme si intenzívnu reprezentáciu, usudzovanie založené na prípadoch a testovanie modelov. Prešli sme k problémom usudzovania pri obmedzených zdrojoch. Spomenuli sme si na prístupy, ktoré už poznáme (obmedzenie jazyka alebo odvodzovacej mašinérie). Potom sme si všimli kompiláciu reprezentácií do nejakej aproximácie pôvodnej reprezentácie. Nasledovala základná informácia o flexibilných algoritmoch. Ide o algoritmy, ktoré môžu vyvažovať čas, ktorý majú k dispozícii a kvalitu riešenia.

Inteligencia ako výpočet?

Dostali sme sa k bodke (vlastne k otázniku) za jadrom textu. Naším cieľom bolo oboznámiť sa s pokusmi výpočtovo modelovať inteligenciu. Prezentovaný prístup používa na toto modelovanie „výpočty na vetách“ – explicitnú reprezentáciu poznatkov nejakými vetami nejakého jazyka a odovodzovanie dôsledkov týchto viet. Akceptovali sme, že pre pochopenie inteligencie (zdravého rozumu, common sensu) je dôležité pochopiť hypotetické usudzovanie: schopnosť spracúvať neúplné a nekonzistentné poznanie, generovať hypotézy, revidovať ich. Oboznámili sme sa s viacerými formalizáciami hypotetického usudzovania pomocou výpočtov na vetách. Predstavili sme si niektoré možnosti sémantickej špecifikácie takýchto výpočtov. Nahliadli sme, že tieto výpočty sú zložité, ťažké.

Je nesporné, že miera detailnosti, potrebná pre výpočtové modelovanie, viedla k výraznému posunu[6] chápania nemonotónneho, hypotetického usudzovania i revízií. Otázku o sile tohto prístupu zodpovie až budúcnosť. Odpoveď však bude založená aj na výsledkoch, s ktorými sa zoznámil čitateľ tejto knihy.

[6] Výtvarné dielo P. Bartoša, použité na obálke tejto knihy sa nazýva *Nič. Bod. Posun.*

Dodatok A

Prípadová štúdia

Systému Artificial Mathematician (AM) venujeme pozornosť preto, aby sme na konkrétnom prípade motivovali východiská prístupu prijatého v tejto práci (boli sformulované v kapitole 1).

Programový systém *AM* [Len 76] vzbudil veľký záujem a rešpekt na prelome sedemdesiatych a osemdesiatych rokov. Jeho cieľom bolo „objavovať", navrhovať matematické pojmy a hypotézy (nie dokazovať tvrdenia). Matematiku ponímal ako činnosť, ako inteligentné správanie, nie ako ukončený produkt, množinu akceptovaných definícií a dokázaných, platných viet.

Objavovanie matematických pojmov a hypotéz je bezpochyby inteligentná činnosť. Iste je zaujímavé pýtať sa, či je takýto typ činnosti výsledkom správania *AM* a či správanie tohto systému možno charakterizovať vlastnosťami z konvencie 1.2.

Podrobnú analýzu *AM* urobili Ritchie a Hanna v [RHa 84], v článku podstatne presahujúcom rámec analýzy jedného programového systému. Svojím zacielením sa týka prevládajúcich metód výskumu v umelej inteligencii.[1]

Skôr než prejdeme k vlastnému *AM*, stručná zmienka o analógii medzi programovým systémom typu *AM* a rozmýšľajúcim človekom. Táto analógia je vlastne kdesi v pozadí za návrhom a realizáciou výpočtového modelu, nazvaného *AM*.

Rozmýšľajúci človek

- disponuje nejakými vedomosťami a pojmami,
- je schopný využívať ich (inými slovami, uskutočňovať nejaké mentálne akcie na týchto pojmoch a vedomostiach),
- spravidla sú tieto akcie nejako riadené – nejakým cieľom, úlohou, pred ktorou človek stojí v jeho prostredí.

[1] Treba podotknúť, že analýza pochádza z počiatku osemdesiatych rokov a reflektovala vtedajší stav. To však neznamená, že dnes nemá čo povedať.

Podobnú rolu plnia v programovom systéme (vo výpočtovom modeli):

- údajové štruktúry (v *AM* sú nimi koncepty),
- procedúry, pravidlá,
- riadiaca štruktúra.

Programový systém, ktorého ambíciou je prínos do umelej inteligencie, by mal byť založený na akejsi „teórii“, ktorá vysvetľuje modelované správanie. Napríklad programový systém, ktorého cieľom je imitovať objavovanie matematických pojmov a hypotéz, môže vychádzať z teórie, podľa ktorej toto objavovanie je možné vďaka použitým údajovým štruktúram, procedúram a pravidlám, vďaka riadiacej štruktúre, ktorá určuje, ako a kedy s nimi narábať.[2]

Počiatočné poznatky *AM* pozostávali z:

- 115 pojmov teórie konečných množín,
- približne 250 heuristických pravidiel (návodov na objavovanie).

Počas svojho behu *AM* okrem iného vytváral nové pojmy a hypotézy s nimi späté. Teda možno povedať, že jeho matematické poznanie sa rozširovalo. Podľa Lenata jeho program pokročil od prednumerického poznania k znalostiam negraduovaného študenta matematiky.

AM rýchlo prišiel na väčšinu obvyklých množinových pojmov a vzťahov, nedostal sa však nikdy k sofistikovanejšej teórii množín. Objavil prirodzené čísla a aritmetické operácie (ako analógie k množinovým operáciám), rozvíjal elementárnu teóriu čísel, rýchlo pokročil k teórii deliteľnosti. Objavil prvočísla. Za výrazné medzníky v jeho činnosti možno považovať práve objavy prirodzených čísel a prvočísel. Vyslovil hypotézu o jednoznačnom rozklade na prvočinitele a Goldbachovu hypotézu.

Toľko o správaní *AM*, o jeho výsledkoch.

Ako vyzeralo „vnútro“ *AM* ? Na jeho charakterizáciu postačuje upresniť, ako vyzerali jeho pojmy, pravidlá a riadiaca štruktúra. Súčasne s upresnením budeme používať aj Lenatove termíny koncept, heuristika a agenda.

Základnou a jedinou údajovou štruktúrou boli *koncepty*.

Príklad A.1 Na obrázku A.1 vidíme príklad konceptu z [DLe 82]. ∎

V *AM* boli koncepty implementované ako zoznamy dvojíc atribút - hodnota. Hodnotou mohol byť ľubovoľne zložito štrukturovaný zoznam. Atribúty

[2]Požiadavka, aby vnútorná štruktúra výpočtového modelu zodpovedala psychike človeka – v takomto prípade môžeme hovoriť o kognitívnom modeli – sa často formuluje ako jedna z ambícií časti výskumníkov v umelej inteligencii. Značná časť si však úlohu nestavia tak ostro.

```
NAME:   LIST-EQUAL
IS-A:   (PREDICATE FUNCTION OP BINARY-PREDICATE
        BINARY-FUNCTION BINARY-OP ANY-THING)
GENL:   (SET-EQUAL BAG-EQUAL OSET-EQUAL STRUC-EQUAL)
SPEC:   (LIST-OF-EQ-ENTRIES LIST-OF-ATOMS-EQUAL EQ)
FAST-ALG: (LAMBDA (x y) (EQUAL x y))
RECUR-ALG:(LAMBDA (x y)
            (COND  ((OR (ATOM x) (ATOM y)) (EQ x y))
                   (T (AND
                        (LIST-EQUAL (CAR x) (CAR y))
                        (LIST-EQUAL (CDR x) (CDR y))
                      )
              )
        )
DOMAIN: (LIST LIST)
RANGE:  TRUTH-VALUE
WORTH:  720
```

...

Obrázok A.1: Príklad konceptu

(výrazy, ktoré sa v každom riadku nášho príkladu nachádzajú pred dvojbodkou) budeme nazývať *aspekty* (v *AM facets*). Bol definovaný pevný počet asi[3] 20 aspektov, ktoré charakterizujú každý z konceptov. Mnohé z aspektov boli vo väčšine konceptov bez hodnoty.

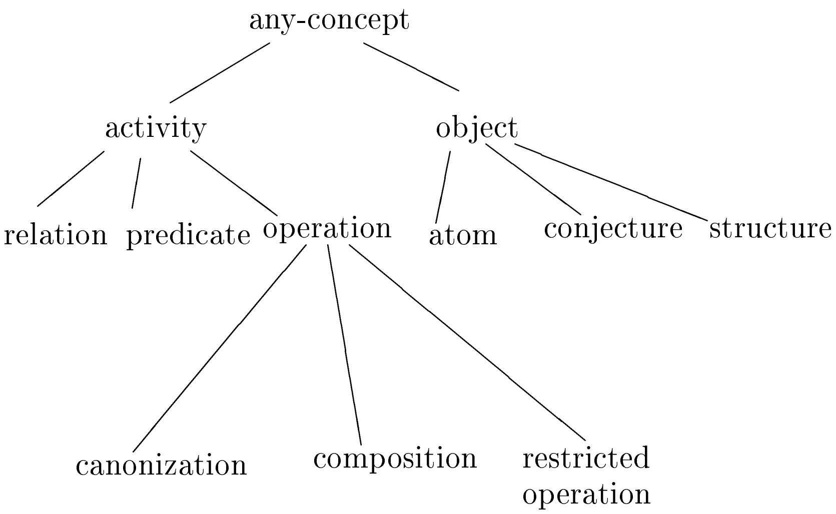
Aspekty SPEC a GENL ukazujú, že koncepty tvoria hierarchiu. Na jej vrchu sú najvšeobecnejšie koncepty (ANY-THING, ANY-CONCEPT). Fragment Lenatovej hierarchie konceptov ilustruje obrázok A.2.

Ďalšie dôležité aspekty obsahujú meno konceptu, jeho definíciu alebo viaceré definície (v uvedenom príklade sú to FAST-ALG a RECUR-ALG); definičný obor a obor hodnôt (DOMAIN, RANGE); hodnotu, ktorá vyjadruje, nakoľko je koncept zaujímavý (WORTH).

Popri konceptoch boli dôležitým komponentom systému *heuristiky*. Sú to pravidlá typu „ak PODMIENKA, tak AKCIA“, pričom PODMIENKA býva konjunkcia viacerých elementárnych podmienok. Implementované boli ako LISPovské funkcie. Heuristické pravidlo vedie k uskutočneniu nejakej akcie (operácie), keď je splnená jeho podmienka. Jeho zmyslom je navádzať systém k akciám, ktoré môžu byť užitočné, plodné pri objavovaní. Heuristiky sú orientované lokálne v tom zmysle, že vyjadrujú poznanie relevantné pre určitý koncept (alebo s ním súvisiace koncepty) a jeho aspekt.

Riadiacu štruktúru systému *AM* možno popísať takto: Činnosť systému

[3] „Asi“ preto, lebo počet aspektov sa menil počas implementácie, pôvodná predstava rozlišovať aspekty a podaspekty sa v záverečných štádiách implementácie *AM* nerealizovala a písomné prezentácie [Len 76, DLe 82] to nie vždy presne zaznamenali. Pozri diskusiu [RHa 84, LeB 84].



Obrázok A.2: Príklad hierarchie konceptov z [DLe 82].

AM riadi plánovač s využitím agendy. *Agenda* obsahuje úlohy, zoradené podľa ich priority. Plánovač vyberá z agendy vždy úlohu s najvyššou prioritou (užívateľ má možnosť výber ovplyvniť v dialógu). *Úloha* je tejto formy: „Vykonaj akciu A na aspekte F konceptu C.“

Teraz sa môžeme venovať vzťahom konceptov, heuristík a riadiacej štruktúry.

Heuristické pravidlá sú uložené priamo v údajovej štruktúre koncept. Sú priradené k aspektom, na ktoré sú aplikovateľné ich akcie.[4] Vidno, že ide o akýsi indexovací mechanizmus. (Pod *indexom* rozumieme ľubovoľnú funkciu definovanú na nejakej množine, ktorá jej prvkom priraďuje množiny viet nejakého jazyka. V našom prípade k aspektu sú priradené tie heuristiky, ktoré možno naň aplikovať.)[5]

Lenat [DLe 82] rozlišuje tri vedľajšie efekty (možné typy akcií) heuristík :

- navrhnúť novú úlohu (do agendy),
- vytvoriť nový koncept,
- vložiť hodnotu do nejakého aspektu nejakého konceptu.

Akciu A, určenú úlohou „vykonaj akciu A na aspekte F konceptu C“, umožňuje vykonať vhodné heuristické pravidlo (typu „ak POMIENKA, tak

[4] Pôvodne mali byť uložené v podaspektoch pomenovaných podľa typu úlohy. Pravidlá, uložené v určitom podaspekte, mali realizovať úlohu na zodpovedajúcom nadradenom aspekte.

[5] Jazyk a jeho vety budeme chápať čo najvšeobecnejšie, aspoň tak, aby vetami boli aj heuristiky (teda kusy LISPovského kódu). Presnejšie vymedzenie vety nám v konkrétnom jazyku nespôsobuje problémy.

AKCIA“). Musí byť splnená jeho podmienková časť a jeho akčnou časťou je práve A.

Pravidlá možno dediť. Ak je nejaké pravidlo priradené k aspektu X konceptu C, možno ho aplikovať na aspekt X všetkých špecializácií tohto konceptu.[6]

Ak sa nájdu viaceré pravidlá, ktoré umožnia vykonať vybratú úlohu, aplikujú sa všetky, pokiaľ neboli prekročené časové a priestorové kapacity, pridelené danej úlohe. (Každá úloha má deskriptor, ktorý obsahuje okrem ďalších charakteristík aj časové a priestorové kapacity, ktorými úloha môže disponovať).

A.1 Analýza systému

Ritchie a Hanna dôkladne analyzovali *AM* a výsledky analýzy využili na pokus o všeobecnú charakterizáciu spôsobu výskumu a prezentovania jeho výsledkov v umelej inteligencii. Treba podotknúť, že ich prácu umožnila neobvykle podrobná (v porovnaní s bežným štandardom) Lenatova dokumentácia z [Len 76, DLe 82]. V tejto časti priblížim, prečo sa Ritchie s Hannom vôbec venovali analýze *AM* a čo z nej vyvodili. Výklad dosť tesne sleduje ich chod úvah.

Za bežný prístup[7] k výskumu v oblasti umelej inteligencie považujú, že bádateľ (resp. ašpirant na bádateľskú pozíciu) napíše rozsiahly, zložitý program, vyprodukuje niekoľko (niekedy jeden alebo dva) pôsobivých výstupov a potom napíše články, v ktorých tvrdí, že urobil to, čo urobil. Tento prístup je teoreticky sterilný, nerozvíja princípy, neobjasňuje a nedefinuje skutočné výskumné problémy. V terminológii, ktorú som zaviedol v kapitole 1: neobjasňuje použitý výpočtový model, zásady jeho konštrukcie.

Iný typický prístup venuje istú pozornosť načrtnutiu princípov, ktoré má program implementovať. Rozsiahly a zložitý program s pôsobivým správaním je založený na prezentovanej idei (teórii). V niektorých prípadoch však „teória“ nezodpovedá úplne programu, bádateľ sa vyhýba analýze tejto nezrovnalosti, niekedy ju dokonca zatajuje.

Tento stav navrhovania a prezentovania rozsiahlych programových celkov, najmä však ich teoretického zdôvodňovania (fundovania), sa negatívne prejavuje pri pokusoch nadväzovať na dosiahnuté výsledky. Nedostatky v špecifikácii detailov základnej myšlienky spôsobujú, že ten, kto chce nadväzovať na ňu, sa musí vracať do východiskovej situácie (v ktorej je známe o málo viac

[6] Vzhľadom na dôležitosť pojmu indexu ako súčasti reprezentácie znalostí si zhrnieme, ako sú indexy využívané v AM. Po prvé, indexy slúžia na reprezenáciu hierarchických vzťahov medzi koceptami (čo podporuje aj zefektívnenie výpočtov, napríklad pri dedení vlastností všeobecnejších konceptov). Po druhé, slúžia na „zhlukovanie“ informácií – jeden koncept je vlastne zhluk súvisiacich informácií. Po tretie, niektoré zhluky prepájajú rôzne druhy znalostí – heuristiky sú pripojené k aspektom konceptov.

[7] Opäť pripomínam, že článok priamo reflektuje stav v prvej polovici osemdesiatych rokov.

ako základná intuitívna myšlienka). „Teória“, ktorú vyvinul jeho predchodca, mu poskytuje nedostatočné usmernenie, pretože charakterizuje správanie výpočtového modelu (dosiahnuté výsledky), ale nie spôsob, ako ho dosiahnuť (nie „vnútro“ výpočtového modelu). Prirodzeným dôsledkom tohto stavu je, že prezentácie dosiahnutých výsledkov sú nekonzistentné alebo nekorektné.

Na príklade Lenatovho *AM* Ritchie a Hanna detailne ilustrujú uvedené všeobecné tézy. Treba zdôrazniť, že akceptujú skutočnosť, že *AM* je schopný dosahovať publikované výstupy, tvrdia „iba“, že popisy systému poskytujú zavádzajúci pohľad na to, ako program pracuje. Tieto nezrovnalosti medzi publikovanými tvrdeniami a implementovaným programom zatemňujú skutočný prínos *AM* a môžu zmiasť tých, ktorí sa usilujú nadviazať na dosiahnuté výsledky.

Ritchie a Hanna si položili pri hodnotení *AM* tieto základné otázky:

1. Zhoduje sa skutočne použitá riadiaca štruktúra systému s Lenatovým popisom?[8]

2. Obsahovali heuristické pravidlá všetko poznanie systému, potrebné pre výber, usporiadanie a vykonávanie úloh? Nebolo toto poznanie uložené aj v iných komponentoch programu *AM*?

3. Čo v skutočnosti *AM* objavil?

Ich odpovede na tieto otázky sa teraz pokúsim stručne vyložiť.

Riadiaca štruktúra

Základná téza o *AM* tvrdí, že použitá jednoduchá riadiaca štruktúra zaručuje jeho pôsobivé správanie a výstupy. Hanna s Ritchiem však na základe skúmania Lenatovej práce tvrdia, že *AM* v skutočnosti nefunguje popísaným spôsobom.

Zopakujme si, že riadiaca štruktúra *AM* je v [DLe 82] opísaná takto: Plánovač vyberá (za asistencie užívateľa) tú úlohu z agendy, ktorá má najvyššiu prioritu. Typická úloha má tvar „Vykonaj akciu A na aspekte F konceptu C“. Túto úlohu sa systém pokúša splniť pomocou heuristík, ktoré sú priradené k aspektu F konceptu C. Bez ohľadu na to, či systém úlohu splnil alebo sa mu ju nepodarilo splniť, pokračuje výberom ďalšej úlohy z agendy (samozrejme, výsledok poslednej úlohy ovplyvňuje priority nasledujúcich úloh).

Podrobné štúdium *AM*, predovšetkým však štúdium zverejnených heuristík, viedlo Ritchieho a Hannu k záveru, že skutočne použitá riadiaca štruktúra sa nezhoduje s opísanou, teda predložená špecifikácia nevysvetľuje správanie vytvoreného výpočtového modelu.

[8] Zdôrazňujú, že im nejde o implementačné detaily, ale o základnú otázku: v Lenatovej prezentácii sa organizácia úloh, konceptov a heuristík vyzdvihuje ako *vysvetlenie* výberu odvodzovacích krokov *AM* v každej situácii.

Opomenieme nejasnosti okolo organizácie aspektov a podaspektov, ktorej sa Ritchie s Hannom obšírne venujú (tohto problému sme sa už dotkli, dá sa vysvetliť meniacimi sa predstavami o organizácii konceptov; prijmime, že nejaká pevná organizácia sa dá vytvoriť a v nej si systém vždy nájde potrebnú heuristiku).

Pravdepodobne vážnejšie je, že niektoré úlohy nezapadajú do schémy, podľa ktorej by mali vykonať akciu A na aspekte F konceptu C. Je pravda, že Lenat túto schému považuje iba za typickú schému úlohy, nie jedinú. Na druhej strane však nepojednáva o iných typoch úloh a nevysvetľuje ich prípadnú rolu vo fungovaní systému a v jeho riadiacej štruktúre. Podobne, niektoré akcie, iniciovateľné heuristikami, sa netýkajú aspektu nejakého konceptu, ale celého konceptu bez toho, že by sa podrobnejšie vyjasnilo ich fungovanie. Niektoré heuristiky majú byť aplikovateľné na ľubovoľný objekt X, bez ohľadu na to, či to je koncept, atóm, udalosť atď. Podľa odhadu Ritchieho a Hannu sa zdá, že tieto „heuristiky“ nie sú skutočnými heuristikami, ktorých text možno odlíšiť od zvyšku systému, sú iba verbálnymi generalizáciami, sumarizujúcimi výsledok rôznych častí celého LISPovského programu vytvárajúceho AM.

Niektoré z popísaných heuristík sú formulované tak, že vznikajú nejasnosti o podmienkach ich aktivácie. Najdôležitejší argument však poskytol sám Lenat, ktorý napísal , že „nie všetky heuristiky sú kódované ako samostatné LISPovské entity ... Skôr sú vplietané do väčších kusov programu“ [Len 76], s.140.

Charakteristické je aj to, ako Lenat s Brownom v [LeB 84] odpovedajú na výhradu o riadiacej štruktúre: „Zdá sa, že ... ponímajú Lenatovu dizertáciu ako formálny dôkaz teorémy, v ktorom nájdenie najmenšej nekonzistentnosti alebo odchýlky od čistej riadiacej štruktúry túto štruktúru vyvracia ... *AM* je demonštráciou toho, že na adekvátne vedenie 'bádateľa' pri formovaní teórie v elementárnej matematike treba o málo viac než heuristiky.“

Porovnanie s formálnym dôkazom teorémy nie je namieste, nejde o nič viac, ale ani o nič menej, ako o jasnú špecifikáciu riadiacej štruktúry. Bez nej sa rozhodujúcim spôsobom komplikuje nezávislé preverovanie výsledkov výskumu v *AM*, či prípadné nadväzovanie na ne. (Napríklad, toto komplikovanie iste súvisí aj s hádaním, čo sa skrýva za formuláciou „o málo viac“). Jedným z možných vysvetlení odchýlok od deklarovanej riadiacej štruktúry môže byť ad hoc dolaďovanie programu s cieľom dosiahnuť požadované správanie (pričom uskutočnené zmeny nie sú zachytené v nijakom opise výpočtového modelu a sú ťažko charakterizovateľné z tela komplikovaného programu).

Poznatky

Ritchie s Hannom ďalej ukázali, že hierarchia konceptov a na ich aspekty napojených heuristík nie je hlavným a jediným zdrojom poznatkov využívaných pri objavovaní. Podľa nich veľa netriviálnej práce odvádzajú procedúry, ktoré sú volané z heuristík. Tvrdia, že miera detailnosti opisu nie je postačujúca, kľúčové časti procesu objavovania ostávajú zakryté. Spomedzi príkladov, ktoré uvádzajú, si zvláštnu pozornosť zasluhujú Lenatove odvolávky na bližšie nešpecifikované triky, špeciálne cielené procedúry a „hacks“ [Len 76].

Objavy

Pokiaľ ide o samotné objavy *AM*, Ritchie a Hanna vyslovili názor, že je možné nadobudnúť dojem, že úspešný „objav“ bol výsledkom špeciálne navrhnutých kusov informácie, zacielených na jeho dosiahnutie.

Kladú si teda otázky ako: Neobsahuje systém ďalšie dodatočné mechanizmy spracovania s významnými účinkami? Aké množstvo dosiahnutých objavov je založené na nevysvetlených procedúrach? Možno preveriť, že matematické objavy, ktoré systém dosiahol a ktoré sa pokladajú za dôležitý aspekt jeho práce, rezultovali priamo z opísanej prehľadávacej procedúry?

A.2 Závery

Teraz možno zhrnúť metodologické dôsledky, ktoré Ritchie a Hanna vyvodili z analýzy *AM*.

Konštatovali, že vážnou negatívnou črtou umelej inteligencie (na rozhraní sedemdesiatych a osemdesiatych rokov) je štýl výskumu. Ten komplikuje až znemožňuje nadväzovanie na prezentované výsledky. Vytvára tak prekážky pre postupný rozvoj teoretického rámca i dosiahnutých implementačných výsledkov.

Konštatovali to nielen na základe nejasností okolo kľúčových aspektov (riadiaca štruktúra, dátové štruktúry, skryté a neobjasnené mechanizmy, závislosť dosiahutých výsledkov na neobjasnených mechanizmoch) jedného z najprestížnejších systémov umelej inteligencie tých čias, *AM*.

Hanna s Ritchiem sformulovali niektoré zásady pre výskum v umelej inteligencii (sami zdôrazňujú, že ide o realistické zásady, liberálnejšie a miernejšie, než sú striktné zásady práce v empirickej vede).

Výskum v umelej inteligencii môže :

1. zaviesť novú, ešte nedostatočne rozpracovanú, myšlienku, neformálny návrh, nedostatočne presný na to, aby ho bolo možné empiricky vyvrátiť (často v tvare sloganu alebo metafory),

2. rozpracovať detaily nejakého prístupu (výskum štartuje z myšlienky charakterizovanej vyššie, môže ju kritizovať, doplniť ďalšie detaily tak, aby transformoval pôvodnú myšlienku do jemnejšej a úplnejšej teórie),

3. aplikovať nejakú vyslovenú teóriu na konkrétnu oblasť alebo údaje, podať správu o výsledkoch (toto je najbližšie k tradičnému experimentu, líši sa však v niektorých dôležitých aspektoch: v typickom prípade

skúmanie prebieha vo forme písania a zbiehania počítačového programu; hodnotenie výsledkov takejto aktivity je ťažké, posúdenie v polohe úspech/zlyhanie nie je možné).

Ritchiemu a Hannovi sa zdá byť podivné, že 5 rokov po zverejnení správy o *AM* (systém bol veľmi vysoko hodnotený v odborných kruhoch) sa neobjavila nijaká práca, ktorá by preverovala dosiahnuté výsledky alebo nadväzovala na ne.

Tento nedostatok považovali za typický pre umelú inteligenciu, nezlučiteľný s praktikami tradičnej vedy, ktorá predpokladá, umožňuje a požaduje (opakované) preverovanie dosiahnutých výsledkov.

Ritchie a Hanna ako pozitívny prínos projektu *AM* hodnotia nápad považovať matematické objavovanie za systematickú formu odvodzovania, založenú na zvláštnom druhu štruktúry poznatkov a na priradení heuristík k relevantným konceptom.

Tvrdia však, že *AM* nepreveril podrobne tieto myšlienky, lebo program *AM* sa od pôvodných návrhov líši.[9] Podľa nich negatívnym výsledkom projektu bolo, že sa vytvoril dojem, že takýto test sa uskutočnil, a to s pôsobivými výsledkami. Takto sa zatemňuje reálna sila navrhnutej schémy (hypotézy, teórie) a sťažuje to situáciu každému, kto by chcel v tomto výskume systematicky pokračovať.

Na tomto mieste môžeme spomenúť nedávny pokus reimplementovať *AM* [Gaš 97]. „Objaviteľské výsledky“ tejto reimplementácie (systém *rAM*) boli podstatne skromnejšie a autor *rAM* nenašiel v Lenatových popisoch dostatočné informácie na to, aby mohol vysvetliť túto odlišnosť správania.

Napríklad, *rAM* neobjavil ani prirodzené čísla. Teno fakt Gašo analyzuje (pozri [Gaš 97], str. 38) a výsledky jeho analýzy vlastne potvrdzujú, že skeptický postoj Ritchieho a Hannu k ohlasovanému objaveniu prirodzených čísel systémom *AM* nebol neodôvodnený.[10]

Uzavrime rozbor *AM*:

- Našiel *AM* vlastnosti výpočtového modelu, potrebné na objavovanie matematických pojmov a hypotéz?

- Vytvoril platformu pre postupné zdokonaľovanie použitého výpočtového modelu?

[9] Rozdiely medzi písomnou prezentáciou a zrealizovanými programami hodnotia až tak vážne, že pre nich znamenajú potvrdenie podozrenia, že neexistuje teoretická báza, na základe ktorej by sa dalo ďalej pracovať.

[10] „How does Canonise come into the computation? This is extremely obscure.“ Rule 206 contains a „special hand-crafted piece of knowledge“ to tell *AM* how to canonise bags into bags-of-T's. Rule 207 is a special purpose rule ... used to guide a series of experiments. None of the sample traces show exactly what rules are used at what points, so we can only speculate about the details of these runs. However, it is possible to gain the impression that the successful „discovery“ was the result of various specially designed pieces of information.

Odpoveď na druhú otázku je negatívna. K pozitívnej odpovedi na prvú otázku chýba presvedčivá demonštrácia toho, že program AM skutočne zrealizoval sformulovanú teóriu, používajúcu terminológiu konceptov, heuristík a agendy.

Zdá sa, že jadro problému je vo vysokej miere zastretosti používaných myšlienok, konštrukcií a dôležitých detailov vo výpočtových modeloch.

Dodatok B

Logika

Budeme definovať syntax (jazyk, dôkaz) a sémantiku (interpretácia, model) predikátovej logiky prvého rádu.

Predpokladáme jazyk s pevne definovanou **abecedou**, zoznamom symbolov. Rozlišujeme štyri typy týchto symbolov. Prv než prejdeme k ich vymenovaniu, základné intuície. Predikátové symboly sú mená vlastností alebo vzťahov (byť prvočíslom, byť medzi atď.); funkčné symboly sú mená funkcií, možno ich použiť na konštruovanie označení objektov (2+3 označuje 5):

(i) **predikátové symboly** (obvykle označované ako p, q, r), každý predikátový symbol má jednoznačne určenú aritu – počet argumentov; špeciálnym prípadom sú **výrokové premenné**, ktorých arita je 0;[1]

(ii) **funkčné symboly** (obvykle f, g), tiež s definovanou aritou; ich špeciálnym prípadom sú konštanty – funkčné symboly, ktorých arita je 0;

(iii) **premenné** (obvykle X, Y, Z);[2]

(iv) **logické symboly** (kvantifikátor $\exists$; spojky $\wedge$, $\neg$; identita $=$).

Intuitívne: $\exists X p(X)$[3] znamená, že existuje objekt, ktorý má vlastnosť p; ak p a q sú výrokové premenné, potom $p \wedge q$[4] znamená, že platí súčasne p i q a $\neg p$[5] znamená, že p neplatí; presnejšie nižšie pri pojednávaní sémantiky.

[1] Ak arita predikátového symbolu p je 2, potom $p(X, Y)$ je správne utvorený výraz (X, Y sú argumenty), ale $p(X)$ nie, ak jeho arita je 0, správne utvorený výraz je p.

[2] Jemným, ale nepodstatným zovšeobecnením tohto jazyka je jazyk viacsortovej predikátovej logiky: odlišujeme rôzne typy konštánt, pre každý z týchto typov zavádzame špeciálny typ premenných, pre funkčné symboly sa definuje, aké typy tvoria jeho definičný obor a akého typu sú jeho hodnoty. S aplikáciou viacsortovej logiky sme sa stretli v kapitole 2. Viacsortovú logiku však môžeme jednoducho redukovať na jednosortovú.

[3] Kvantifikátor $\exists$ sa nazýva existenčný.

[4] Formulu tohto tvaru nazývame konjunkciou, operandy p, q, konjunktami.

[5] Formulu tohto tvaru nazývame negáciou.

Ďalšie logické symboly sa dajú definovať štandardným spôsobom: $\forall X p(X)$[6] ako $\neg\exists X \neg p(X)$, $p \vee q$[7] ako $\neg(\neg p \wedge \neg q)$, $p \rightarrow q$[8] ako $\neg(p \wedge \neg q)$. V texte sa často využíva fakt, že na definovanie ostatných spojok stačia dve uvedené (alebo iná dvojica, napríklad $\neg$ a $\vee$, prípadne $\neg$ a $\rightarrow$) a v definícii jazyka sa zavedie iba dvojica spojok, kým v ďalšom výklade sa bez explicitného definovania pracuje aj s niektorou z ďalších spojok.

Logiku, ktorej jazyk obsahuje iba výrokové premenné a spojky, nazývame **výrokovou logikou**. **Predikátová logika prvého rádu** používa jazyk so všetkými štyrmi uvedenými druhmi symbolov.

Symboly typu (i) a (ii) sa nazývajú mimologickými symbolmi.[9]

Term definujeme (rekurzívne) takto:

(i) premenná alebo konštanta,

(ii) ak $t_1, \ldots, t_n$ sú termy a f je funkčný symbol arity n, potom $f(t_1, \ldots, t_n)$ je term.

Základný term neobsahuje žiadne premenné.

Atóm je výraz tvaru $p(t_1, \ldots, t_n)$, skonštruovaný z predikátového symbolu p a termov $t_1, \ldots, t_n$. Arita predikátu p je n. (Špeciálnym prípadom atómu je $u = v$, kde u, v sú termy a identita je predikátom.)[10] Arita predikátu je pevne určená a počet jeho argumentov, termov, musí byť v zhode so špecifikovanou aritou.

Základný atóm neobsahuje žiadnu premennú. **Literál** je atóm alebo negácia atómu. **Základný literál** neobsahuje žiadnu premennú.

Rekurzívna definícia **formuly**:

(i) Každý atóm je formula.

(ii) Ak F, G sú formuly a X premenná, potom aj $\exists X F$, $F \wedge G$, $\neg F$ sú formuly.

Dôkladná definícia voľnej a viazanej premennej vyžaduje zavedenie takých formálnych detailov, ktoré nie sú pre naše ciele potrebné. Preto iba základné intuície. Ak sa premenná X vyskytuje vo formule F, potom jej výskyt je vo formule $\exists X F$ **viazaný**. V uvedenom príklade je formula F dosahom kvantifikátora $\exists X$. Dosah kvantifikátora nebudeme definovať, pokladáme ho za dostatočne jasný. Niekedy je dôležité zdôrazniť ho zátvorkovaním – v $\exists X F \wedge G$ je dosahom kvantifikátora formula F a v $\exists X(F \wedge G)$ formula $F \wedge G$. Potom

[6] Pre každé X platí, že má vlastnosť p. Všeobecný kvantifikátor.

[7] p alebo q, disjunkcia, operandy sú disjunkty, prípadne alternatívy.

[8] Ak p, tak q, implikácia. p je antecedent, q je konzekvent. Tú istú implikáciu niekedy píšeme aj ako $q \leftarrow p$ (v súlade s konvenciou, používanou v logickom programovaní: ak chcem vypočítať q, najprv musím vypočítať p).

[9] Zoberme si príklad logickej formalizácie databáz z kapitoly 2. Tam predikátové symboly charakterizujú to, čo je pre danú databázu špecifické (funkčné symboly sme na formalizáciu databáz nepotrebovali).

[10] Miesto obvyklého zápisu $u = v$ možno použiť i zápis $= (u, v)$, ak chceme dôsledne zachovať uvedenú syntax. My sa však podobným formalitám budeme vyhýbať.

voľný výskyt premennej vo formule F je taký jej výskyt, ktorý nie je v dosahu žiadneho kvantifikátora. Premenná X je vo formule F voľná (viazaná), ak každý jej výskyt v F je voľný (viazaný).

Uzavretá formula je taká formula, ktorá neobsahuje nijaké voľné výskyty premenných. Otvorená formula obsahuje aspoň jeden voľný výskyt nejakej premennej.

Nech F je formula. Jej všeobecný (existenčný) uzáver $\forall(F)$ $(\exists(F))$ získame z F tak, že všeobecným (existenčným) kvantifikátorom viažeme všetky voľné premenné z F.

Nejakú množinu (uzavretých) formúl jazyka môžeme zvoliť za **axiómy** (nejakej teórie). Pod teóriou rozumieme množinu axióm spolu s množinou ich dôsledkov. **Odvodzovacie pravidlá** určujú, čo možno odvodiť z axióm. Sú to dvojice (Pre, Con), kde Pre a Con sú množiny formúl. Pre nás bude Con vždy jednoprvková množina (jediná formula). Intuitívne, Pre sú predpoklady odvodenia a Con záver (konzekvencia).[11]

Odvodzovať možno z ľubovoľnej množiny formúl. Odvodzovanie z axióm sa nazýva dôkazom. Budeme predpokladať, že je daná nejaká množina formúl predikátovej logiky prvého rádu, ktoré sú jej axiómami. Nebudeme však čitateľa zaťažovať nejakou konkrétnou axiomatizáciou.

Dôkaz formuly F je postupnosť formúl $D = \langle D_0, \ldots, D_k \rangle$, kde $D_k = F$, D_0 je axióma a pre každé i, $0 < i \leq k$, buď D_i je axióma alebo existuje odvodzovacie pravidlo (Pre, Con) také, že $Con = \{D_i\}$ a každý prvok z Pre je buď axióma alebo sa vyskytuje v D na miestach predchádzajúcich i. Ak existuje dôkaz formuly ϕ z axióm predikátovej logiky prvého rádu, hovoríme, že ϕ je **teoréma** predikátovej logiky. Nech M je konečná množina formúl a M' je konjunkcia všetkých formúl z M. Ak $M' \rightarrow \psi$ je teoréma, potom ψ je odvoditeľné z M.

Keď sme si uvádzali intuície o logických symboloch, dosť nešikovne sme tajili, že sa odvolávame na nejaký mimojazykový svet. Teraz to odtajníme. (A začneme sa venovať sémantike.)

Interpretácia[12] jazyka J je dvojica $(\mathcal{I}, U)$, kde:

- U je **univerzum** (doména), nejaká množina[13] (objektov),

---

[11] Tento pojem odvodenia bol v jadre textu zavedený v definícii 3.3 z kapitoly 3. Všeobecnejší pojem odvodenia z tej istej kapitoly presahuje referenčný rámec predikátovej logiky prvého rádu.

[12] Zavádzame štandardný – v kontexte logiky – pojem interpretácie. V texte (od kapitoly 3) je zavedený a používaný modifikovaný pojem. Univerzum U v jadre vlastného textu zväčša nepotrebujeme zvlášť určovať, lebo používame Hebrandovské interpretácie (kde univerzum tvoria symboly jazyka).

[13] Ak používame viacsortovú logiku, miesto univerza U prijímame domény $U_1, \ldots, U_k$, každú pre jednu sortu. Bez viacsortovej logiky sa však môžeme zaobísť. Predstavme si, že univerzum U sa skladá z objektov dvoch sort U_1 a U_2, pričom $U = U_1 \cup U_2$ a $U_1 \cap U_2 = \emptyset$. Nech oblasťou premennosti premennej X je U_1 a oblasťou premennosti Y je U_2. Všimnime si atóm (viacsortového jazyka) $p(X, Y)$. V jazyku bez sort ju môžeme definovať takto: $(u_1(X) \wedge u_2(Y)) \rightarrow p(X, Y)$, kde zamýšľaný význam predikátových symbolov u_1 a u_2 je zrejmý.

- $\mathcal{I}$ je funkcia, ktorá
  - každej konštante priraďuje nejaký prvok z U,
  - každému funkčnému symbolu (nenulovej arity n) funkciu z U^n do U,
  - každej výrokovej premennej jedinú hodnotu z dvojice (t, f) (pravda, nepravda); niekedy sa používa aj $(1, 0)$,
  - každému predikátovému symbolu (nenulovej arity n) podmnožinu U^n (reláciu ρ).

Využijeme, že konštantám sú priradené prvky z U. Predpokladajme, že každý term t_i, kde $i = 1, \ldots n$, je interpetovaný, t.j. $\mathcal{I}(t_i)$ je nejaký prvok z U. Potom $\mathcal{I}(f(t_1, \ldots, t_n))$ je prvok z U. Teda, máme (rekurzívne) definovanú interpretáciu termu.

Uzavretý atóm $p(t_1, \ldots, t_n)$ je **pravdivý** v interpretácii $(\mathcal{I}, U)$, ak pre $\mathcal{I}(p) = \rho \subseteq U^n$ platí $(\mathcal{I}(t_1), \ldots, \mathcal{I}(t_n)) \in \rho$ (n-tica objektov priradených príslušným termom patrí do relácie priradenej predikátovému symbolu). Inak je nepravdivá.

Ak F a G sú formuly, potom $F \wedge G$ je pravdivá práve vtedy, keď aj F aj G sú pravdivé (inak je nepravdivá). $\neg F$ je pravdivá práve vtedy, keď F je nepravdivá. $\exists X F$ je pravdivá, ak pre nejaké priradenie objektu z U premennej X je F pravdivá (v opačnom prípade je nepravdivá).

Model (uzavretej) formuly F je taká interpretácia, pri ktorej je F pravdivá. Ak S je množina (uzavretých) formúl, jej modelom je taká interpretácia, pri ktorej je každá veta z S pravdivá.

Interpretácia $\mathcal{I}$ **spĺňa** (uzavretú) formulu F, ak F je pravdivá pri $\mathcal{I}$. Formula je **splniteľná**, ak existuje interpretácia, ktorá ju spĺňa. Formula je **platná**, ak je splnená pri každej interpretácii. Podobne pre množinu formúl.

Formula ϕ **logicky vyplýva** z množiny formúl M, ak každý model množiny M je aj modelom ϕ (značenie: $M \models \phi$). Pojem logického vyplývania podáva presnú definíciu toho, čo rozumieme pod dedukciou.

Predpokladajme, že M je konečná množina formúl a M' je konjunkcia všetkých formúl z M. Potom $M \models \phi$ práve vtedy, keď teorémou predikátového počtu prvého rádu je implikácia $M' \to \phi$. Ďalšie ekvivalentné podmienky sú:

- $M' \to \phi$ je platná,
- ϕ je odvoditeľná z M,
- $M \cup \{\neg\phi\}$ je nekonzistentná.

Všetky uvedené vlastnosti sú dôsledkom Gödelovej vety o úplnosti, ktorá zakladá presnú korešpondenciu syntaxe a sémantiky predikátového počtu prvého rádu: všetko, čo je dokázateľné, je platné a naopak, všetko, čo je platné, je dokázateľné.

Výrokový počet je rozhodnuteľný – existuje algoritmus, ktorý umožňuje rozhodovať, či nejaká formula výrokovej logiky je platná (dokázateľná). Hovoríme aj, že je **tautológiou** výrokovej logiky. Predikátový počet však nie je rozhodnuteľný, analogický algoritmus neexistuje. Pozri dodatok H.

Venovali sme sa iba predikátovej logike prvého rádu. Je to logika, ktorá je schopná rozlíšiť akési objekty a ich vlastnosti alebo vzťahy medzi nimi. Nie je schopná kvantifikovať predikátové symboly (vlastnosti alebo vzťahy). Na rozlíšenie vzťahov medzi vlastnosťami alebo vlastností vlastností (a môžeme to komplikovať ďalej) nemáme dostatočné vyjadrovacie prostriedky. Predikátový počet **vyšších rádov** umožňuje zvýšiť vyjadrovaciu silu. V jeho rámci používame predikátové symboly, ktorých argumentami môžu byť iné predikátové symboly (nižšieho rádu) a dovoľujeme kvantifikáciu predikátových symbolov.

Dodatok C

Relačné databázy

V tomto dodatku si stručne uvedieme základné pojmy z teórie relačných databáz a k tomu nutné predpoklady z elementárnej teórie množín a algebry.

Relačná schéma pozostáva z

- mena relácie (tabuľky),
- zoznamu mien atribútov (stĺpcov),
- špecifikácie domény (typu, množiny možných hodnôt) – pre každý atribút.

V jazyku logiky, formula viacsortovej predikátovej logiky

$$p(X_1 : T_1, \ldots, X_k : T_k)$$

je relačná schéma, ak p je meno relácie, X_i je meno atribútu, T_i je jeho doména.

Kartézsky súčin $T_1 \times \cdots \times T_k$ množín $T_1, \ldots, T_k$ je množina všetkých k-tic $(a_1, \ldots, a_k)$ takých, že $a_i \in T_i$. **Relácia** je každá podmnožina nejakého kartézskeho súčinu. Relácia ρ je **inštanciou** relačnej schémy $p(X_1 : T_1, \ldots, X_k : T_k)$ vtedy, keď $\rho \subseteq T_1 \times \cdots \times T_k$).

Databázová schéma je množina relačných schém. Definícia relačnej databázy pozostáva z databázovej schémy a z množiny integritných obmedzení. **Integritné obmedzenie** je formula nejakého jazyka logiky (napríklad logiky prvého rádu alebo logického programovania).[1]

Množina relácií $\rho_1, \ldots, \rho_k$ je **inštanciou** databázovej schémy $S_1, \ldots, S_k$, kde S_i sú relačné schémy, ak každé ρ_i je inštanciou relačnej schémy S_i. **Model** databázovej schémy je taká jej inštancia, ktorá spĺňa všetky integritné obmedzenia.

[1] Zmysel integritných obmedzení je podrobne vysvetlený v kapitole 2. Podrobný výklad integritných obmedzení možno nájsť napríklad v [FaV 84].

Nech $\mathcal{D}$ je databázová schéma. **Modifikáciu** databázy so schémou $\mathcal{D}$ si môžeme definovať ako zobrazenie typu $Mod(\mathcal{D}) \longrightarrow Mod(\mathcal{D})$, kde $Mod(\mathcal{D})$ značí triedu modelov schémy $\mathcal{D}$. Teda ide o zobrazenie z modelov tejto schémy do seba.

Nech β je kartézsky súčin $T_{i_1} \times \cdots \times T_{i_m}$ niektorých typov z $\mathcal{D}$. **Otázka** typu β nad databázovou schémou $\mathcal{D}$ je čiastočné zobrazenie, ktoré každej inštancii schémy $\mathcal{D}$ priradí nejakú reláciu ρ, pričom $\rho \subseteq \beta$.

Podmnožinu takto definovaných otázok možno vyjadriť v relačnej algebre alebo – ekvivalentne – v relačnom kalkule. (Relačné) databázové jazyky slúžia na vyjadrenie otázok. Základné teoretické modely relačných databázových jazykov sú relačná algebra a reačný kalkul.

Relačná algebra poskytuje jazyk, v ktorom sa priamo pomenúvajú relácie z danej inštancie databázy a na týchto reláciách sa špecifikujú operácie. Všetky operácie sa definujú z týchto základných:

- **kartézsky súčin**,
- **zjednotenie** (ak máme relácie ρ_1, ρ_2 toho istého typu, t.j. podmnožiny kartézskeho súčinu $T_1 \times \cdots \times T_k$, potom ich zjednotením je relácia $\rho_1 \cup \rho_2$, do ktorej patrí každá k-tica, ktorá patrí buď do ρ_1 alebo do ρ_2),
- **rozdiel** (ak máme relácie ρ_1, ρ_2 toho istého typu, t.j. podmnožiny kartézskeho súčinu $T_1 \times \cdots \times T_k$, potom ich rozdielom je relácia $\rho_1 \setminus \rho_2$, do ktorej patrí každá k-tica, ktorá patrí do ρ_1, ale nepatrí do ρ_2),
- **projekcia** (ak máme reláciu ρ arity k so schémou $p(X_1 : T_1, \ldots, X_k : T_k)$, potom jej projekciou na stĺpce $i_1, \ldots, i_l$ alebo atribúty $X_{i_1}, \ldots, X_{i_l}$, kde $l < k$ – značenie $\pi_{X_{i_1},\ldots,X_{i_l}}$ alebo $\pi_{i_1,\ldots,i_l}$ – je množina všetkých l-tíc $(\nu_1, \ldots, \nu_l)$ takých, že existuje k-tica $(\mu_1, \ldots, \mu_k) \in \rho$ taká, že $\nu_j = \mu_{i_j}$),
- **selekcia** (ak ϕ je formula, ktorá obsahuje koštanty z domény, označenia stĺpcov – alebo atribútov – relácií danej databázy, aritmetické relačné symboly ako $<, >$, logické spojky $\wedge, \vee, \neg$, potom selekcia $\sigma_\phi(\rho)$ je množina tých k-tic z ρ, ktoré spĺňajú podmienku ϕ; ϕ sa nazýva selekčnou podmienkou, jazyky, v ktorých sa píšu selekčné podmienky môžu byť aj silnejšie než ten, ktorý bol načrtnutý vyššie).

Relačný kalkul je databázový jazyk nad relačnými databázami, ktorý vychádza z jazyka logiky a používa sa na definovanie relácií, ktoré majú byť odpoveďou na otázku.

Formulami relačného kalkulu sú:

(i) Každý atóm tvaru $p(t_1, \ldots, t_k)$, kde p je meno relácie z danej databázovej schémy a t_i je buď premenná[2] alebo je konštanta označujúca objekt domény tohto atribútu.

[2] V jadre tohto textu, kapitola 2, sme videli súvislosť medzi premennými a atribútmi. Každý atribút z databázovej schémy možno chápať ako premennú, ktorej hodnotami môžu byť objekty z domény (typu) daného atribútu.

(ii) Každý atóm tvaru $t_1\theta t_2$, kde t_i sú buď premenné alebo konštanty, θ je relačný symbol jazyka, v ktorom sa píšu selekčné podmienky.[3] Atómy tohto tvaru nazývame selekčnými podmienkami.

(iii) Ak F a G sú formuly, potom aj $F \wedge G$, $F \vee G$, $\neg F$, $\exists X F$ sú formuly.

Teraz budeme definovať bezpečnú formulu. Zavedieme tým pojem umožňujúci iba také otázky v relačnej algebre, na ktoré sa dá odpovedať iba konečnými reláciami. Najprv potrebujeme pojem limitovanej premennej. Intuitívne voľná premenná je limitovaná, ak možno za ňu substituovať nejakú konštantu z nejakej databázovej relácie priamo alebo prostredníctvom série rovností. Voľná premenná X je **limitovaná**, ak sa vyskytuje:

(i) vo formule F, pričom F nie je selekčná podmienka a nie je negovaná,

(ii) v atómoch tvaru $a = X$ alebo $X = a$, kde a je konštanta,

(iii) v atómoch tvaru $X = Y$ alebo $Y = X$, kde Y je limitovaná premenná.

Bezpečná formula spĺňa tieto obmedzenia:

- Oba disjunkty v disjunkcii $F \vee G$ majú rovnakú množinu voľných premenných.
- Ak $F_1 \wedge \cdots \wedge F_n$ je maximálna podformula danej formuly, ktorá má tvar konjunkcie, potom každá voľná premenná v tejto konjunkcii musí byť limitovaná.
- Negovaná formula $\neg G$ sa môže vyskytovať iba v konjunkcii tvaru

  $$F_1 \wedge \cdots \wedge F_k \wedge \neg G \wedge H_1 \wedge \cdots \wedge H_m,$$

  pričom aspoň jedno F_i a aspoň jedno H_j nie je negácia a v konjunkcii sa vyskytujú iba limitované voľné premenné.

Otázka v relačnom kalkule má tvar $\{X_1, \ldots, X_n | F(X_1, \ldots, X_n)\}$, kde F je bezpečná formula relačného kalkulu a $X_1, \ldots, X_n$ sú voľné premenné v F.

Tvrdenie C.1 *Každá otázka vyjadriteľná v relačnej algebre je vyjadriteľná v relačnom kalkule. Každá otázka vyjadriteľná v relačnom kalkule je vyjadriteľná v relačnej algebre.*

Poznámky: Definitoricky sme otázky relačného kalkulu obmedzili na bezpečné otázky relačného kalkulu. Keďže otázku chápeme ako funkciu z množiny inštancií databázovej schémy do množiny relácií istého typu, vyššieuvedené tvrdenie vlastne hovorí, že nejaký výraz relačnej algebry špecifikuje nejakú funkciu tohto typu práve vtedy, keď nejaká otázka relačného kalkulu špecifikuje tú istú funkciu.

[3] θ môže byť napríklad $<$ alebo $\leq$.

Dodatok D

Logické programovanie

Úvodom treba pripomenúť, že tento dodatok sa svojím obsahom čiastočne prekrýva s časťou 2.3. Dôvody pre tieto prekrytia sú nasledovné: V časti 2.3 je v centre pozornosti dedukcia a rozšírenie prostriedkov reprezentácie poznatkov od relačných databáz k deduktívnym. Pre čitateľa, ktorý už pozná teóriu logického programovania, má tento dodatok slúžiť ako príručka. Môže v nej nájsť (občerstviť) nejaké definície a tvrdenia. Pre čitateľa, ktorý sa ešte nestretol s teóriou logického programovania, môžu byť spomínané partie ťažšie. Preto pokus o systematický náčrt tejto teórie môže byť pre neho užitočný, podobne aj prípadné opakovanie informácií, ktoré našiel v jadre textu.

Logické programovanie je po výpočtovej stránke založené na automatickom dokazovaní teorém logiky, na rezolvenčnej verzii automatického dokazovania.

Najprv teda pojmy súvisiace s automatickým dokazovaním.

Rezolvenčná procedúra sa spúšťa na formulách, ktoré sú upravené štandardným spôsobom – sú v Skolemovej normálnej forme. Prv si však potrebujeme objasniť pojem prenexnej a konjunktívnej normálnej formy. Najprv zavedieme definíciu, ktorá bola v časti 2.3 uvedená ako definícia 2.38.

Definícia D.1 Literál je atóm alebo jeho negácia. **Klauza** je disjunkcia literálov. **Hornovská klauza** je klauza, ktorá obsahuje nanajvýš jeden pozitívny literál. **Definitná klauza** je hornovská klauza, ktorá obsahuje presne jeden pozitívny literál.

Základná klauza neobsahuje žiadnu premennú. ■

Napríklad $p \vee \neg q_1, \vee \cdots \vee \neg q_k$ je hornovská klauza, ale $p \vee q$ nie je hornovská klauza.

Formula jazyka predikátovej logiky prvého rádu je v **prenexnej normálnej forme**, ak je tvaru $\kappa_1, \ldots, \kappa_k\ \Phi$, kde $\kappa_1, \ldots, \kappa_k$ sú všetky kvantifikátory formuly a Φ je v **konjunktívnej normálnej forme**, t.zn Φ je konjunkcia disjunkcií literálov (Φ je tvaru $\xi_1 \wedge \cdots \wedge \xi_m$, kde ξ_i je $\psi_1 \vee \cdots \vee \psi_n$, pričom ψ_j je literál).

Tvrdenie D.2 *Každú formulu F predikátového počtu prvého rádu možno transformovať na formulu F' v prenexnej normálnej forme tak, že F a F' sú ekvivalentné.*

Majme uzavretú formulu v prenexnej normálnej forme. Ukážeme si, ako môžeme vystačiť bez kvantifikátorov. Procedúra, ktorá to umožňuje, sa nazýva **skolemizácia**.

Existenčný kvantifikátor $\exists X$, pred ktorým nestojí žiadny všeobecný kvantifikátor, odstránime tak, že každý výskyt premennej X vo formuli Φ nahradíme novou, ešte nepoužitou konštantou (**Skolemova konštanta**).

Ak pred existenčným kvantifikátorom $\exists X$ stoja všeobecné kvantifikátory $\forall Y_1, \ldots \forall Y_l$, každý výskyt premennej X nahradíme výskytom termu $f(Y_1, \ldots, Y_l)$, kde f je nový, ešte nepoužitý funkčný symbol.

Keď sme týmto spôsobom odstránili existenčné kvantifikátory, môžeme zabudnúť aj na všeobecné kvantifikátory. Formula je totiž uzavretá – postačí nám teda dohoda o tom, že každú premennú môžeme považovať za všeobecne kvantifikovanú.

Teda, formula je v **Skolemovej normálnej forme**, ak ju získame skolemizáciou z uzavretej formuly, ktorá je v prenexnej normálnej forme. Keďže formula v prenexnej normálnej forme je konjunkcia nejakých disjunkcií, dá sa reprezentovať ako množina zoznamov literálov – každý prvok tejto množiny je jeden konjukt, každý zoznam literálov sa chápe ako ich disjunkcia.

Pripomeňme si, že formula F je nekonzistentná, ak neexistuje interpretácia, pri ktorej je F pravdivá. Množina formúl M je nekonzistentná, keď neexistuje interpretácia, pri ktorej sú všetky formuly z M súčasne pravdivé.

Tvrdenie D.3 (Transformácia na Skolemovu normálnu formu) *Nech S je Skolemova normálna forma formuly F. Potom S je nekonzistentná práve vtedy, keď F je nekonzistentná.*

Uvedená teoréma zabezpečuje, že transformácia na Skolemovu normálnu formu neznehodnotí rezolvenčné dokazovanie. Rezolvenčný dôkaz je totiž nepriamy. Cieľom je ukázať, že negácia dokazovanej formuly spôsobí nekonzistentnosť, ak ju pripojíme k množine formúl, z ktorých odvodzujeme. Teoréma ukazuje, že nekonzistentnosť sa skolemizáciou zachováva.

Poznámka: Skolemova normálna forma S konzistentnej formuly F nemusí byť ekvivalentná s F.

Teraz treba venovať pozornosť samotnému rezolvenčnému dôkazu. Pri rezolvenčnom dokazovaní sa používa jediné odvodzovacie pravidlo, rezolvenčné pravidlo. Toto pravidlo je vlastne zovšeobecnením pravidla modus ponens $(\{p, p \to q\}, \{q\})$, ktoré z predpokladov p a $p \to q$ umožňuje odvodiť záver q. Ak si toto pravidlo vyjadríme pomocou disjunkcie a negácie, dostaneme:

$(\{p, \neg p \vee q\}, \{q\})$. Vidíme teda, že keď medzi predpokladmi máme p i $\neg p$, môžeme na oba tieto konfliktné literály zabudnúť (eliminovať ich, odstrániť ich) a pokračujeme v odvodzovaní z ich dôsledku. Na to, aby sme túto myšlienku mohli formalizovať v plnej všeobecnosti, potrebujeme prejsť do jazyka predikátovej logiky: konflikt uvedeného typu môže nastať, ak máme dva literály, $\psi(X)$ a $\neg\psi(a)$ (alebo naopak, $\psi(a)$ a $\neg\psi(X)$), či $\psi(X)$ a $\neg\psi(Z)$. Proti sebe tu stoja v konflikte jedna všeobecná formulácia proti jednej konkrétnej (základnému literálu) alebo dve všeobecné, ale syntakticky odlišné, formulácie. Potrebujeme jednoduchú formálnu operáciu, ktorá umožní oba komponenty konfliktného páru mechanicky porovnať, v istom zmysle stotožniť (až na to, že jedna z nich je negáciou druhej). Touto operáciou je **unifikácia**. Intuitívne, unifikácia spočíva v postupnom substituovaní za premenné v nejakej množine formúl. Unifikácia končí úspechom, ak sa nám podarí túto množinu stotožniť s jedinou formulou. Teda, množina formúl M je unifikovateľná, ak existuje spoločná inštancia všetkých formúl z M.

Substitúcia priraďuje každej premennej (jazyka predikátového počtu J) nejaký term.

Teraz si zadefinujeme, čo urobí substitúcia so zložitejším termom a s formulou. Samotná definícia je zdĺhavá, jej základná myšlienka jednoduchá: substitúcia za všetky výskyty premennej v celom terme alebo v celej formule.

Nech σ je substitúcia, t term, F formula, M množina formúl, potom:

- $\sigma(t)$ je term, ktorý získame z t tak, že každú voľnú premennú ξ v t prepíšeme termom $\sigma(\xi)$,
- $\sigma(F)$ je formula, ktorú získame z F tak, že každú voľnú premennú ξ v F prepíšeme termom $\sigma(\xi)$,
- $\sigma(M)$ je taká množina formúl, ktorú získame z M substitúciou každej voľnej premennej ξ v M termom $\sigma(\xi)$.

Dôležitou operáciou je aplikovanie dvoch substitúcií po sebe. Táto operácia sa nazýva kompozíciou. Nech σ a θ sú substitúcie. Predpokladáme, že najprv sa vykonala substitúcia σ, po nej θ. Ich **kompozíciou** $\sigma \circ \theta$ je substitúcia, ktorá každej premenej ξ priraďuje term $\theta(\sigma(\xi))$.

Klauza ϕ je **inštanciou** klauzy ψ, ak ϕ možno získať z ψ substitúciou. Klauza ϕ je **základnou inštanciou** klauzy ψ, ak ϕ je základná klauza a je inštanciou ψ.

Unifikácia je špeciálny prípad substitúcie – substitúcia, ktorá stotožní viac formúl (pred aplikáciou tejto substitúcie rôznych). Inak: **unifikátorom** množiny M je taká substitúcia σ, pre ktorú $\sigma(M)$ je jednoprvková množina.

Príklad D.4 Nech $M = \{p(X, a), p(f(b), Y), p(U, V)\}$. Nech $\sigma(X) = f(b) = \sigma(U), \sigma(Y) = a = \sigma(V)$. Dostávame $\sigma(M) = \{p(f(b), a)\}$. ■

Unifikáciu množiny formúl môžeme robiť rôznymi spôsobmi.

Príklad D.5 Unifikátormi $M = \{p(X), p(Y)\}$ sú napríklad $\{p(a)\}$ aj $\{p(X)\}$. Unifikácia na $\{p(X)\}$ je v istom zmysle všeobecnejšia ako na $\{p(a)\}$: existuje substitúcia, ktorá všeobecnejší tvar prevedie na menej všeobecnejší (je to substitúcia a za X). ■

Tieto úvahy nás priviedli k dôležitému pojmu najvšeobecnejšieho unifikátora.

Najvšeobecnejším unifikátorom M, množiny formúl jazyka J, je substitúcia σ taká, že pre každý unifikátor θ množiny M existuje substitúcia γ, pre ktorú $\theta(X) = \gamma(\sigma(X))$ pre každú premennú X vyskytujúcu sa v M.

Algoritmus D.6 (Unifikačný algoritmus)
Vstup: *M – množina formúl.*
Výstup: *Rozhodnutie, či M je unifikovateľná. Ak je, výstupom je aj konštrukcia najvšeobecnejšieho unifikátora množiny M.*

1. *Inicializácia: $k := 0, \sigma_0 := \epsilon$, kde ϵ je identická substitúcia.*

2. *Ak $\sigma_k(M)$ je jednoprvková množina, σ_k je najvšeobecnejší unifikátor formúl z množiny M. Algorimus zastaví.*

3. *V opačnom prípade – nájdi najľavejší znak $l(F)$ každej formuly F z M, ktorým sa tieto formuly líšia. Nech D_k je množina všetkých termov (podvýrazov formúl z M), ktoré začínajú v $l(F)$.*

4. *Ak v D_k existuje premenná ξ a term t (pritom ξ sa nevyskytuje v t), bude $\sigma_{k+1} := \sigma_k \circ \gamma$, kde γ priradí premennej ξ term t a ostatné premenné necháva nedotknuté, $k := k + 1$. Prechod ku kroku 2.*

5. *V opačnom prípade, M nie je unifikovateľná, algoritmus zastaví.*

Tvrdenie D.7 (Unifikačná teoréma) *Nech M je konečná množina formúl. Unifikačný algoritmus v každom prípade zastaví. Ak M je unifikovateľná, potom vypočíta najvšeobecnejší unifikátor množiny M. Ak M nie je unifikovateľná, algorimus to zistí.*

Rezolvenčné pravidlo: Nech C a C' sú klauzy. C obsahuje pozitívny literál L, C' obsahuje negatívny literál $\neg L'$. Ak σ je najvšeobecnejší unifikátor literálov L a L', potom rezolventou kláuz C a C' je klauza $(\sigma(C) \setminus \{L\}) \cup (\sigma(C') \setminus \{\neg L'\})$. (Množinové operácie sme si mohli dovoliť vzhľadom na množinovú konvenciu pri zápise kláuz.)

Procedúra rezolvencie:

- štartujeme z nejakej množiny kláuz, medzi ktorými je aj negácia cieľa, ktorý chceme dokázať,

- aplikujeme pravidlo rezolvencie dovtedy, kým sa podarí odvodiť spor (reprezentovaný prázdnou klauzou, ktorú budeme značiť ako [ ]).

Procedúra rezolvencie nemusí zastaviť, bližšie pozri v dodatku H.

Tvrdenie D.8 (Úplnosť rezolvencie) *Ak M je nekonzistentná množina kláuz, potom pomocou pravidla rezolvencie ju možno transformovať v prázdnu klauzu.*

Prv než prejdeme k SLD-rezolvencii, špeciálnej rezolvenčnej stratégii,[1] využívanej v logickom programovaní, trochu terminológie a konvencií značenia.

Klauza C je variantom klauzy C' vtedy, keď existuje také premenovanie premenných σ (môžeme ho chápať ako substitúciu premenných za premenné), po ktorom je $\sigma(C)$ zhodné s C'.

Definitná klauza obsahuje nanajvýš jeden pozitívny literál. Pretože $(\neg p \vee q) \equiv (q \leftarrow p)$ a $\neg(\neg q_1 \vee \cdots \vee \neg q_k) \equiv q_1 \wedge \cdots \wedge q_k$ sú tautológie výrokovej logiky, definitné klauzy môžeme písať v tvare $p \leftarrow q_1, \ldots, q_k$, kde čiarky na pravej strane znamenajú konjunkciu.

Logický program je množina definitných kláuz tvaru $A \leftarrow B_1, \ldots, B_k$. Logické programy tohto tvaru sa nazývajú definitné. Definitný cieľ je hornovská klauza tvaru $\leftarrow B_1, \ldots, B_k$. V oboch prípadoch A, B_i sú atómy. A nazývame hlavou, B_i telom klauzy.

Ak po pridaní cieľa $\leftarrow B_1, \ldots, B_k$ k programu P dostaneme (vďaka vhodným substitúciám) nekonzistentnú množinu kláuz, dokázali sme, že konjunkcia $B_1 \wedge \cdots \wedge B_k$ (presnejšie, nejaká jej inštancia) vyplýva z P. (Totiž, $\sigma(\leftarrow B_1, \ldots, B_k)$ je ekvivalentné so $\sigma(\neg B_1 \vee \cdots \vee \neg B_k)$, to je ekvivalentné so $\sigma(\neg(B_1 \wedge \cdots \wedge B_k))$. Pritom $P \cup \{\sigma(\neg(B_1 \wedge \cdots \wedge B_k))\}$ je nekonzistentná množina kláuz práve vtedy, keď $P \models \sigma((B_1 \wedge \cdots \wedge B_k))$.

Ak C je klauza a σ substitúcia, budeme popri značení $\sigma(C)$ používať v rovnakom význame aj bežnejšie $C\sigma$. Teraz už môžeme prejsť k SLD-rezolvencii. Najprv si definujeme odvodzovací krok.

Definícia D.9 Majme cieľ G:

$$\leftarrow A_1, \ldots, A_i, \ldots, A_k$$

a klauzu C:

$$A \leftarrow B_1, \ldots, B_m.$$

Nech θ je najvšeobecnejší unifikátor atómov A a A_i. Hovoríme, že cieľ G' získame **odvodzovacím krokom** z G a C pomocou θ, ak G' je

$$\leftarrow A_1\theta, \ldots, A_{i-1}\theta, B_1\theta, \ldots, B_m\theta, A_{i+1}\theta, \ldots, A_k\theta.$$

G' nazývame **rezolventou** G a C. ■

---

[1] Skratka SLD znamená: používa sa nejaká selekčná funkcia (S), ktorá vyberá jediný cieľ z množiny cieľov; odvodenie je lineárne (L) v tom zmysle, že v každom kroku odvodenia sa na daný cieľ aplikuje jediná klauza a vygeneruje sa jediný nový cieľ; napokon – stratégia sa aplikuje na definitné (D) klauzy, pojem definitnej klauzy nasleduje.

SLD-rezolvencia obvykle pozostáva z celej série odvodzovacích krokov.

Definícia D.10 Nech P je definitný program a G_0 je definitný cieľ. *SLD*-**odvodením** z P a G_0 nazývame trojicu postupností:

$$(G_0, G_1, \dots), (C_1, C_2, \dots), (\theta_1, \theta_2, \dots),$$

pričom G_i sú ciele, C_i sú varianty kláuz z programu P, θ_i sú najvšeobecnejšie unifikátory také, že G_{i+1} získame odvodzovacím krokom z G_i a C_{i+1} pomocou θ_{i+1}.[2]

SLD-**vyvrátenie** pre $P \cup \{G\}$ je konečným *SLD*-odvodením z P a G, kde posledným cieľom je prázdna klauza [ ]. ■

Teda, SLD-rezolvencia je úspešná, ak sa jej podarí vyvrátiť cieľ (opakujeme, ide o nepriamy dôkaz). Idea logického programovania spočíva v tom, že problémy, ktoré treba riešiť (úlohy, ktoré treba vypočítať) sa formulujú ako otázky. Tie otázky sa položia programu. Program na ne odpovedá. Dokazovanie sa používa ako nástroj na odpovedanie. Technika, ktorá sa pri dokazovaní používa, je *SLD*-vyvrátenie. Ešte raz: odpoveď na otázku sa získa dôkazom z daného programu. Poznamenajme, že *SLD* rezolvencia je cieľovo orientovaná. Štartuje sa z cieľa G a v programe P sa postupne hľadajú vhodné klauzy, kým sa nedokáže spor (kým sa neodvodí [ ]).

Definícia D.11 Nech P je definitný program a G je definitný cieľ. **Vypočítaná odpoveď** pre $P \cup \{G\}$ je substitúcia θ, ktorú získame kompozíciou $\theta_1 \circ \dots \circ \theta_n$ všetkých najvšeobecnejších unifikátorov získaných pri *SLD*-vyvrátení $P \cup \{G\}$. ■

Teraz môžeme prejsť k sémantickej charakterizácii logických programov.

Pokiaľ nebude ďalej vyslovene uvedené inak, budeme hovoriť iba o herbrandovských interpretáciách a modeloch. Sú to interpretácie (modely) skonštruované z predikátových symbolov, konštánt a funkčných symbolov uvažovaného jazyka.

Príklad D.12 Datalogovská databáza $\mathcal{D}$:

$$\begin{array}{rcl} p(a) & \leftarrow & , \\ q(b) & \leftarrow & , \\ r(X,Y) & \leftarrow & p(X), q(Y) \end{array}$$

má okrem iných tieto herbrandovské modely (herbrandovské interpretácie tu - a všade ďalej - reprezentujeme množinou základných atómov, ktoré daná interpretácia „robí pravdivými“):

[2] SLD-odvodenie môže byť aj nekonečné.

$$M_1 = \{p(a), p(b), q(b), r(a,b), r(b,b), r(b,a)\}.$$

M_1 je modelom $\mathcal{D}$, pretože $\mathcal{D}$ je pravdivá, ak sú všetky atómy z M_1 pravdivé.

$$M_2 = \{p(a), q(b), r(a,b)\}.$$

M_2 je minimálny model $\mathcal{D}$, znamená to toľko, že neexistuje vlastná podmnožina M_2, ktorá by bola modelom $\mathcal{D}$.

Keďže herbrandovské modely môžeme chápať ako množiny atómov, má zmysel uvažovať reláciu inklúzie medzi modelmi. Ak máme triedu modelov $\mathcal{M}$, minimálnym modelom v tejto triede je taký model $M \in \mathcal{M}$, pre ktorý platí, že neexistuje model $M' \in \mathcal{M}$, taký, že $M' \subset M$.

Každá datalogovská databáza má jediný minimálny (herbrandovský) model. (Stačí si uvedomiť, že množina $B_{\mathcal{D}}$ všetkých základných atómov, konštruovateľných v jazyku tejto databázy, je jej modelom a že prienik množiny modelov datalogovskej databázy je jej modelom.)

K minimálnemu modelu sa môžeme dostať aj akýmsi výpočtom zdola nahor (od faktov cez pravé strany kláuz k stále novým cieľom, užitočné bude porovnať tento postup s algoritmom 2.47). Tento postup nazývame $T_{\mathcal{D}}$-iteráciou. Ilustrujeme ho na našom príklade. Nech $\emptyset$ je prázdna interpretácia - ani jeden atóm nie je pri nej pravdivý.

$$\begin{aligned} T_{\mathcal{D}}(\emptyset) &= \{p(a), q(b)\} = N_1 \\ T_{\mathcal{D}}(N_1) &= \{p(a), q(b), r(a,b)\} = N_2 \\ T_{\mathcal{D}}(N_2) &= N_2 \end{aligned}$$

N_2 je (najmenším) pevným bodom operátora $T_{\mathcal{D}}$ pre danú databázu.

Najmenší pevný bod operátora $T_{\mathcal{D}}$ je minimálnym modelom $\mathcal{D}$. ■

Teraz definície:

Definícia D.13 Nech $Ground(P)$ je množina základných inštancií databázy (programu) P, t.j. množina všetkých kláuz, ktoré získame z kláuz v P substitúciou konštánt jazyka databázy P za všetky premenné.

Potom T_P je zobrazenie z herbrandovských interpretácií do herbrandovských interpretácií, pričom

$$T_P(I) = \{A \in B_P : A \leftarrow A_1, \ldots, A_n \in Ground(P) \wedge \{A_1, \ldots, A_n\} \subseteq I\}.$$

■

Definícia D.14 I je **pevným bodom** operátora T_P práve vtedy, keď

$$T_P(I) = I.$$

I je najmenší pevný bod operátora T_P, ak pre každý pevný bod J tohto operátora platí, že $I \subseteq J$. ■

Definícia D.15 Nech P je definitný program, G je definitný cieľ a θ je substitúcia za (niektoré) premenné v G. Potom θ je **korektná odpoveďová substitúcia** pre $P \cup \{G\}$, ak $\forall(G\theta)$, všeobecný uzáver formuly $G\theta$, logicky vyplýva z P. ■

Podmienkou korektnosti riešenia problémov v logickom programovaní je teda vzťah logického vyplývania.

Tvrdenie D.16 *Nech P je definitný program, G je definitný cieľ a θ je substitúcia taká, že $G\theta$ neobsahuje žiadnu premennú. Potom nasledujúce podmienky sú ekvivalentné:*

(i) *θ je korektná odpoveďová substitúcia,*

(ii) *$G\theta$ je pravdivá v každom herbrandovskom modeli programu P,*

(iii) *$G\theta$ je pravdivá v minimálnom herbrandovskom modeli programu P,*

(iv) *$G\theta$ patrí do najmenšieho pevného bodu operátora T_P.*

Poznámka: Podobná ekvivalencia neplatí pre ľubovoľné korektné odpoveďové substitúcie.

Tvrdenie D.17 *Nech M_P je minimálny model logického programu P, $\mathit{lfp}(T_P)$ je najmenší pevný bod operátora T_P. Potom $M_P = \mathit{lfp}(T_P)$.*

Tvrdenie D.18 *SLD-rezolvencia je korektná a úplná: Každá SLD-vypočítaná odpoveď je korektná odpoveďová substitúcia. Pre každú korektnú odpoveďovú substitúciu θ existuje všeobecnejšia SLD-vypočítaná odpoveď σ (to znamená, že existuje aj substitúcia γ taká, že $\theta = \sigma \circ \gamma$).*

Ide teda o úplnosť v slabšom zmysle slova, keďže SLD-výpočtová procedúra produkuje iba najvšeobecnejšie unifikátory.

Všeobecným logickým programom (programom, v ktorých tele sa môžu vyskytovať aj negatívne literály) sa venuje značná pozornosť v kapitole 6.

Dodatok E

Algebra a množiny

V tomto dodatku sú zhrnuté základné pojmy z algebry, teórie grafov a teórie množín, používané v jadre textu. Podrobnejšie napríklad v [Wec 92], [Gru 97], [Buk 85] a v množstve iných kníh.

Môžeme predpokladať, že pojem množiny je dostatočne intuitívny, nespôsobuje nijaké ťažkosti pri bežnom používaní (čitateľ je však pravdepodobne oboznámený s dôvodmi obozretného a sofistikovaného narábania s pojmom množiny). **Množina** pozostáva z prvkov. Fakt, že prvok a patrí do množiny M značíme ako $a \in M$. Ak a nepatrí do M: $a \notin M$. Prázdna množina neobsahuje žiadny prvok, značíme ju ako $\emptyset$.

Množina sa dá určiť buď vymenovaním jej prvkov (u konečných množín) alebo nejakou vlastnosťou. Oba spôsoby ilustrujeme na príklade: $\{1, 6, 23\}$ je množina, ktorá obsahuje $1, 6$ a 23 a neobsahuje nijaké iné prvky, $\{x : x > 6\}$ je množina všetkých prirodzených čísel väčších ako 6. Samozrejme, v druhom príklade sme predpokladali, že sa bavíme iba o prirodzených číslach. Ak to nemôžeme z nejakých dôvodov predpokladať, píšeme $\{x \in \mathit{Nat} : x > 6\}$, kde *Nat* je množina všetkých prirodzených čísel. Vidíme, že pomocou nejakej (charakteristickej) vlastnosti môžeme určiť aj nekonečné množiny.

Množina M je **podmnožinou** množiny N, značenie $M \subseteq N$, ak každý prvok z M patrí aj do N. Pomocou logickej formuly: $\forall x\,(x \in M \to x \in N)$. M je **vlastná** podmnožina množiny N, ak $M \subseteq N$, ale $M \neq N$, značenie $M \subset N$. Množinu všetkých podmnožín nejakej množiny M značíme ako 2^M: ak $X \subseteq M$, potom $X \in 2^M$. (Množina všetkých podmnožín nejakej množiny M sa zvykne nazývať aj **potenčnou** množinou množiny M a značiť ako $\mathcal{P}(M)$.)

Nech $\mathcal{M}$ je nejaká množina množín (množina, ktorej prvkami sú množiny). Definujeme **zjednotenie** $\bigcup$ a **prienik** $\bigcap$ (trochu všeobecnejšie ako v dodatku C): $\bigcup\ \mathcal{M} = \{x : (\exists N \in \mathcal{M})\ x \in N\}$, $\bigcap\ \mathcal{M} = \{x : (\forall N \in \mathcal{M})\ x \in N\}$. Ak prienik dvoch množín je prázdna množina, hovoríme, že tieto množiny sú **disjunktné**. **Rozdiel** dvoch množín $M \setminus N$ je definovaný ako množina, ktorá obsahuje iba a práve tie prvky z M, ktoré nepatria do N (pomocou formuly: $M \setminus N = \{x \in M : x \notin N\}$). Ak je definovaný nejaký základný priestor (univerzum), nejaká množina U, potom **komplementom** množiny N (v priestore U) je množina všetkých prvkov zo základného priestoru U, ktoré nepatria do N. Značenie: $\overline{N} = U \setminus N$. **Kartézsky súčin** množín $M_1, \ldots, M_n$ je množina všetkých možných n-tíc vytvorených (po komponentoch) z množín M_i, presnejšie: $M_1 \times \cdots \times M_n = \{(x_1, \ldots, x_n) : \forall i \in \{1, \ldots, n\}\ (x_i \in M_i)\}$. **Relácia** na $M_1, \ldots, M_n$ je ľubovoľná podmnožina $M_1 \times \cdots \times M_n$. Kartézsky súčin $M \times M$ budeme značiť ako M^2, analogicky sa zavádza značenie M^n pre ľubovoľné prirodzené číslo n. Reláciu na $M_1 \times M_2$ voláme **binárnou** reláciou. Budeme hovoriť, že relácia na M^2 je reláciou na M.

Prejdeme k definícii **funkcie (zobrazenia)**. Nech M a N sú ľubovoľné množiny, teda môžu to byť aj relácie. Funkcia (zobrazenie) f z M do N, značenie $f : M \longrightarrow N$, sa definuje ako taká relácia $f \subseteq M \times N$, že pre každé $x \in M$ existuje presne jedno $y \in N$ také, že $(x, y) \in f$. Značenie: $f(x) = y$, prípadne $f(x_1, \ldots, x_n) = y$. M sa nazýva **definičným oborom** (doménou) funkcie f, značenie $Dom(f)$ a N **oborom hodnôt**, $Range(f)$. Ak $f(x) = y$, budeme hovoriť, že x je vzor (f-vzor) a y je obraz (f-obraz). Čiatočná funkcia (čiastočné zobrazenie) nie je definovaná pre každé x z jej definičného oboru. **Čiastočná** funkcia $f : M \longrightarrow N$ je taká relácia $f \subseteq M \times N$, že pre $(x, y) \in f$ a $(x, y') \in f$ platí, že $y = y'$. Môže sa však stať, že pre nejaké $x \in M$ neexistuje $y \in N$ také, že $(x, y) \in f$. Funkcia f, ktorá je definovaná pre každý prvok množiny $Dom(f)$, sa nazýva aj všade definovaná alebo totálna.

Majme funkciu $f : M \longrightarrow N$. Ak $M' \subset M$, g je funkcia z M' do N, $g(x) = f(x)$ pre všetky $x \in M'$, hovoríme, že g je **reštrikciou** (obmedzením) f na M'.

Zobrazenie f je **injekcia**, ak pre $x \neq y$ platí $f(x) \neq f(y)$, teda rôzne f-vzory majú rôzne f-obrazy. **Surjekcia** je také zobrazenie $f : M \longrightarrow N$, že pre každé $y \in N$ existuje $x \in M$, že $f(x) = y$. Teda, každý prvok z N má nejaký f-vzor v M, t.j. $Range(f) = N$. **Bijekcia** je zobrazenie, ktoré je súčasne injekciou i surjekciou.

Majme množiny M a N také, že existuje bijekcia $f : M \longrightarrow N$. Hovoríme, že M a N majú rovnakú **kardinalitu (mohutnosť)**, značenie $card(M) = card(N)$. V takom prípade existuje aj bijekcia $g : N \longrightarrow M$, g sa nazýva zobrazením inverzným k f a značí sa aj ako f^{-1}, ak platí $f(x) = y$ práve vtedy, keď $g(y) = x$. Zrejme platí $(f^{-1})^{-1} = f$.

M je **spočítateľná nekonečná** množina, ak existuje bijekcia z množiny všetkých prirodzených čísel na M. **Spočítateľná** je buď konečná množina alebo spočítateľná nekonečná množina.

Ak M je spočítateľná množina, potom množina všetkých jej konečných podmnožín je spočítateľná. Kartézsky súčin dvoch spočítateľných množín je spočítateľná množina. Množina všetkých celých čísel je spočítateľná. Množina všetkých racionálnych čísel je spočítateľná. Možina všetkých reálnych čísel nie je spočítateľná (jej mohutnosť sa nazýva *mohutnosťou kontinua*). Mohutnosť množiny všetkých funkcií z *Nat* do *Nat* je kontinuum. Mohutnosť všetkých podmnožín spočítateľnej nekonečnej množiny je kontinuum.

Predpokladajme ďalej množinu M a budeme definovať nejaké **usporiadania** na tejto množine. Nech $\rho \subseteq M^2$ je binárna relácia na M. Hovoríme, že ρ je **reflexívna**, ak platí $(x,x) \in \rho$ pre každé x.[1] Podobne, ρ je symetrická, ak pri platnosti $(x,y) \in \rho$ platí aj $(y,x) \in \rho$ pre každé $x,y \in M$. Relácia ρ je **antisymetrická**, ak z $(x,y) \in \rho$ a $(y,x) \in \rho$ dostávame $x = y$ (symetria môže byť iba medzi identickými prvkami). Binárna relácia ρ je **tranzitívna**, ak z $(x,y) \in \rho$ a $(y,z) \in \rho$ dostávame $(x,z) \in \rho$ pre každé $x,y,z \in M$. **Tranzitívny uzáver** binárnej relácie ρ je relácia

$$\{(x,y) : \exists n \; (\exists x_1, \ldots x_n) \; (x,x_1), (x_1,x_2), \ldots, (x_n,y) \in \rho\}.$$

Očividne, takáto relácia je tranzitívna. (Podobne možno definovať reflexívny uzáver, symetrický uzáver.)

Ekvivalencia je binárna relácia, ktorá je reflexívna, symetrická a tranzitívna. Ak $\equiv$ je ekvivalencia na M, potom M možno rozložiť na množiny vzájomne ekvivalentných prvkov: Nech u je ľubovoľný prvok z M. Pod $[u]_\equiv$ rozumieme množinu všetkých prvkov ekvivalentných s u, t.j. $[u]_\equiv = \{x \in M : x \equiv u\}$. Nazývame ju aj **triedou ekvivalencie**. Potom $M = \bigcup\{[u]_\equiv : u \in M\}$. Treba podotknúť, že platí: ak $[u]_\equiv \neq [v]_\equiv$, potom $[u]_\equiv \cap [v]_\equiv = \emptyset$. Množina X neprázdnych množín sa nazýva rozkladom množiny M, ak $\bigcup X = M$ a všetky (rôzne) množiny z X sú po dvojiciach disjunktné.

Čiastočným usporiadaním je relácia ρ, ktorá je reflexívna, antisymetrická a tranzitívna. Nech M je množina čiastočne usporiadaná reláciou ρ. Ak $x,y \in M$ a $(x,y) \in \rho$, potom hovoríme, že y je horným ohraničením pre x (x dolným ohraničením pre y). Podobne, nech $X \subseteq M$: y je horným (dolným) ohraničením X, ak pre každé $x \in X$ je y horným (dolným) ohraničením x. Supremom X ($supX$) je najmenšie horné ohraničenie X, t.j. $(supX, y) \in \rho$ pre každé horné ohraničenie y množiny X. Infimom X ($infX$) je najväčšie dolné ohraničenie X, t.j. $(y, infX) \in \rho$ pre každé dolné ohraničenie y množiny X. Zväz je taká čiastočne usporiadaná množina, že pre každú dvojicu x,y jej prvkov existuje ich infimum (priesek) $x \sqcap y = inf\{x,y\}$ i ich supremum (spojenie) $x \sqcup y = sup\{x,y\}$.

Lineárnym (totálnym) usporiadaním množiny M je také čiastočné usporiadanie, že pre každé dva rôzne prvky z $x,y \in M$ platí buď $(x,y) \in M$ alebo $(y,x) \in M$. Špeciálnym prípadom lineárneho usporiadania je nasledovné usporiadanie na k-ticiach: Nech $\xi = (x_1, \ldots, x_k) \in M_1 \times \cdots \times M_k$. Podobne $\xi' = (x'_1, \ldots, x'_k) \in M_1 \times \cdots \times M_k$. Budeme hovoriť, že na $M_1 \times \cdots \times M_k$ je definované **lexikografické usporiadanie** $\prec$, ak $\xi \prec \xi'$ platí práve vtedy, keď existuje j také, že $1 \leq j \leq k$, kde pre všetky $i < j$ je $x_i = x'_i$ a $x_j < x'_j$.

Zadefinujeme prvok, ktorý je minimálny vzhľadom na nejakú reláciu. Prvok minimálny vzhľadom na nejaké usporiadanie bude špeciálnym prípadom.

[1] Čitateľovi odporúčame značenie v tvare $(x,y) \in \rho$ prepisovať do tvaru $x \leq y$, ak mu tu používaná notácia prekáža.

Prvok $x \in M$ je **minimálny** v M vzhľadom na ρ, reláciu na M, ak neexistuje $y \in M$, také, že $x \neq y$, $(y, x) \in \rho$. V M môže existovať viac minimálnych prvkov vzhľadom na ρ. Ak ρ je lineárne usporiadanie, môže existovať iba jediný minimálny prvok danej množiny. Relácia ρ na M je **dobre založená**, ak každá neprázdna podmnožina M má minimálny prvok (vzhľadom na ρ). Množina M je **dobre usporiadaná** reláciou ρ, ak ρ je dobre založená a je lineárne usporiadanie.

Operácia (na M) je zobrazenie z M^n do M. Nech f je operácia. Zjednotenie, prienik, rozdiel, kartézsky súčin sú príkladmi operácií (na množinách). Budeme hovoriť, že množina M je **uzavretá** vzhľadom na f, ak f je totálna (všade definovaná) funkcia. (Množina prirodzených čísel nie je uzavretá vzhľadom na odčítanie, množina celých čísel je uzavretá vzľadom na odčítanie.)

Funkciu z množiny množín do množiny množín nazývame niekedy **operátorom** (všimnite si napríklad operátor Cn z jadra tohto textu). Keďže funkcia je vlastne nejaká množina (relácia so špeciálnymi vlastnosťami), funkcia definovaná na funkciách a produkujúca funkcie je operátor. Budú nás zaujímať pevné body operátorov: ak Φ je operátor a x patrí do $Dom(\Phi)$, potom x je **pevným bodom** pre Φ, ak $\Phi(x) = x$. **Najmenším** (vzhľadom na usporiadanie ρ) pevným bodom operátora Φ je taký jeho pevný bod x, že pre každý jeho pevný bod y platí $(x, y) \in \rho$. **Minimálnym** pevným bodom je pevný bod x, ak neexistuje pevný bod y taký, že $x \neq y$ a $(y, x) \in \rho$. Ak ρ je čiastočné usporiadanie, môže existovať viac minimálnych pevných bodov daného operátora. Podobne definujeme najväčší a maximálny pevný bod.

Algebraická štruktúra je n-tica $(U, \circ_1, \ldots, \circ_k, \rho_1, \ldots, \rho_m)$, kde U je nejaká množina (nosič, doména tejto štruktúry), $\circ_i$ sú operácie na U a ρ_j relácie na U, k alebo m môžu byť rovné 0, nie však súčasne.

Nech (U, π) je algebraická štruktúra, kde π je n-árna operácia na U. X nech je množina premenných a $f : X \longrightarrow U$ nejaké zobrazenie. Priblížime pojem rekurzie. **Rekurzívne** budeme definovať asociované zobrazenie F. Bude to zobrazenie z termov nad algebraickou štruktúrou (U, π) do U. Najprv pojem **termu** nad (U, π):

- každá premená z X je term,
- ak $t_1, \ldots, t_n$ sú termy a π je n-árna operácia, potom $\pi(t_1, \ldots, t_n)$ je term.

Konštrukcia asociovaného zobrazenia:

- $F(x) = f(x)$ pre $x \in X$,
- $F(\pi(t_1, \ldots, t_n)) = \pi(F(t_1), \ldots, F(t_n))$.

Tento typ konštrukcie sa nazýva **rekurziou**: rekurzívne vyhodnocujeme hodnotu funkcie aplikovanej na nejaký bohato štrukturovaný term tak, že postupne (a paralelne) aplikujeme funkciu na stále jednoduchšie komponenty tohto termu, kým nenarazíme na elementárne komponenty.

Teraz zavedieme pojem ordinálu, využívaný v jadre tohto textu (napríklad v kapitole 6).

Majme algebraické štruktúry (M, ρ), (N, σ), kde M, N sú množiny a ρ, σ relácie (na M resp. N). Zobrazenie $f : M \longrightarrow N$ také, že pre $(x, y) \in \rho$ platí $(f(x), f(y)) \in \sigma$ sa nazýva **morfizmom**. Ak f je bijekcia, nazývame ho izomorfizmom. Značenie: $(M, \rho) \cong (N, \sigma)$. Hovoríme, že (M, ρ) a (N, σ) sú izomorfné.

Budeme hovoriť, že dve izomorfné algebraické štruktúry majú rovnaký **typ** a budeme predpokladať existenciu nejakého objektu, ktorý sa nazýva typom algebraickej štruktúry. Ak máme izomorfné algebraické štruktúry (M, ρ), (N, σ), kde ρ a σ sú totálne usporiadania, hovoríme že obe štruktúry majú rovnaký **ordinálny typ** (a zodpovedajúci objekt nazveme ordinálnym typom). Zvláštnu pozornosť venujme prípadu, keď ρ a σ sú dobré usporiadania.

Tvrdenie E.1 *Dve konečné dobre usporiadané množiny majú rovnaký ordinálny typ práve vtedy, keď majú rovnakú mohutnosť.*

Nech M je množina dobre usporiadaná reláciou ρ. Nech $N \subset M$ má nasledovnú vlastnosť: ak $x \in N$ a $(y, x) \in \rho$, potom $y \in N$. Vtedy budeme hovoriť, že N je **úsekom** množiny M.

Tvrdenie E.2 *Nech $(M, \rho) \cong (N, \sigma)$, kde ρ, σ sú dobré usporiadania. Potom platí: buď (M, ρ) a (N, σ) majú rovnaký ordinálny typ alebo existuje úsek $M' \subset M$ taký, že (M', ρ) a (N, σ) majú rovnaký ordinálny typ alebo existuje úsek $N' \subset N$ taký, že (M, ρ) a (N', σ) majú rovnaký ordinálny typ.*

Dôsledok: Označme ordinálny typ štruktúry (M, ρ) ako $t(M, \rho)$. Po zavedení vhodných (a zrejmých) definícií dostávame, že platí buď $t(M, \rho) = t(N, \sigma)$ alebo $t(M, \rho) < t(N, \sigma)$ alebo $t(M, \rho) > t(N, \sigma)$. Práve táto vlastnosť umožňuje definovať nad dobre usporiadanými množinami ordinály a aritmetiku ordinálov.

Ordinálom (alebo ordinálnym číslom) budeme rozumieť ordinálny typ dobre usporiadanej množiny (presnejšie ordinálny typ štruktúry (M, ρ), kde ρ je dobrým usporiadaním množiny M).

Zavedieme túto reprezentáciu ordinálov (reprezentácia pochádza od von Neumanna): Najmenším ordinálom je 0, definuje sa ako ordinálny typ prázdnej množiny (ak chceme pedantnejšie, je to ordinálny typ algebraickej štruktúry $(\emptyset, \rho)$, kde $\rho \subseteq \emptyset^2$). Nasledujú ordinály

- $1 = \{\emptyset\} = \{0\}$ (teda ordinál 1 je množina obsahujúca ordinál 0),
- $2 = \{0, 1\} = \{0, \{0\}\}$,
- pre každé k platí $k + 1 = k \cup \{k\}$.

Takto zavedené ordinály nazývame **nasledovníky**.

Ordinál α je **izolovaný**, ak existuje ordinál β taký, že $\beta < \alpha$ a neexistuje žiadny ordinál medzi α a β. Nasledovníky sú izolované. Nenulové ordinály, ktoré nie sú izolované sa nazývajú **limitné**. Limitné ordinály nie sú nasledovníkmi žiadnych ordinálov. Napríklad typ (dobre usporiadanej množiny) prirodzených čísel ω nie je nasledovníkom žiadneho ordinálu:

$$\omega = \bigcup_{k=0}^{\infty} k = \{0, 1, 2, \ldots, k, \ldots\}.$$

Nad limitnými ordinálmi môžeme konštruovať ďalšie nasledovníky: $\omega \cup \{\omega\} = \omega + 1$ atď. Tak dosiahneme ďalší limitný ordinál (a potom ďalšie a ďalšie). Limitné ordinály budeme používať ako prostriedok formalizácie (zápisu) iteratívnych procesov. Procesy iterácie možno takto charakterizovať vďaka dobrému usporiadaniu, ktoré sa pri ordináloch predpokladá.

Orientovaný graf je algebraická štruktúra (N, E), kde N je množina **vrcholov** a $E \subseteq N^2$ je množina **hrán** (keďže prvky relácie E sú usporiadané dvojice, hrany možno považovať za orientované). Ak E je symetrická relácia, hovoríme jednoducho o **grafe** (môžeme použiť aj prívlastok neorientovaný). *Terminál* je taký vrchol x, že neexistuje hrana (x, y). Ostatné vrcholy sú *neterminály*. *Cesta* je taká postupnosť vrcholov $\langle u_0, \ldots, u_n \rangle$, že pre každé $i < n$ je (u_i, u_{i+1}) hrana. Graf je *acyklický*, ak neexistuje cesta $\langle u, \ldots, u \rangle$ pre žiadny vrchol u.

Strom je taký graf, ktorý má presne jeden vrchol, zvaný koreň, do ktorého nevchádza žiadna hrana (ak r je koreňom stromu, neexistuje hrana (x, r)). Do všetkých ostatných vrcholov vchádza presne jedna hrana, t.j. pre každý vrchol x (rôzny od koreňa) existuje jediný vrchol y taký, že $(y, x) \in E$. To znamená, že strom je acyklický a *súvislý* (každý vrchol leží na nejakej ceste z vrcholu).

Ľubovoľnú množinu symbolov môžeme nazvať *abecedou*. Konečnú postupnosť symbolov danej abecedy nazývame *reťazcom* (nad danou abecedou). Nad konečnou abecedou existuje spočítateľná množina konečných slov. Na grafoch môžeme definovať **ohodnotenie** – funkciu, ktorá vrcholom alebo hranám grafu priradí reťazce nad nejakou abecedou. (V takom prípade hovoríme o **ohodnotenom** grafe.)

Dodatok F

Zápisy algoritmov

Tento dodatok zavádza konvencie a notáciu, používanú pri zápise algoritmov. V texte sú algoritmy zapisované intuitívne a neformálne (prípadne poloformálne). Napriek tomu používajú niektoré bežné konštrukty, ktoré možno pre niektorých čitateľov vyžadujú krátke vysvetlenie (považujme ho za špecifikáciu poloformálneho „programovacieho" jazyka BORNEO :-).

Pri pohľade „zhora" algoritmus dostáva nejaké vstupy a produkuje nejaké výstupy. Vstupmi a výstupmi môžu byť čísla, texty, rozhodnutie o pravdivosti, ale aj zložitejšie štruktúry ako defaultová teória alebo jej extenzia.

Na detailnejšej úrovni, algoritmus predpisuje nejaké (elementárne) akcie. Tie sa vykonávajú buď za každej situácie alebo vtedy, keď sú splnené nejaké podmienky. Jazyk, v ktorom píšeme algoritmy má tri druhy výrazov: výrazy, ktorými zapisujeme

- akcie,
- podmienky (situácie, stavy, v ktorých sa akcie vykonávajú),
- konštruktory, pomocou ktorých (z elementárnych akcií) konštruujeme (zložitejšie) akcie.

Výrazy, ktorými zapisujeme akcie, sa volajú príkazy.

Elementárnosť akcií závisí od daného jazyka (a od zariadenia, ktoré má akcie vykonávať). Tu budeme dosť „hrubozrnní": Za elementárnu akciu budeme napríklad považovať výber nejakého prvku z nejakej množiny. Poloformálny jazyk, ktorý používame, pripúšťa dokonca dosť komplikované elementárne akcie. Napríklad test, či všetky klauzy z nejakej množiny sú splnené pri nejakej interpretácii alebo konštrukcia množiny defaultových pravidiel D_i:
$D_i := \{\alpha : \beta/\gamma \in D \mid h_i \models \alpha,\ h_i \not\models \neg\beta,\ H_j \not\models \neg\beta\}$ sa v tomto texte považuje za elementárnu akciu.

Na začiatku zápisu algoritmu sú v tomto texte vždy špecifikované jeho vstupy a výstupy, prípadne aj požadované správanie.

Používame tieto konštrukcie:

Priradenie.
Zápis má tvar *Premenná* := *hodnota*, napríklad $j := 0$. Po tejto akcii je hodnotou premennej j číslo 0.

Kompozícia.
Ak $A_1, \ldots, A_k$ sú akcie, potom $A_1; A_2; \ldots; A_k$ je akcia. Podobne **begin** $A_1; A_2; \ldots; A_k$ **end** je akcia.

Niekedy budeme nejaké bloky programu, prípadne celú akčnú časť programu, ohraničovať konštruktormi **begin** a **end**.

Cykly.

repeat – until Zápis má tvar **repeat** BLOK **until** PODMIENKA. Tento zápis predpisuje opakovanie série (bloku) akcií, ktoré tu „zakrýva" symbol BLOK. Opakuje sa dovtedy, kým nie je splnená PODMIENKA.

while – do Zápis má tvar **while** PODMIENKA **do** BLOK. Kým je splnená PODMIENKA, vykonávame blok akcií BLOK.

for Príklad použitia: **for** $i = 1$ **to** k **do** BLOK. Zamýšľaný význam – BLOK je nejaký blok akcií, v jeho špecifikácii figuruje premenná i. V priebehu cyklu sa hodnota premennej i postupne zvyšuje od 1 po n (obvykle pripočítame 1 k danej hodnote i) a pre novú hodnotu i sa opakuje BLOK.

Iný príklad použitia: **for** $i = k$ **downto** 2 **do** BLOK. Zamýšľaný význam rovnaký ako vyššie, iba hodnota premennej i sa postupne znižuje (od k po 2); počiatočnú, koncovú hodnotu a krok zvyšovania či znižovania možno voliť ľubovoľne.

každý Používame aj úplne neformálny zápis cyklu, približne (v rôznych variantoch) vyjadrujúci, že pre každý prvok nejakej množiny sa vykoná nejaký blok akcií.

loop Niekedy budeme pred blok akcií, ktorý treba vykonávať opakovane, písať **loop** a na záver bloku **end loop**.

Vetvenie.
Zápis má tvar **if** PODMIENKA **then** AKCIA1 (**else** AKCIA2). Ak je splnená podmienka PODMIENKA, vykoná sa akcia AKCIA1, v opačnom prípade sa vykoná akcia AKCIA2. **else**-časť konštrukcie nebudeme používať vždy. V takom prípade AKCIA2 je žiadna akcia (keď nie je splnená podmienka, nevykoná sa nič).

Niekedy používame ešte menej formálne zápisy ako tu.

Podprogramy.
Pomocou konštruktora **procedure** budeme pomenuvávať nejaký algoritmus.

Zápis má tvar **procedure** MENO. Algoritmus s týmto menom sa vykoná ako súčasť iného algoritmu vtedy, keď sa v inom algoritme vyskytne MENO. Niekedy procedúra MENO môže mať nejaké parametre (premenné) – napríklad MENO(X, Y). V takom prípade v tele iného algoritmu môže byť zápis MENO(3, D), čo znamená, že pri volaní procedúry MENO premenné X a Y nadobúdajú vstupné hodnoty 3 a D.

To isté, čo platí pre **procedure**, platí aj pre **function**, ibaže **function** – keďže je riadna funkcia – vracia jedinú hodnotu (hodnotu funkcie).

Rekurzia.
V tele procedúry (alebo funkcie) MENO môže byť volanie seba samého. Teda meno MENO (prípadne s nejakými argumentami) sa môže vyskytovať v tele procedúry (funkcie, algoritmu) MENO.

Ukončenie činnosti. Ak v nejakom bode zápisu algoritmu je možné zastaviť jeho činnosť, zaznamenáme to pomocou **return** alebo pomocou **return** HODNOTA. Druhý z uvedených zápisov znamená, že výsledkom činnosti algoritmu je hodnota HODNOTA.

Komentáre. Komentáre vyznačujeme takto: (* *komentár* *), teda text komentára je medzi zátvorkami s hviezdičkou. Text takto označený nie je súčasťou algoritmu, ale iba pomocným vysvetlením.

Dodatok G

Prehľadávanie, backtracking

Mnoho úloh umelej inteligencie sa výpočtovo rieši tak, že sa prehľadáva priestor možných riešení a každý bod tohto priestoru sa testuje, či spĺňa požadované vlastnosti. Priestor sa dá prehľadávať podľa rôznych stratégií. Ak je štrukturovaný ako strom, môžeme ho prehľadávať buď do hĺbky alebo do šírky, môžeme využívať rôzne heuristiky.

Aj v tomto texte, niekedy explicitne a veľmi často implicitne sme predpokladali nejaký priestor možných riešení nejakého problému a prehľadávanie tohto priestoru ako metódu nachádzania korektných riešení (extenzie defaultových teórií, abduktívne vysvetlenia, stabilné modely atď).

Príklad G.1 Pripomeňme si problém ôsmich dám (príklad 4.152 z časti 4.3). Ide o to, rozostaviť na šachovnici osem dám tak, aby sa vzájomne neohrozovali.

Veľmi priamočiary spôsob spočíva v postupnom preberaní všetkých poradí čísel $1, \ldots, 8$ a pre každé poradie preverovať, či sú splnené podmienky nekonfliktnej konfigurácie. Toto je metóda generuj a testuj: vygeneruje sa kandidát na riešenie a potom sa testuje, či je skutočne riešením problému.

Jemnejšie metódy prelínajú fázu generovania a testovania: Napríklad, po položení i-tej dámy na šachovnicu hneď preveruj, či sa položených i dám vzájomne neohrozuje. Za predpokladu, že predchádzajúcich $i-1$ dám sa nehohrozovalo a i sa už ohrozuje, polož i-tu dámu na iné pole šachovnice. Ak nenájdeš také pole, na ktoré by sa dala i-ta dáma nekonfliktne umiestniť, vráť sa k $i-1$. dáme a ulož ju na iné miesto ako bola uložená pôvodne. Tento návrat sa nazýva **backtracking**.[1] ■

V tabuľke G.1 je schéma backtrackingu. Túto schému možno rozšíriť tak, aby postupne hľadala všetky možné riešenia.

[1] Prijatý slovenský termín je *prehľadávanie s návratom.*

Algoritmus G.2 *(* predpokladáme, že po jednom pridávame nejaké prvky k množine, ktorá bude predstavovať potenciálne riešenie *)*
  inicializuj (vyber prvý prvok do potenciálneho riešenia *)*
  repeat
    vyber ďalší prvok do potenciálneho riešenia
    if *prijateľný (* test podmienky na príslušnosť k riešeniu *)*
      then *zaznamenaj ho*
      else *zruš ho, v prípade potreby aj jeho predchodcu*
  until *riešenie je kompletné*

Tabuľka G.1: Schéma backtrackingu

Dodatok H

Vypočítateľnosť a zložitosť

Tento dodatok rozsahom podstatne prevyšuje ostatné. Hlavným dôvodom je potreba aspoň trochu kompenzovať fakt, že otázky, súvisiace s vypočítateľnosťou a výpočtovou zložitosťou neboli v poprední našej pozornosti. Uvádzali sme informácie o výpočtových a zložitostných vlastnostiach niektorých typov hypotetického usudzovania, ale nešli sme do detailov. Preto čitateľ bez spoľahlivého poznatkového pozadia (akým môže byť napríklad štúdium informatiky na Matematicko-fyzikálnej fakulte) pravdepodobne bude s týmito informáciami spájať minimálny obsah. Čitateľ, ktorý má spomínané pozadie, nemusí túto časť čítať (azda jedinou výnimkou sú časti o polynomickej a aritmetickej hierarchii).

Sústredíme sa predovšetkým na zavedenie a vysvetlenie pojmov a faktov, ktoré boli použité v jadre textu. Okrem toho sa zameriame na charakterizáciu problémov, ktoré sú výpočtovo ťažké. Dôvod je priamočiary – problémy umelej inteligencie tvoria klasickú zásobáreň nevypočítateľných alebo výpočtovo ťažkých problémov.

Naším cieľom je oboznámiť sa s pojmom algoritmu, zhrnúť základné informácie o tom, aké problémy sú či nie sú riešiteľné algoritmicky a u tých riešiteľných získať predstavu o ich náročnosti z hľadiska výpočtového.

V prvej časti si budeme všímať zásadný problém: či každý problém možno riešiť nejakým algoritmom. V druhej časti sa budeme venovať efektívnosti algoritmov. Prvé dve časti uvedú dva druhy problémov, ktoré sú z výpočtového hľadiska ťažké. Prvý druh: neexistuje algoritmus, ktorý ich rieši. Druhý druh: existuje taký algoritmus, ale nie je schopný nájsť riešenie v rozumnom čase. V tretej časti si všimneme problémy, ktoré sú „ťažšie" ako „najľahšie" z práve spomenutých dvoch typov ťažkých problémov.

H.1 Vypočítateľnosť

Budeme používať intuitívny pojem algoritmu (podobne ako Shoenfield, pozri [Sho 71]).[1] **Algoritmom** budeme rozumieť pravidlo (návod na akciu, konanie), ktoré

- možno zapísať konečným jazykovým výrazom,
- pozostáva z volania elementárnych akcií,
- elementárne akcie možno vykonať mechanicky a ich výsledok je determinovaný vstupmi,
- volanie elementárnych akcií prebieha v diskrétnych krokoch: v kroku i sa spracuje výsledok elementárnych akcií z krokov predchádzajúcich i,
- jednoznačne možno rozhodnúť, kedy je akcia ukončená a s akým výsledkom,
- je použiteľné na riešenie celej triedy problémov.

Teraz príklad algoritmu. Predstavme si, že vyberáme predsedu parlamentu a chceme vybrať toho najnižšieho zo všetkých uchádzačov:

1. Uchádzačov náhodne usporiadame.
2. Predbežne považujeme prvého uchádzača (z náhodného usporiadania) za najmenšieho.
3. Od druhého po posledného uchádzača z tohto náhodného usporiadania opakujeme:
    - ak daný uchádzač je menší ako predbežne najmenší, považujeme ho za predbežne najmenšieho.
4. Keď sme prešli všetkých uchádzačov, predbežne najmenšieho uchádzača môžeme považovať za najmenšieho (a teda za predsedu parlamentu).

Existuje mnoho formalizácií pojmu algoritmu (rekurzívne funkcie, Turingove stroje, λ-definovateľné funkcie, Markovove normálne algoritmy, Postove prepisovacie systémy, RAM atď.). Tieto formalizácie sú vzájomne ekvivalentné. O ich vzťahu k intuitívnemu pojmu algoritmu hovorí Church-Turingova hypotéza: *Nech F je formalizácia intuitívneho pojmu algoritmu. Potom pre každý algoritmus existuje formálny objekt definovaný formalizáciou F, ktorý realizuje tento algoritmus.* Keďže obsahom hypotézy je korešpondencia intuitívneho a formálneho pojmu, nemožno ju dokázať. Nie je však známy

[1] Pre účely tohto dodatku je toto používanie úplne postačujúce. Záujemcov o podrobnejšie informácie odkazujeme napríklad na [Pap 93], [GaJ 79], [Gru 97], [HoU 69], [Mal 65], [Sho 71]. Rozsiahlejší neformálny výklad možno nájsť v [Har 87]. Tento dodatok je závislý na práve citovaných zdrojoch.

príklad intuitívne akceptovateľného algoritmu, ktorý by nebolo možné upresniť každým zo známych spôsobov. Napokon aj existencia viacerých – vzájomne ekvivalentných – formalizácií pojmu algoritmu ukazuje, že táto hypotéza je veľmi vieryhodná.

Pojem algoritmu teraz použijeme na opísanie intuície (čiastočne) vypočítateľnej funkcie. Budeme sa zaujímať iba o funkcie definované na prirodzených číslach. Prijmeme, že každý vstup a každý výstup ľubovoľného algoritmu možno kódovať prirodzenými číslami. Pre čitateľa to nie je nič prekvapujúceho. Dobre vie, že všetky algoritmy, všetky vstupy a výstupy možno kódovať pomocou $0, 1$. Na pochopenie vlastností algoritmov preto stačí skúmať funkcie z prirodzených čísel do prirodzených čísel.[2]

Nech teda *Nat* je množina všetkých prirodzených čísel. Hovoríme, že algoritmus $\mathcal{A}$ počíta funkciu $f : Nat \longrightarrow Nat$ vtedy, keď pre každý vstup n vypočíta $f(n)$. Funkciu, ktorú počíta nejaký algoritmus, nazveme **vypočítateľnou funkciou**. Ak f je čiastočná funkcia, hovoríme, že algoritmus $\mathcal{A}$ počíta čiastočnú funkciu, keď pre vstup n vypočíta $f(n)$, ak je $f(n)$ definované. Čiastočnú funkciu, ktorú počíta nejaký algoritmus, nazveme **čiastočne vypočítateľnou funkciou**.

Ak algoritmus počíta čiastočnú funkciu, potom pre tie vstupy, kde čiastočná funkcia nie je definovaná, nedá nijaký výstup (jeho výpočet sa nezastaví). Niekedy sa rozlišuje medzi algoritmom a procedúrou, napríklad v [HoU 69]. Algoritmus sa v takom prípade chápe ako procedúra, ktorá pre každý vstup ukončí svoju činnosť. Napriek tomu, že budeme používať obidva termíny, uvedené rozlíšenie nebudeme robiť a pripustíme, že algoritmus nemusí ukončiť pre niektoré vstupy svoju činnosť. V takom prípade algoritmus počíta nejakú čiastočnú funkciu.

Čitateľ môže obskočiť definície H.1 a H.2, ak sa uspokojí

- s konštatovaním, že v teórii rekurzívnych funkcií sa pojem algoritmu formalizuje pomocou pojmu čiastočne rekurzívnej funkcie,
- s intuíciou, že pojem čiastočne rekurzívnej funkcie zodpovedá predstave o čiastočne vypočítateľnej funkcii.

Intuitívne, čiastočná funkcia je **čiastočne rekurzívna** vtedy, keď existuje algoritmus, ktorý ju počíta. Teda – podľa Church-Turingovej hypotézy – pojem čiastočne rekurzívnej funkcie presne formalizuje pojem algoritmu.

Definícia H.1 Triedu funkcií $I^n_m(x_1, \ldots, x_n) = x_m$ nazývame **projekciami**. Ďalej predpokladáme funkcie $0^n(x_1, \ldots, x_n) = 0$ a funkciu **nasledovníka** $s(x) = x + 1$.

Ak sú dané funkcie

$$f^m_1(x_1, \ldots, x_m), \ldots, f^m_n(x_1, \ldots, x_m) \text{ a } f^n(x_1, \ldots, x_n),$$

[2] Existuje oveľa viac funkcií na prirodzených číslach, ktoré nepočíta žiadny algorimus, než tých, ktoré nejaký algoritmus počíta.

hovoríme, že funkciu

$$g^m(x_1,\ldots,x_m) = f^n(f_1^m(x_1,\ldots,x_m),\ldots,f_n^m(x_1,\ldots,x_m))$$

sme dostali operáciou **skladania** z funkcií $f^n, f_1^m, \ldots, f_n^m$.

Čiastočná funkcia f vzniká z čiastočných funkcií g, h operáciou **primitívnej rekurzie**, ak platí:

$$\begin{aligned} f(0,x_2,\ldots,x_n) &= h(x_1,\ldots,x_n), \\ f(n+1,x_2,\ldots,x_n) &= g(n,f(n,x_2,\ldots,x_n),x_1,\ldots,x_n). \end{aligned}$$

Primitívna rekurzívna funkcia je každá taká funkcia, ktorú skonštruujeme z funkcií 0, nasledovníka a projekcie konečným počtom aplikácii operácií skladania funkcií a primitívnej rekurzie.

■

Definícia H.2 Výraz $\mu_y(g(y,x_1,\ldots,x_n) = 0$ označuje najmenšiu hodnotu y, ktorá spĺňa rovnicu $g(y,x_1,\ldots,x_n) = 0$. (Čiastočná) funkcia g vzniká z (čiastočnej) funkcie f operáciou minimalizácie, ak platí $g(x_1,\ldots,x_n) = \mu_y(f(y,x_1,\ldots,x_n) = 0)$.[3]

(Čiastočná) funkcia f je **(čiastočná) rekurzívna funkcia**, ak ju možno získať z funkcií projekcie, 0 a nasledovníka konečným počtom operácií skladania, primitívnej rekurzie a minimalizácie. ■

Teraz si utvoríme predstavu o konštruktívne definovateľnej množine. Nech M je množina. Jej **charakteristická funkcia** χ_M je definovaná tak, že pre každé $x \in M$ je $\chi_M(x) = 1$ a pre každé $x \notin M$ je $\chi_M(x) = 0$. M je **rekurzívna množina** práve vtedy, keď jej charakteristická funkcia je rekurzívna. (Teda, existuje algoritmus $\mathcal{A}$, ktorý počíta χ_M a $\mathcal{A}$ dokáže rozhodnúť, či $x \in M$ alebo $x \notin M$.) **Čiastočná charakteristická funkcia** je taká funkcia χ, ktorá pre každé $x \in M$ dáva $\chi_M(x) = 1$, inak je nedefinovaná. **Množina** je **čiastočne rekurzívna**, ak jej čiastočná charakteristická funkcia je čiastočne rekurzívna.

Samozrejme, algoritmy používame na riešenie problémov (a vždy privítame, keď pre nejaký problém nájdeme algoritmus, ktorý ho rieši). Začnime sa teda zaujímať o súvislosť problémov a algoritmov.

Budeme si spočiatku všímať iba rozhodovacie problémy. Rozhodovaciemu problému zodpovedá otázka *je ϕ pravdivá?* kde ϕ je nejaká veta (nejakého jazyka). Je zrejmé, že ostatné druhy problémov možno redukovať na rozhodovacie. Každý problém má triedu inštancií. Napríklad, ak ϕ vo vyššie uvedenej otázke sú: 2 je prvočíslo, 8 je prvočíslo, 3 je prvočíslo, 33 je prvočíslo atď., hovoríme, že všetko to sú inštancie vetnej formy *X je prvočíslo*. Táto vetná forma reprezentuje celú triedu inštancií. Algoritmus, ktorý rozhoduje

---

[3] g môže byť čiastočnou funkciou aj vtedy, keď f je totálna: môže sa stať, že $f(y,x_1,\ldots,x_n) = 0$ nemá žiadne riešenie y.

problém, či nejaké číslo je prvočíslo, funguje pre celú triedu vstupov. Pre každý zo vstupov dáva odpovede „áno“ alebo „nie“.

Ďalej budeme predpokladať, že každý algoritmus $\mathcal{A}$, ktorý rieši rozhodovací problém, je korektný. To znamená, že vždy, keď $\mathcal{A}$ dá na otázku

$$\textit{je } \phi(X_1,\ldots,X_n)) \textit{ pravdivé?}$$

odpoveď áno (nie) pre inštanciu $(t_1,\ldots,t_n)$, potom skutočne $\phi(t_1,\ldots,t_n)$ je pravdivé (nepravdivé).

Korektnosť znamená, že algoritmu môžeme veriť, ak dá nejakú odpoveď. Teraz je už dôležité iba to, či dáva odpoveď. Máme na mysli, či pre každý problém možno nájsť algoritmus, ktorý ho rieši. Teda, či je uskutočniteľná ideálna predstava, že pre ľubovoľný problém vždy existuje algoritmus, ktorý by ho riešil. Žiaľ (alebo našťastie?), odpoveď je negatívna.

Prejdime teda k nerozhodnuteľným problémom. Treba zdôrazniť, že sa hovorí o probléme, nie o algoritme. To znamená – ak je problém nerozhodnuteľný, nie je to záležitosť nejakého konkrétneho algoritmu, ale je to vlastnosť problému. Tu je vhodné poznamenať, že hlboké výsledky, charakterizujúce algoritmy, je možné dosiahnuť až vtedy, keď sa pojem algoritmu nejakým spôsobom upresní. Bez presného pojmu algoritmu nie je možné napríklad dokázať neexistenciu algoritmu riešiaceho nejaký problém.

Budeme hovoriť, že algoritmus $\mathcal{A}$ **rozhoduje** problém *je* $\phi(X_1,\ldots,X_n)$ *pravdivé?* vtedy, keď dáva na každú inštanciu tejto otázky buď odpoveď „áno“ alebo odpoveď „nie“. Problém je **rozhodnuteľný**, ak existuje algoritmus, ktorý ho rozhoduje. Problém je **nerozhodnuteľný**, ak neexistuje algoritmus, ktorý ho rozhoduje.

Existujú nerozhodnuteľné problémy, napríklad:

- Je ϕ teorémou predikátového počtu prvého rádu? (Tento problém si budeme označovať ako TEORÉMY; nerozhodnuteľné sú aj ekvivalentné problémy: problémy platnosti, odvoditeľnosti, vyplývania, konzistentnosti, pozri dodatok B.)

- Zastaví algoritmus $\mathcal{A}$ pre vstup x? ($\mathcal{A}$ je ľubovoľný algoritmus, x jeho ľubovoľný vstup, tento problém si budeme označovať ako ZASTAVENIE.)

Ukážeme si hlavnú myšlienku dôkazu, že problém ZASTAVENIE je skutočne nerozhodnuteľný. To, že algoritmus $\mathcal{A}$ zastaví pre vstup x, si označíme ako $x \in Z_{\mathcal{A}}$ (x patrí medzi hodnoty, pre ktoré $\mathcal{A}$ zastaví). Uvedomme si, že každý algoritmus je konečný objekt (konečný zápis nad konečnou abecedou). Existuje teda spočítateľne mnoho algoritmov. Predstavme si postupnosť $\{\mathcal{A}_n\}$ všetkých algoritmov. Predpokladajme teraz, že existuje algoritmus $\mathcal{B}$, ktorý rozhoduje problém ZASTAVENIE. Pre ľubovoľnú dvojicu $(x,\mathcal{A})$, kde $\mathcal{A}$ je nejaký algoritmus a x jeho vstup, teda $\mathcal{B}$ zistí, či $x \in Z_{\mathcal{A}}$ alebo či $x \notin Z_{\mathcal{A}}$.

Konštruujme algoritmus $\mathcal{N}$. Pre každé prirodzené číslo n na vstupe počíta takto: Volá $\mathcal{B}$. Ak $\mathcal{B}$ zistí, že $n \in Z_{\mathcal{A}_n}$, tak $\mathcal{N}(n) = \mathcal{A}_n(n) + 1$. To znamená, že N sa líši od každého algoritmu $\mathcal{A}_n$, ktorý zastaví pre vstup n – vypočíta hodnotu o 1 väčšiu. V opačnom prípade, ak $\mathcal{B}$ zistí, že $n \notin Z_{\mathcal{A}_n}$, tak $\mathcal{N}(n) = 0$. To znamená, že $\mathcal{N}$ sa líši aj od každého algoritmu $\mathcal{A}_n$, ktorý nezastaví pre vstup n.

Tento algoritmus sa teda líši od každého algoritmu z postupnosti všetkých algoritmov $\mathcal{A}_n$. Preto $\mathcal{N}$ nie je algoritmus. To znamená, že nemôže existovať ani predpokladaný algoritmus $\mathcal{B}$, ktorý rozhoduje vzťah $x \in Z_{\mathcal{A}}$. Totižto, na predpoklade existencie tohto algoritmu sme skonštruovali $\mathcal{N}$.

Nerozhodnuteľnosť problému TEORÉMY možno dokázať redukciou na problém ZASTAVENIE, pozri [BoJ 80]. Technika redukcie patrí do základnej výbavy uvažovania o algoritmoch. Ak nájdeme postup, ako výpočet algoritmu $\mathcal{A}$ redukovať na výpočet algoritmu $\mathcal{B}$ (rozhodnutie, ktoré má dať $\mathcal{A}$, dá po tej redukcii $\mathcal{B}$), potom môžeme $\mathcal{A}$ charakterizovať pomocou toho, čo vieme o $\mathcal{B}$. Ukážeme, že platí: ak by bol rozhodnuteľný problém TEORÉMY, musel by byť rozhodnuteľný aj problém ZASTAVENIE.

Vtip konštrukcie je takýto: V prvorádovom jazyku sa dajú efektívne skonštruovať konečná množina viet Φ a veta ϕ také, že $\Phi \models \phi$ práve vtedy, keď daný algoritmus $\mathcal{A}$ zastaví pre daný vstup x. (Nejde o nič iného, ako o opis algoritmu $\mathcal{A}$ v jazyku logiky.) Ak by vzťah $\models$ (t.j. aj problém TEORÉMY) bol rozhodnuteľný, potom by musel byť rozhodnuteľný aj problém ZASTAVENIE.

Pozrime sa pozornejšie na naše príklady nerozhodnuteľných problémov.

Príklad H.3 Všetky dôkazy teorém predikátového počtu prvého rádu sú konečné objekty. Možno mechanicky zistiť (nahliadnutím na tvar výrazov v dôkaze a preverením niekoľkých odvodzovacích pravidiel), či daná konečná postupnosť jazykových výrazov je dôkaz.

Teda, ak je (formálny) dôkaz skonštruovaný, môžeme ho (na základe nejakého algoritmu) overiť. ■

Môžeme však viac ako identifikovať hotové dôkazy. V dodatku D sme sa stručne oboznámili so základmi automatického dokazovania prvorádových teorém. Rezolvenčná metóda poskytuje algoritmus, ktorý – veľmi voľne – charakterizuje tabuľka H.1. Ak ϕ je dokázateľná formula predikátového počtu prvého rádu, potom tento algoritmus skonštruuje rezolvenčný dôkaz pre ϕ. (Je ním postupný výpis hodnôt premennej Φ'.) Pre ostatné formuly však nie je záruka, že procedúra (rezolvenčného dokazovania) zastaví svoju činnosť.

Príklad H.5 Predstavme si algoritmus $\mathcal{B}$, ktorý na vstupe dostane dvojicu $(x, \mathcal{A})$ a jeho činnosť spočíva v tom, že aplikuje $\mathcal{A}$ na vstup x.[4] Ak $\mathcal{A}$ zastaví,

[4] Takto definovaný algoritmus $\mathcal{B}$ sa nazýva univerzálnym algoritmom. Je schopný simulovať ľubovoľný algoritmus. Reálne počítače vlastne počítajú na základe univerzálneho algoritmu (ak odhliadneme od obmedzenia, že pracujú s konečným priestorom a v konečnom čase).

Algoritmus H.4
vstup: *množina kláuz* Φ *a klauza* ϕ
výstup: *dôkaz* ϕ *z* Φ

begin
$\Phi' := \Phi \cup \{\neg\phi\}$ *; vypíš* Φ'
repeat
ak na Φ' *nie je aplikovateľné pravidlo rezolvencie,* **RETURN** neúspech
$\Phi' := r(\Phi')$, *kde* $r(\Phi')$ *je výsledok aplikácie pravidla rezolvencie na* Φ'
vypíš Φ'
until $r(\Phi') = [\,]$, *kde* $[\,]$ *je prázdna klauza*
end

Tabuľka H.1: Algoritmus rezolvovania

konštruovaný algoritmus $\mathcal{B}$ dá odpoveď „áno“. V opačnom prípade beží bez zastavenia.

Teda aj pre problém ZASTAVENIE máme predstavu o procedúre s podobnými vlastnosťami ako pre problém TEORÉMY: nekončiaci beh pre nejaké vstupy. ■

Príklady naznačujú, že bude užitočné všímať si algoritmy, ktoré svoju činnosť nezastavia pre niektorú vstupnú hodnotu.

Algoritmus, ktorý **polo**rozhoduje problém *je* $\phi(X_1, \ldots, X_n)$ *pravdivé?*, dáva na všetky jeho inštancie jednu z dvoch odpovedí: „áno“, „nie“ alebo nedáva žiadnu odpoveď (nezastaví svoju činnosť). Problém je **polo**rozhodnuteľný, ak existuje algoritmus, ktorý ho polorozhoduje. Videli sme už, že problémy TEORÉMY a ZASTAVENIE sú polorozhodnuteľné.

Vidíme, že jeden zo základných problémov, súvisiacich s usudzovaním, je polorozhodnuteľný. Mnohé problémy, súvisiace s hypotetickým usudzovaním, nie sú ani polorozhodnuteľné, pozri napríklad kapitoly 3, 4, 5 alebo 7.1. Preto bude zaujímavé a dôležité venovať sa polorozhodnuteľným a ťažším problémom.

Najprv však terminologická poznámka. Kvôli neformálnemu prístupu v tomto dodatku bude niekedy naša terminológia „kĺzavá“: Pre danú vlastnosť V budeme striedavo hovoriť, že nejaké problémy, množiny, algoritmy a/alebo funkcie majú túto vlastnosť V.

V tejto chvíli zdôvodníme toto „kĺzanie“ pre problémy a množiny. Problém môžeme reprezentovať ako množinu. Každá vlastnosť sa totiž dá vymedziť množinou objektov, ktoré danú vlastnosť majú. Ak teda problémom je zistiť, či daný objekt má nejakú vlastnosť, stačí zistiť, či patrí do zodpovedajúcej množiny. Teda, problém môžeme reprezentovať množinou jeho riešení.

Charakterizujme si teraz bližšie tie algoritmy, ktoré polorozhodujú nejaký problém. Algoritmus $\mathcal{A}$ **akceptuje** množinu M, ak na rozhodovací problém *je pravda, že* $x \in M$? odpovedá „áno“ pre každé $x \in M$, v opačnom prípade nezastaví výpočet. Budeme hovoriť, že množina M je **rozpoznateľná**, ak ju nejaký algoritmus akceptuje. Na základe vyššie uvedenej terminologickej dohody budeme hovoriť, že problém je rozpoznateľný, ak množina jeho inštancií je rozpoznateľná.

Tvrdenie H.6 *Každá rekurzívna množina je rozpoznateľná.*

Idea dôkazu: algoritmus, ktorý rozhoduje M, možno modifikovať tak, že miesto odpovedí „nie“ bude bežať bez zastavenia. ■

Príklad H.7 Vráťme sa k algoritmu H.4. Je známe, že H.4 pre niektoré vstupy nezastaví. Modifikujme ho tak, že nezastaví aj v prípadoch, kedy by mal hlásiť neúspech. Majme na vstupe prázdnu množinu kláuz a ϕ, teorému predikátového počtu. Potom tento modifikovaný algoritmus zastaví, skonštruuje dôkaz a na otázku, či ϕ je teorémou predikátového počtu prvého rádu, odpovedá *áno*. Dostali sme tak algoritmus, akceptujúci množinu teorém predikátového počtu prvého rádu. Teda táto množina je rozpoznateľná. Vieme však, že nie je rekurzívna. ■

Tvrdenie H.8 *Existuje rozpoznateľná množina, ktorá nie je rekurzívna.*

Príklad H.9 Ďalším príkladom rozpoznateľnej a nie-rekurzívnej množiny je množina dvojíc $H = \{(x, \mathcal{A}) : \mathcal{A} \text{ zastaví na } x\}$, kde $\mathcal{A}$ je ľubovoľný algoritmus a x je jeho vstup. Táto množina je rozpoznateľná. V príklade H.5 máme popísaný algoritmus $\mathcal{B}$, ktorý akceptuje H: pre vstup $(x, \mathcal{A})$ treba aplikovať $\mathcal{A}$ na x. Ak $\mathcal{A}$ zastaví, $\mathcal{B}$ odpovedá „áno“, v opačnom prípade nezastaví. Vieme pritom, že H nie je rekurzívna. Symbol H, označujúci množinu dvojíc $(x, \mathcal{A})$ takých, že algoritmus $\mathcal{A}$ zastaví na vstupe x budeme ďalej ešte používať. ■

Skôr než vyslovíme nejaké tvrdenia o vzťahu medzi rekurzívnosťou a rozpoznateľnosťou, malé odbočenie k redukcii problémov a k úplným problémom.

Množina dvojíc H z príkladu H.9 je rozpoznateľná. Dôležité je však jej významné postavenie medzi rozpoznateľnými množinami. Ak by existoval algoritmus rozhodujúci, či $(x, \mathcal{A}) \in H$, potom by každá rozpoznateľná množina bola rekurzívna (problém, či $x \in M$ by bol rozhodnuteľný pre každú rozpoznateľnú množinu M):

Nech $\mathcal{A}$ rozpoznáva množinu M. To znamená, pre $x \in M$ dáva $\mathcal{A}$ odpoveď „áno“, pre $x \notin M$ nezastaví. Predpokladajme, že H je rozhodnuteľná. Teda, existuje algoritmus, ktorý pre každý vstup $(x, \mathcal{A})$ dáva odpoveď „áno“ alebo „nie“ na otázku, či $(x, \mathcal{A}) \in H$. V prvom prípade $\mathcal{A}$ zastaví pri vstupe x, teda $x \in M$. V druhom prípade vieme, že $(x, \mathcal{A}) \notin H$, to znamená, že $\mathcal{A}$ nezastaví pre x, preto $x \notin M$. Teda M by bola rekurzívna, ak by H bola rekurzívna.

Práve sme ilustrovali redukciu problémov a úplnosť problému vzhľadom na nejakú triedu problémov. Intuitívne, problém Π je **redukovateľný** na problém Π' vtedy, keď riešenie problému Π (pre každú jeho inštanciu) môžeme nájsť riešením problému Π'. Videli sme, že každý rozpoznateľný problém je redukovateľný na problém zastavenia. Poznamenajme, že redukcia musí byť vypočítateľná, nazveme ju algoritmickou redukovateľnosťou (skrátene: a-redukovateľnosť). Presnejšie: Množina M sa a-**redukuje** na množinu N, ak existuje rekurzívna funkcia $f : M \longrightarrow N$ taká, že $x \in M$ práve vtedy, keď $f(x) \in N$. (Hovoríme striedavo o problémoch a množinách.)

Ďalej, budeme hovoriť, že problém je **úplný** pre triedu problémov, keď každý problém z danej triedy možno naň redukovať. Úplný problém predstavuje najťažší typ problémov v danej triede. Videli sme, že problém zastavenia je úplný v triede rozpoznateľných problémov. Predpokladajme, že jeden problém úplný v triede rozpoznateľných problémov by bol rozhodnuteľný. Potom by boli rozhodnuteľné všetky problémy z tejto triedy. Spresnenie (pre množiny): Množina M je a-**úplná** množina v triede množín T, ak M patrí do T a každú množinu typu T možno a-redukovať na M.

Môžeme sa vrátiť k vzťahu medzi rekurzívnosťou a rozpoznateľnosťou. Vyjadruje ho nasledujúce tvrdenie.

Tvrdenie H.10

a) *Množina M je rekurzívna práve vtedy, keď M aj jej komplement $\overline{M}$ sú rozpoznateľné.*

b) *Ak M je rekurzívna, potom aj $\overline{M}$ je rekurzívna.*

Idea dôkazu

b): Ak M je rekurzívna, algoritmus rozhodujúci $\overline{M}$ je jednoduchou modifikáciou algoritmu rozhodujúceho M – stačí zameniť každú odpoveď „áno" za „nie" a naopak.

a):

$\Rightarrow$ Ak M je rekurzívna, aj $\overline{M}$ je rekurzívna, teda M a $\overline{M}$ sú rozpoznateľné (podľa tvrdenia H.6).

$\Leftarrow$ Predpokladajme, že M a $\overline{M}$ sú rozpoznateľné (na základe algoritmov $\mathcal{A}$, $\overline{\mathcal{A}}$). Algoritmus $\mathcal{B}$ spracúva vstup x tak, že volá $\mathcal{A}$ pre vstup x a aj $\overline{\mathcal{A}}$ pre vstup x takým spôsobom, že $\mathcal{A}$ urobí prvý krok, potom $\overline{\mathcal{A}}$ urobí prvý krok, potom $\mathcal{A}$ druhý krok atď. Ak $\mathcal{A}$ zastaví, $\mathcal{B}$ odpovedá „áno", ak zastaví $\overline{\mathcal{A}}$, odpovedá „nie". Za každých okolností zastaví presne jeden z volaných algoritmov. ■

Už z definície je zrejmé, že riešenie polorozhodnuteľných problémov (a aj akceptovanie množiny) je asymetrické. Odpoveď „áno" je v kontraste s behom bez zastavenia. Vyjadruje to aj triviálny dôsledok tvrdenia H.10.

Dôsledok H.11 *Množina M nie je rekurzívna práve vtedy, keď M alebo $\overline{M}$ nie sú rozpoznateľné.*

Máme aj príklad nerozpoznateľnej množiny:

Tvrdenie H.12 *Existuje množina, ktorá nie je ani rozpoznateľná.*

Dôkaz: H je rozpoznateľná množina (podľa tvrdenia H.8). Preto $\overline{H}$, komplement množiny H, nie je ropoznateľná množina – keby bola, množina H by bola rekurzívna (podľa tvrdenia H.10). ■

Polorozhodnuteľné (a ťažšie) problémy patria medzi typické problémy umelej inteligencie. Preto sa im ešte budeme venovať. Dopochopeniu pojmov rozpoznateľnej množiny i polorozhodnuteľného problému pomôžu ich alternatívne vyjadrenia.

Najprv o vzťahu rozpoznateľnosti a generovateľnosti. Budeme hovoriť, že algoritmus $\mathcal{A}$ **generuje** množinu M, ak $\mathcal{A}$ nedostane žiadny vstup a na výstupe postupne produkuje prvky M. (M je množina všetkých prvkov, vypísaných počas činnosti algoritmu $\mathcal{A}$.)

Tvrdenie H.13 *M je množina generovaná nejakým algoritmom $\mathcal{A}$ práve vtedy, keď M je rozpoznateľná.*

Dôkaz:
$\Rightarrow$
Konštruujme algoritmus $\mathcal{B}$ akceptujúci M: $\mathcal{B}$ volá algoritmus $\mathcal{A}$; ak $\mathcal{B}$ spracúva vstup $x \in M$, po nejakom počte krokov sa na výstupe algoritmu $\mathcal{A}$ objaví x a $\mathcal{B}$ oznámi „áno“. Algoritmus $\mathcal{A}$ však nedá na výstupe x, ak $x \notin M$, preto $\mathcal{B}$ nezastaví. Teda, $\mathcal{B}$ akceptuje M.
$\Leftarrow$
Nech $\mathcal{B}$ akceptuje M. Konštruujme algoritmus $\mathcal{A}$, ktorý nedostáva žiadny vstup a generuje množinu M. Najprv usporiadajme množinu

$$M = \{x_1, \ldots, x_n, \ldots\}.$$

Pre každé $i = 1, 2, \ldots$ algoritmus $\mathcal{A}$ zavolá algoritmus $\mathcal{B}$ postupne pre prvých i vstupov a pre každý vstup urobí $\mathcal{B}$ nanajvýš i krokov. Ak $\mathcal{B}$ pre niektorý zo vstupov, označme ho x, oznámi „áno“, potom $\mathcal{A}$ vypíše x.

Preverme, že $\mathcal{A}$ robí to, čo má – generuje M: Nech $x \in M$, potom $\mathcal{B}$ zastaví pre vstup x a $\mathcal{A}$ v niektorom kroku vypíše x, v opačnom prípade sa nemôže stať, že by $\mathcal{A}$ vypísal x. ■

Práve uvedený pohľad (generovanie množiny M nejakým algoritmom $\mathcal{A}$) možno vyjadriť ešte takto: $\mathcal{A}$ počíta čiastočne rekurzívnu funkciu f z množiny všetkých prirodzených čísel Nat na M. Pritom $f(n) = m$, ak algoritmus $\mathcal{A}$ vypíše (vygeneruje) v n-tom kroku m. Platí teda: $M = \{m : (\exists n \in Nat)\ f(n) = m\}$. Budeme hovoriť, že $\mathcal{A}$ spočítava[5] množinu M a že M (ako množina hodnôt čiastočne rekurzívnej funkcie) je **rekurzívne spočítateľná**.

[5]Funkcia f spočítava prvky množiny M do postupnosti $f(0), f(1), \ldots$.

Pozrime sa ešte na vzťah vypočítateľnosti funkcií a množín. **Graf funkcie** $f : X_1 \times \cdots \times X_n \longrightarrow Y$ je množina všetkých $n+1$-tíc $(x_1, \ldots, x_n, y)$ takých, že $f(x_1, \ldots, x_n) = y$.

Tvrdenie H.14 *Funkcia je čiastočne rekurzívna práve vtedy, keď jej graf je rozpoznateľný.*

Idea dôkazu:
$\Rightarrow$
Nech algoritmus $\mathcal{A}$ počíta čiastočnú funkciu f. Algoritmus $\mathcal{B}$, akceptujúci jej graf, sa pri vstupe $(x_1, \ldots, x_n, y)$ správa takto: volá $\mathcal{A}$ pre vstup $(x_1, \ldots, x_n)$. Ak $\mathcal{A}$ zastaví a vypočíta y, potom $\mathcal{B}$ zastaví a povie „áno", v opačnom prípade $\mathcal{B}$ nezastaví.
$\Leftarrow$
Nech $\mathcal{B}$ rozpoznáva graf $\mathcal{G}$ funkcie f. Definujme funkciu g tak, že

$$g(x_1, \ldots, x_n, y) = 0$$

práve vtedy, keď $(x_1, \ldots, x_n, y) \in \mathcal{G}$, inak je nedefinovaná. Pre vstup

$$(x_1, \ldots, x_n, y) \in \mathcal{G}$$

algoritmus $\mathcal{B}$ zastaví a povie „áno". To znamená, že existuje algoritmus, počítajúci čiastočnú funkciu g. Preto existuje aj algoritmus, počítajúci funkciu $f(x_1, \ldots, x_n) = y$, keďže

$$f(x_1, \ldots, x_n) = \mu_y(g(x_1, \ldots, x_n, y) = 0)$$

je čiastočná rekurzívna funkcia. ■

Tvrdenie H.15 *Funkcia je rekurzívna práve vtedy, keď jej graf je rekurzívny.*

Tvrdenie H.14 načrtáva ďalšiu možnosť ako definovať rozpoznateľnú (generovateľnú, rekurzívne spočítateľnú) množinu:

Dôsledok H.16 *Množina M je rozpoznateľná, ak existuje rekurzívna relácia R taká, že $x \in M$ práve vtedy, keď $\exists y\ (x, y) \in R$.*

Idea dôkazu: Z relácie R konštruujme funkciu f tak, že $f(y) = x$ pre najmenšie y také, že $(x, y) \in R$, inak (teda pre tie y, ktoré so žiadnym x netvoria dvojicu (x, y) z R) bude f nedefinovaná. Funkcia f je čiastočne rekurzívna (pretože R je rekurzívna). Podľa tvrdenia H.14 je jej graf $\mathcal{G} = \{(y, x) : f(y) = x\}$ rozpoznateľný. Množina $M = \{x : (\exists y)\ f(y) = x\}$ je tiež rozpoznateľná – podľa tvrdenia H.13 algoritmus počítajúci f generuje M.

Príklad H.17 Vzťah, vyjadrený v dôsledku H.16, ilustrujeme na alternatívnom dôkaze tvrdenia H.8: Nech $x \in Z_{\mathcal{A},n}$ znamená, že pri vstupe x algoritmus $\mathcal{A}$ produkuje výstup po n krokoch. Očividne, relácia $\{(x, \mathcal{A}, n) : x \in Z_{\mathcal{A},n}\}$ je rekurzívna. Relácia $\rho = \{(x, \mathcal{A}) : x \in Z_{\mathcal{A}}\}$ je definovateľná pomocou $x \in Z_{\mathcal{A}} \equiv \exists n(x \in Z_{\mathcal{A},n})$. Teda, ρ je rozpoznateľná (rekurzívne spočítateľná), hoci nie je rekurzívna (ZASTAVENIE nie je rozhodnuteľný problém). ■

Ukázali sme si, že popri triede problémov, ktoré môžeme riešiť pomocou nejakého algoritmu, sú problémy, ktoré nemožno algoritmicky riešiť. Prostá úvaha (funkcií z prirodzených čísel do prirodzených čísel je kontinuum[6], algoritmov iba spočítateľne veľa) dokonca ukazuje, že ich je drvivá väčšina. Okrem vypočítateľných problémov sme sa zoznámili s čiastočne vypočítateľnými problémami. Ako príklad možno uviesť problém TEORÉMY. Množina teorém (logiky prvého rádu) je rozpoznateľná (generovateľná, rekurzívne spočítateľná). O komplemente tejto množiny však nemožno povedať, že by mala takéto (celkom slušné) vlastnosti. Neskôr si ukážeme, že z algoritmického hľadiska existujú ešte ťažšie problémy.

Vráťme sa však k algoritmicky vypočítateľným (rozhodnuteľným) problémom. Uvidíme, že ani tu nie je z hľadiska praktickej vypočítateľnosti situácia bezproblémová. Keď je nejaký problém rozhodnuteľný, to ešte neznamená, že jeho riešenie možno efektívne počítať. Niektoré z tých algoritmov, ktoré rozhodujú nejaký problém, ho môžu riešiť iba vtedy, keď abstrahujeme od nárokov na (nekonečný) čas a priestor. Zvykne sa hovoriť, že riešia problém principiálne.

H.2 Zložitosť

Časová náročnosť výpočtu je jedným z kľúčových parametrov charakterizujúcich zložitosť nejakého problému a algoritmov, ktoré ho riešia. Budeme si všímať algoritmy, ktoré riešia nejaký problém a pýtať sa: Ako rýchlo ho riešia? Dá sa nájsť algoritmus, ktorý by ho riešil efektívnejšie? Možno charakterizovať celé triedy algoritmov na základe ich efektívnosti?[7]

Aby sme mohli hovoriť o časovej[8] náročnosti výpočtov, musíme sa dohodnúť o tom, ako ju merať. Táto dohoda by mala abstrahovať od náhodilostí (jednotlivých počítačov alebo modelov výpočtov). Ostaneme však na intuitívnej úrovni, tak ako doteraz.

Uspokojíme sa s dohodou, že výpočet podľa každého algoritmu je rozložený do nejakých krokov a abstrahujeme od prípadných časových rozdielov,

[6]⟨ kontinuum: dodatok E ⟩

[7]Záujemcu o dôkladnejšie informácie odkazujem na [Pap 93, GaJ 79, Gru 97].

[8]Samozrejme, náročnosť výpočtov sa dá merať aj podľa toho, aký priestor potrebujú (aj podľa ďalších mier zložitosti). Tu sa tomuto hľadisku venovať nebudeme. Čitateľ si však môže na základe analógie vytvoriť obraz, ktorý mu bude stačiť, aby pochopil niekoľko málo miest v jadre textu, vyžadujúcich nejakú predstavu o priestorových nárokoch na výpočet.

potrebných na vykonanie dvoch rôznych elementárnych krokov.[9]

Príklad H.18 Predstavme si, že hľadáme v telefónnom zozname nejakého mesta telefónne číslo nejakého človeka. Môžeme postupovať tak, že začneme prvým záznamom v telefónnom zozname, preveríme, či sme našli meno nášho človeka. Ak sme ho našli, prečítame jeho číslo a s úspechom končíme.

V opačnom prípade prejdeme na ďalší záznam a pokračujeme v tej istej činnosti. Keď prezrieme n záznamov, vykonáme n-krát viac elementárnych akcií a spotrebujeme n-krát väčší čas ako pri prezretí jedného záznamu. Robíme tak dovtedy, kým nenájdeme hľadané číslo (môže sa stať, že prejdeme celý zoznam a nenájdeme ho).

V najhoršom prípade teda musíme prezrieť všetkých n záznamov (ktoré zoznam obsahuje). Najhorším prípadom sa venujeme preto, lebo si myslíme, že je dôležité byť pesimistom – ten najhorší prípad nám pri reálnom výpočte môže veľmi nepríjemne skomplikovať život. Vo všeobecnosti, zaujíma nás, ako závisí dĺžka (čas) výpočtu na rozsiahlosti (veľkosti) vstupných dát. Usilujeme sa konštruovať funkciu, ktorá dokáže odhadnúť čas potrebný na zvládnutie uvedeného preverovania. Hľadáme teda funkciu f takú, že celkový čas, potrebný na nájdenie telefónneho čísla (v najhoršom prípade) je $f(n)$, kde n je počet záznamov v telefónnom zozname.

Ak porovnáme výpočet pre $n_1 = 100$ a $n_2 = 1000$, ukáže sa – podľa očakávania – že druhý výpočet je dlhší o nejaký násobok (určený časom potrebným na vykonanie elementárnych akcií pre jeden záznam) čísla 900 $(= n_2 - n_1)$.

Časovú náročnosť vyššie opísaného výpočtu očividne charakterizuje lineárna funkcia. S dĺžkou zoznamu lineárne rastie čas, potrebný na jeho prezretie. Grafom funkcie f by mala byť priamka, množina bodov spĺňajúca podmienku $an+b = 0$ pre nejaké konštanty a, b. Už sme sa však dohodli, že tieto konštanty závisia od rôznych náhodných okolností a budeme od nich odhliadať. O chvíľu si predstavíme značenie, ktoré túto abstrakciu podporí.

Keďže sme sa naučili robiť rozumnejšie telefónne zoznamy, vieme vyhľadať potrebné telefónne číslo oveľa rýchlejšie: siahneme niekde do stredu, ak meno hľadaného človeka je v abecede pred prvým menom na otvorenej dvojstrane, posúvame sa niekde dopredu, ak je za posledným menom na tejto dvojstrane, posúvame sa dozadu. Pri dobrej organizácii tejto činnosti časovú náročnosť výpočtu môžeme charakterizovať logaritmickou funkciou: miesto 1000 elementárnych akcií nám stačí približne 10. ■

Prvým krokom k určeniu časovej zložitosti nejakého algoritmu je teda určenie funkcie f, ktorá pre n, daný rozsah vstupu, určí hodnotu $f(n)$, t.j.

[9] Takáto dohoda by bola pripravená dôkladne vtedy, keby sme definovali, aké elementárne inštrukcie sa predpokladajú. Nebudeme však očakávať nijaké záludnosti, akou by napríklad mohla byť elementárna inštrukcia typu *nájdi rezolvenčný dôkaz formuly* ϕ. Ak čitateľ potrebuje mať istú predstavu o elementárnych inštrukciách, môže ich nájsť v dodatku F. Iný typ predstavujú elementárne inštrukcie Turingovho stroja: posun doprava, posun doľava, zápis znaku, zmena stavu, zastavenie výpočtu.

čas výpočtu pre vstup danej veľkosti. Táto funkcia odhaduje čas, potrebný na vykonanie algoritmu pre vstup danej veľkosti (v najhoršom možnom prípade).

Druhý krok spočíva v odseparovaní náhodných vplyvov zariadenia, na ktorom počítame. Budeme hovoriť iba o funkciách definovaných na prirodzených číslach. Zaujímajú nás iba neklesajúce funkcie: pre každé n predpokladáme, že $f(n+1) \geq f(n)$. Napokon, abstrahovať budeme od odlišných konštánt. Inými slovami, časovú náročnosť výpočtu budeme odhadovať iba rádovo – bude nás zaujímať, či $f(n)$ je funkcia lineárna, kvadratická, logaritmická, exponenciálna atď.

Príklad H.19 Iba pre ilustráciu kvadratický algoritmus. Naivný algoritmus, ktorý utriedi nejakú množinu (napríklad čísel, stačí však, aby to boli objekty, na ktorých je definované lineárne usporiadanie).

Algoritmus H.20
vstup: *množina M, obsahujúca n čísel $a_1, \dots, a_n$*
výstup: *M usporiadaná od najmenšieho po najväčšie číslo*

begin
for $i = 2$ **to** n **do**
  for $j = n$ **downto** i **do**
    if $a_{j-1} > a_j$ **then**
      $b := a_{j-1}; a_{j-1} := a_j; a_j := b$
end

Algoritmus vnára do seba dva cykly, ich efektom je, že sa postupne susedné čísla na i.mieste a $i+1$.mieste vymenia, ak $a_i > a_{i+1}$. Tieto dva cykly spotrebujú (približne) n^2 elementárnych akcií, potrebných na správne zaradenie všetkých n čísel. Pekne vidno, že pri tomto odhade abstrahujeme od detailov, časová náročnosť výpočtu nás zaujíma iba rádovo a určuje ju vlastne počet cyklov. V našom prípade dva cykly určujú pre vstup rozsahu n kvadratickú zložitosť n^2.

Samozrejme, aj pre tento problém existujú efektívnejšie algoritmy. ■

Zavedieme značenie, ktoré umožňuje abstrakciu od konštánt. Pre $f(n)$ budeme používať značenie $\mathcal{O}(g(n))$ v tom význame, že funkcia f nerastie rýchlejšie ako funkcia g. Presnejšie, existujú prirodzené čísla n_0 a c také, že pre všetky $n \geq n_0$ platí $f(n) \leq cg(n)$. V opačnom zmysle ako symbol $\mathcal{O}$ používame symbol Ω: $f(n) = \Omega(g(n))$, ak $g(n) = \mathcal{O}(f(n))$. Teda, f nerastie pomalšie ako g. Napokon, $f(n) = \Theta(g(n))$ práve vtedy, keď $f(n) = \mathcal{O}(n)$ a $f(n) = \Omega(g(n))$, čiže f a g sú „rádovo rovnaké“.

Príklad H.18 ukázal, že pre daný problém možno skonštruovať algoritmy, ktoré ho riešia s rôznou efektívnosťou. Prirodzená otázka je, či neexistuje nejaký algoritmus, ktorý by daný problém riešil efektívnejšie ako všetky známe algoritmy. Odpoveď na takúto otázku možno nájsť analýzou horných a dolných odhadov. Tieto odhady istým spôsobom ohraničujú možnosti algoritmického riešenia daného problému. Ak je známy algoritmus $\mathcal{A}$, ktorý rieši problém Π v čase $\mathcal{O}(f(n))$, môžeme konštatovať, že $\mathcal{O}(f(n))$ predstavuje horný odhad pre algoritmické riešenie problému Π: menej efektívny algoritmus nás už nebude zaujímať, problém Π možno riešiť prinajhoršom v čase $\mathcal{O}(f(n))$. Skonštruovaním efektívnejšieho algoritmu môžeme posunúť horný odhad smerom „nadol“.

Dôležitá otázka teraz je, ako ohraničiť zdola efektívnosť algoritmického riešenia problému Π. Teda, určiť, že žiadny algoritmus riešiaci Π nemôže počítať rýchlejšie ako je dolný odhad. Túto otázku možno zodpovedať iba dôkazom. Úlohou dôkazu je ukázať, že pre daný *problém* neexistuje algoritmus, ktorý by ho riešil rádovo rýchlejšie. Ak sa podarí dokázať, že pre daný problém neexistuje rýchlejší (efektívnejší) algoritmus, máme aj dolné ohraničenie. To znamená, ak dolný odhad je $g(n)$, tak každý algoritmus, ktorý rieši Π, počíta v čase $f(n)$, kde f nerastie pomalšie ako g, t.j. $f(n) = \Omega(g(n))$. Samozrejme, dolný odhad $g(n)$ ešte nehovorí, že existuje algoritmus, ktorý rieši Π v čase $g(n)$. Hovorí, že ho rieši *prinajlepšom* v čase $g(n)$. Po čase možno nájsť presnejší dolný odhad g', kde $g'(n) = \Omega(g(n))$.

V prípade, keď sa horný a dolný odhad zhodujú, nie je možné nájsť algoritmus, ktorý by daný problém riešil principiálne lepšie než známe najefektívnejšie algoritmy. Súčasne môžeme povedať, že poznáme výpočtovú zložitosť zodpovedajúceho problému. Daný algoritmický problém možno považovať za uzavretý. Samozrejme, vylepšovanie, ktoré nezlepší jeho efektívnosť rádovo, je možné. Hovoríme o uzavretosti iba vzhľadom na optiku $\mathcal{O}$. Oveľa zaujímavejšia situácia je, keď sa dolný a horný odhad rádovo líšia (keď sa narazí na algoritmickú priepasť). Existujú problémy, ktorých dolný odhad je lineárny a horný exponenciálny. V prípade algoritmickej priepasti sú možné dve riešenia: buď sa časom skonštruuje efektívnejší algoritmus než je ten, ktorý určuje horný odhad alebo sa dokáže presnejší výsledok o dolnom odhade. (Zopakujme: Dolný odhad hovorí, že neexistuje efektívnejší algoritmus. Nehovorí, že algoritmus s danou zložitosťou existuje, že ho možno skonštruovať – odhad môže byť nepresný.) Samozrejme, algoritmická priepasť môže ostať „bez riešenia“.

Doteraz sme sa pohybovali v oblasti zložitosti konkrétnych algoritmov – veľmi zbežne sme sa zoznámili s pojmami, ktoré sa používajú na ohodnocovanie zložitosti rôznych (konkrétnych) algoritmov.

Globálnejšiu charakterizáciu výpočtov dosiahneme, ak začneme analyzovať, aké problémy sú *zvládnuteľné* (vypočítateľné v rozumnom čase). Táto otázka je veľmi dôležitá:

Príklad H.21 Predpokladajme, že pre dané n, určujúce rozsah vstupov, výpočet programu Π trvá 2^n mikrosekúnd (zložitosť programu je $\mathcal{O}(2^n)$). Pre $n = 60$ to znamená 366 storočí. ■

Samozrejme, výpočty s takouto časovou zložitosťou nemožno považovať za realizovateľné. Kľúčovou otázkou je, kde leží hranica medzi realizovateľnými a nerealizovateľnými výpočtami. Viac ako výpočty nás však budú zaujímať triedy problémov. Stále sme na úrovni hrubých intuícií: Za zvládnuteľný problém budeme považovať taký, pre ktorý existuje algoritmus, čo sa dá počítať realizovateľným spôsobom (pre netriviálne veľké vstupy v rozumnom čase vykonateľným výpočtom). V opačnom prípade budeme hovoriť o nezvládnuteľnom probléme.

Optika je ešte hrubozrnnejšia ako v prípade $\mathcal{O}$-charakterizácií. Neprekvapuje, že vedenie takto ostrej hranice, rozdeľujúcej vypočítateľné funkcie (algoritmicky riešiteľné problémy) presne do dvoch tried, môže viesť k niektorým individuálnym zvláštnostiam. Rozhodujúce však je, že zvolená optika podáva dobrý globálny obraz. Výpočtové problémy možno rozklasifikovať do rôznych **zložitostných tried**.

Symbolom $\mathit{TIME}(f(n))$ sa označuje trieda problémov, riešiteľných algoritmom zložitosti $\mathcal{O}(f(n))$.[10] Takéto triedy nazývame zložitostnými triedami. Problémy riešiteľné v lineárnom čase tvoria triedu $\mathit{TIME}(n)$, v kvadratickom čase triedu $\mathit{TIME}(n^2)$, v polynomickom čase $P = \bigcup_k \mathit{TIME}(n^k)$.[11]

Za problémy, ktoré možno riešiť nejakým algoritmom v rozumom čase (zvládnuteľné problémy), sa v súčasnosti považujú problémy triedy P.[12]

Príklad H.22 Predstavme si algoritmus, preverujúci splniteľnosť množiny Φ formúl výrokovej logiky. (Problém splniteľnosti označujeme ako SAT.)

Algoritmus vygeneruje všetky možnosti priradenia pravdivostných hodnôt premenným. Týchto možností je 2^n, kde n je počet premenných, ktoré sa vyskytujú v danej množine formúl. Potom pre každú z týchto možností ohodnotí každú formulu z Φ podľa známych tabuliek. Čiže ide o exponenciálny algoritmus. Samotné generovanie vyžaduje exponenciálny čas. Takéto algoritmy neumožňujú vo všeobecnosti počítať v rozumnom čase. Nie je známe, či existuje efektívnejší algoritmus na riešenie SAT (s výnimkou špeciálnych prípadov, napríklad, v jadre textu sme videli polynomický algoritmus 2.47 pre hornovské formuly). ■

Prijmime teda definitívne, že algoritmy, ktorých časovú náročnosť charakterizuje exponenciálna funkcia, nie sú pre vstupy väčšieho rozsahu rozumne počítateľné.

[10] Pripomíname, že sa pohybujeme na intuitívnej úrovni. Presná definícia vyžaduje určenie typu zariadenia, na ktorom sa problém rieši algoritmom zložitosti $f(n)$.

[11] Problémy triedy P riešia algoritmy, ktorých časová zložitosť je $O(p(n))$, kde $p(n)$ je nejaká polynomická funkcia. Pretože máme nasadenú $\mathcal{O}$-optiku, abstrahujeme od komplikovanejších polynómov – napríklad n^3 nám reprezentuje všetky polynómy tvaru $an^3 + bn^2 + cn + d$.

[12] Vývoj poznania a paradigiem výpočtov však túto hranicu môže modifikovať. Ako možní kandidáti prichádzajú do úvahy charakterizácie paralelných alebo pravdepodobnostných (náhodných, randomizovaných) výpočtov.

Aby sme pokročili, využijeme ďalšiu idealizáciu, pojem **nedeterministického algoritmu**. Tak ako v prípade pojmu algoritmu ostaneme na intuitívnej úrovni.

Nedeterministický algoritmus pracuje v dvoch fázach. Najprv odhaduje, potom preveruje [GaJ 79].

Príklad H.23 Nech je daná množina výrokovologických formúl Φ, v ktorých sa vyskytujú výrokové premenné $p_1, \ldots, p_n$. Ilustrujme predstavu o nedeterministickom algoritme, riešiacom problém splniteľnosti (SAT). Vo fáze odhadu sa nejakým zázračným spôsobom vygeneruje ν, n-tica pravdivostných hodnôt (i-ty prvok tejto n-tice je hodnotou premennej p_i). Vo fáze preverenia sa zisťuje, či ν splňuje Φ.

Ak je Φ splniteľná, nejaký výber bude úspešný. ■

Na začiatku tohto dodatku sme problém spojili s nejakou výrokovou formou ϕ a otázkou *je ϕ pravdivé?*. Každá inštancia ϕ vedie k inštancii tohto problému. Niektoré z inštancií sú pozitívnymi inštanciami – sú to tie inštancie problému, na ktoré je správna odpoveď *áno*. Predpokladajme teda rozhodovací problém Π a nejakú jeho inštanciu I. Pri riešení Π sa využíva trieda štruktúr S (v príklade problému SAT by touto triedou bola množina, ktorá pre každé n obsahuje množinu všetkých možných n-tíc pravdivostných hodnôt). **Nedeterministický algoritmus** pre dané I:

1. Vo fáze odhadu vyberie z S jedinú štruktúru s.

2. Vo fáze preverovania zistí, či S dovoľuje na I odpovedať *áno*, či *nie*.

Hovoríme, že nedeterministický algoritmus **rieši** rozhodovací problém Π vtedy, keď pre každú jeho inštanciu I platí:

1. Ak I je pozitívna inštancia, potom *existuje* štruktúra $s \in S$ taká, že keď ju vyberieme vo fáze odhadu, vo fáze preverenia dostaneme odpoveď *áno*.

2. V opačnom prípade *neexistuje* štruktúra $s \in S$, ktorá by viedla cez fázy odhadu a preverenia k odpovedi *áno*.

Ide tu o veľmi liberálny pohľad na riešenie problémov (zodpovedá predstave, že nejak sa správne riešenie dá nájsť). Dobre ho napokon ilustruje príklad H.23.

Budeme hovoriť, že nedeterministický algoritmus $\mathcal{N}$, ktorý rieši problém Π je **polynomický**, ak existuje polynóm p taký, že pre každú jeho pozitívnu inštanciu I existuje odhad $s \in S$, ktorý vo fáze preverenia dá odpoveď *áno* v čase $p(n)$, kde n určuje rozsah vstupu inštancie I. Na označenie algoritmov s práve charakterizovanou vlastnosťou budeme používať termín **nedeterministický polynomický** algoritmus. Celú triedu problémov, riešiteľných pomocou nedeterministických polynomických algoritmov budeme označovať pomocou NP.

Množina problémov rozhodnuteľných nedeterministickým algoritmom pre vstup rozsahu n v čase $f(n)$ sa označuje ako $NTIME(f(n))$. Tie, čo sú rozhodnuteľné nedeterministickým algoritmom v čase kvadratickom, tvoria triedu $NTIME(n^2)$, v čase polynomickom $NP = \bigcup_k NTIME(n^k)$.

Každý problém z triedy P je očividne aj v triede NP. Otázka je, či existujú problémy z NP, ktoré nie sú z P, teda nie sú problémami vypočítateľnými deterministicky v polynomickom čase. Videli sme napríklad exponenciálny deterministický algoritmus a polynomický nedeterministický algorimus, ktorý rieši SAT (príklady H.22 a H.23). Existuje lepší ako exponenciálny deterministický algoritmus na riešenie SAT? Je to vážna otázka z teoretického aj praktického hľadiska. Z hľadiska praktického ide o hranice zvládnuteľného počítania. Z hľadiska teoretického ide o vhľad do toho, čo je riešiteľné, poznateľné pri našich obmedzených zdrojoch.

Žiaľ, hranica medzi zvládnuteľnými a nezvládnuteľnými problémami nie je s definitívnou platnosťou určená. Nie je známe, či platí $P = NP$ alebo $P \neq NP$. Predpokladá sa však, že $P \neq NP$ a že hranica leží práve medzi P a NP.[13]

Zvláštnu pozornosť si zasluhujú najťažšie problémy z triedy NP, zvané – ako inak – NP-úplnými. Trieda ***NP*-úplných** problémov sa identifikuje pomocou pojmu polynomickej redukovateľnosti. Hovoríme, že problém Π je polynomicky redukovateľný (budeme používať aj značenie p-redukovateľný)[14] na problém Π', ak existuje funkcia p, vypočítateľná v polynomickom čase, ktorá každému vstupu x problému Π priraďuje vstup $p(x)$ problému Π', pričom riešenie problému Π' pre vstup $p(x)$ je aj riešením problému Π pre vstup x.

Problém Π je NP-úplný, ak patrí do triedy NP a každý iný problém z triedy NP možno polynomicky redukovať na Π. Očividne, polynomická redukcia je tranzitívna. Teda, všetky NP-úplné problémy sú vzájomne redukovateľné.

Trieda NP-úplných problémov má túto dôležitú vlastnosť: Ak sa nájde aspoň jeden problém z tejto triedy, ktorý je aj z triedy P (je vypočítateľný v polynomickom čase deterministicky), potom všetky NP-úplné problémy sú z triedy P. V opačnom prípade žiaden z nich nie je (deterministicky) vypočítateľný v polynomickom čase. Vyjadrujú to nasledujúce tvrdenia.

Tvrdenie H.24 *Ak problém Π možno polynomicky redukovať na problém Π' a Π' patrí do triedy P, potom aj Π patrí do triedy P.*

---

[13] Ďalej to budeme predpokladať aj my. Mnohé tvrdenia, ktoré budeme uvádzať, platia za predpokladu, že $P \neq NP$.

[14] Kontrast k pojmu a-redukovateľnosti ukazuje, že ide o podstatne prísnejší pojem: pri a-redukcii stačí vypočítateľná transformácia, pri p-redukovateľnosti musí ísť o zvládnuteľne vypočítateľnú transformáciu.

Ak problém Π *možno polynomicky redukovať na problém* Π' *a* Π *nepatrí do triedy* P*, potom ani* Π' *nepatrí do triedy* P.

Samozrejme, ťažko preverovať, či každý problém z triedy *NP* možno redukovať na Π. Nasledujúca lema umožňuje na základe jedného známeho *NP*-úplného problému konštruovať množinu *NP*-úplných problémov.

Lema H.25 *Ak* Π *a* Π' *patria do triedy NP,* Π *je polynomicky redukovateľný na* Π' *a* Π *je NP-úplný, potom aj* Π' *je NP-úplný.*

SAT bol historicky prvý problém, ktorého *NP*-úplnosť bola dokázaná.

Tvrdenie H.26 *Problém splniteľnosti formúl výrokovej logiky (SAT) je NP-úplný.*

Každý *NP*-úplný problém možno redukovať napríklad na problém SAT. Dôkaz *NP*-úplnosti nejakého problému Π spočíva (pozri [GaJ 79]) v tom, že:

1. ukážeme, že $\Pi \in NP$,
2. nejaký *NP*-úplný problém Π' redukujeme na Π (ukážeme tým, že Π nie je ľahší ako Π', z riešenia Π dostaneme riešenie Π').

Zaujímavý pohľad na komplikovanú štruktúru problémov z triedy *NP* dáva:

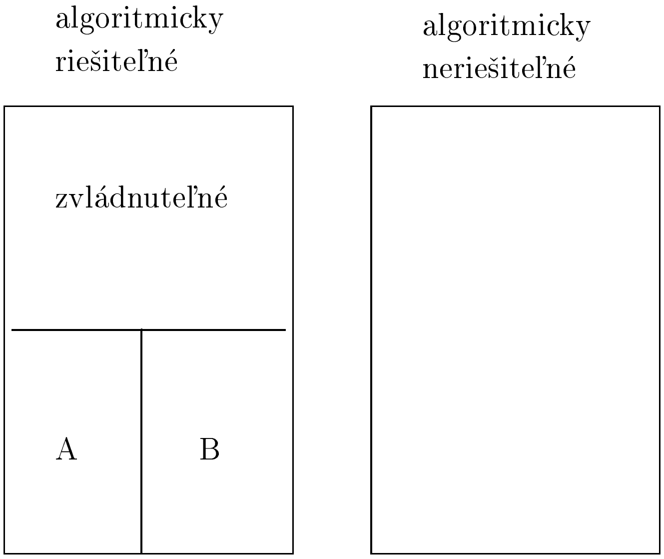
Tvrdenie H.27 ([Pap 93], Theorem 14.1) *Ak* $P \neq NP$*, potom existuje problém z triedy NP, ktorý nepatrí do* P *a ani nie je NP-úplný.*

H.3 Ťažké problémy

Videli sme, že problémy, ktoré sú algoritmicky riešiteľné, nemusia byť reálne riešiteľné výpočtami v rozumnom čase. Algoritmická riešiteľnosť (rozhodnuteľnosť) je idealizácia, ktorá hovorí o principiálnej vypočítateľnosti a abstrahuje od nárokov na čas a priestor, potrebných pre výpočet.

Videli sme, že niektoré problémy z principiálne vypočítateľných možno slušne a úspešne riešiť algoritmicky. Nazývame ich zvládnuteľnými problémami a stotožňujeme ich s triedou problémov P.

Popri tom existujú problémy principiálne vypočítateľné, ale nezvládnuteľné. Nie sú predpoklady, že pre vstupy väčšieho rozsahu ich možno v rozumnom čase algoritmicky riešiť. Triedou problémov základného významu s touto vlastnosťou sú *NP*-úplné problémy a charakterizujeme ich ako najťažšie problémy riešiteľné pomocou polynomických nedeterministických algoritmov. Teraz sa zoznámime s problémami, ktoré sú ešte ťažšie ako *NP*-úplné. Stále však ide o problémy, ktoré sú principiálne riešiteľné s využitím nejakého algoritmu („principiálne“ znamená, že sa abstrahuje od nárokov na čas aj priestor,



Obrázok H.1: Principiálna a praktická nevypočítateľnosť.

potrebných pre úspešný výpočet). Problémy ťažšie ako *NP*-úplné vytvárajú hierarchiu, s ktorou sa tu oboznámime. Nazývame ju polynomickou hierarchiou.

Popri problémoch, ktoré možno algoritmicky riešiť, sme sa zoznámili s takými, pre ktoré je to vylúčené. Teda, nemožno ich riešiť ani „principiálne“ s využitím nejakého algoritmu, využívajúceho neobmedzený čas a priestor. Pomerne veľa miesta sme venovali „najľahším“ z týchto problémov – polorozhodnuteľným problémom. V tejto časti ukážeme aj to, že existujú ešte ťažšie problémy. Budeme hovoriť o stupňoch nerozhodnuteľnosti a zavedieme celú hierarchiu problémov ťažších ako sú polorozhodnuteľné. Túto hierarchiu nazývame aritmetickou.

Rôzne druhy problémov z hľadiska principiálnej a reálnej nevypočítateľnosti názorne ilustruje obrázok H.1. Obdĺžnik vľavo predstavuje algoritmicky riešiteľné problémy. Z nich časť tvoria zvládnuteľné problémy, časť nezvládnuteľné problémy. Nezvládnuteľné sú na obrázku rozdelené opäť na dve časti, A a B. A predstavuje *NP*-úplné problémy a B problémy ťažšie ako sú *NP*-úplné. Do štruktúry problémov z B vhliadneme pomocou polynomickej hierarchie, ktorej sa o chvíľu budeme venovať. Obdĺžnik vpravo predstavuje algoritmicky neriešiteľné problémy. Aritmetická a analytická hierarchia poskytujú istý vhľad do ich štruktúry (prvej z nich sa onedlho budeme venovať).

Výpočty, problémy, aplikácie Skôr než prejdeme k spomínaným druhom „ešte nerozhodnuteľnejších" a „ešte nezvládnuteľnejších" problémov (a k zodpovedajúcim hierarchiám), stručne o dôvodoch, prečo sa nimi budeme zaoberať. Najstručnejšie – týmto dôvodom je, že mnohé problémy umelej inteligencie patria medzi veľmi ťažké výpočtové problémy. Trochu podrobnejšie:

Pri polorozhodnuteľných a NP-úplných problémoch sme narazili na výraznú asymetriu. Všimnime si najprv rozpoznateľné množiny, ktoré nie sú rekurzívne. (Tieto množiny môžeme považovať za množiny riešení polorozhodnuteľných problémov.) Algoritmus, ktorý akceptuje rozpoznateľnú množinu M, odpovedá „áno" pre každé $x \in M$, v opačnom prípade nezastaví výpočet. Použijeme metaforu certifikátu. Každý prvok rozpoznateľnej množiny predstavuje certifikát pre jednu inštanciu polorozhodnuteľného problému.

Príklad H.28 Ak zoberieme nejaký objekt x (napríklad formulu predikátového počtu) a spýtame sa, či tento objekt patrí do množiny M (či je teorémou predikátového počtu) a máme k dispozícii y, konečnú množinu formúl, ktorá je skutočne dôkazom x, toto y slúži ako certifikát pre rozhodnutie o danej inštancii problému (teda potvrdzuje, že x je teorémou).

Vieme však, že nemáme podobný certifikát pre komplementárny problém (či x nie je teorémou). Ak by sme mali aj takýto certifikát, TEORÉMY by boli rozhodnuteľným problémom (podľa tvrdenia H.10).

Je nepreberné množstvo potenciálnych dôkazov nejakej teorémy. Na potvrdenie toho, že nejaká formula je dokázateľná, stačí nájsť a preveriť jeden dôkaz.

Na vyvrátenie predpokladu, že teoréma je dokázateľná (na tvrdenie, že nie je dokázateľná), však musíme mať istotu, že *žiadna* postupnosť formúl nie je jej dôkazom.

Teda, s polorozhodnuteľnými problémami je vnútorne spätá výrazná asymetria: máme iba jednostranné certifikáty, potvrdzujúce ich riešenie (alebo riešenie ich komplementu, nikdy však nie oba certifikáty). ■

Spravidla je ťažké nájsť certifikát. Keď ho však už máme, ľahko danú inštanciu problému vyriešime. Každý certifikát je totiž konečný (napríklad – pre problém TEORÉMY je ním dôkaz, pre problém ZASTAVENIE kroky algoritmu $\mathcal{A}$ od štartu pri vstupe x až po zastavenie).

Predpokladajme, že je definovaná množina U (univerzum, základný priestor, v ktorom sa pohybujeme).[15] Pre rekurzívne spočítateľné množiny budeme používať značenie RE (s významom $M \in RE$ práve vtedy, keď $M \subseteq U$ a M je rekurzívne spočítateľná). Symbolom $coRE$ označíme množinu $\{\overline{M} : \overline{M} = U \setminus M \wedge M \in RE\}$. Teda, do $coRE$ patria komplementy množín z RE. (A symetricky, $M \in coRE$, ak jej komplement je z RE.) Ak si triedu rekurzívnych množín označíme ako R, potom platí:

[15] Vieme, že pre potreby našej analýzy môžeme stotožniť U napríklad s množinou Nat všetkých prirodzených čísel.

Tvrdenie H.29 *$M \in R$ práve vtedy, keď $M \in RE$ a $\overline{M} \in coRE$.*

Ide iba o inú formuláciu tvrdenia H.10 a). Teda, symetria pre problém a jeho komplement je charakteristická iba pre problémy triedy R. V triede RE nutne narážame na asymetriu možností ako preveriť/vyvrátiť návrh (kandidáta) na riešenie problému.

Prejdime teraz k ďalšej asymetrii. Nedeterministický algoritmus nejako vyberie riešenie problému (ak toto riešenie existuje). Oveľa ťažšiu situáciu máme, keď riešenie neexistuje. Opäť si pomôžeme metaforou certifikátu.

Príklad H.30 Konkrétna výrokovologická interpretácia je certifikátom nejakej inštancie problému SAT (spĺňania), ak s jej využitím preveríme, že daná množina formúl je splnená. Asymetricky, táto interpretácia nemôže slúžiť ako certifikát pre komplementárny problém, pre problém nesplniteľnosti (tam totiž nesmie danú množinu formúl spĺňať žiadna z prípustných interpretácií). ∎

Nedeterministický algoritmus „vyrieši" (odpovie „áno" na) rozhodovací problém vtedy, keď *nejaký* z jeho výberov je úspešný. Odpovie „nie" iba vtedy, keď *žiadny* z výberov nie je úspešný. Ostrá asymetria. K pojmu certifikátu sa ešte vrátime.

Zaujímavý postreh pochádza od [Pap 93]: pojmy akceptovania, polorozhodnuteľnosti, nedeterminizmu nie sú pravé algoritmické pojmy, sú to skôr užitočné nástroje na klasifikovanie problémov (viac hovoria o problémoch ako výpočtoch). Sú to pojmy, ktorými charakterizujeme výpočtovo ťažké problémy (resp. aplikácie výpočtov, aplikácie algoritmov). Idea je asi takáto – výpočet v pravom zmysle slova (zdravý výpočet) je taký výpočet, ktorý dá nejaký výsledok (zastaví) v konečnom čase. Ak je nejaký problém výpočtovo veľmi ťažký a chceme ho nejako (pseudo)výpočtovo charakterizovať, zavádzame istú fikciu (pseudo)výpočtu (idealizovaný výpočet), ktorú možno charakterizovať pomocou pojmov ako akceptovanie, polorozhodnuteľnosť, nedeterminizmus.

Problémy, ktoré kladie umelá inteligencia, patria medzi takéto ťažké problémy. Spravidla ide o výber nejakého riešenia spomedzi exponenciálneho množstva (vzhľadom na rozsah vstupov) možných riešení.

Ťažšie ako polorozhodnuteľnosť Prechod od rozhodovania k akceptovaniu je prechodom k problémom, ktoré sú z algoritmického hľadiska ťažšie. Máme teda prvý príklad „výpočtov", ťažších ako sú „rekurzívne výpočty". Môžeme sa celkom prirodzene pýtať, či neexistujú ďalšie druhy „výpočtov", ktorých výpočtová náročnosť postupne rastie (viac a viac). Odpoveď je taká, že celkom prirodzene môžeme hovoriť o stupňoch nerozhodnuteľnosti.

Základným nástrojom, ktorý nám takúto hierarchiu umožní skonštruovať, je pojem **orákula**. Nech h je nejaká funkcia. Orákulum pre h si môžeme predstaviť ako nejakého špecializovaného poradcu, ktorý na otázku o hodnote funkcie h pre argument k odpovedá hodnotou $h(k)$.

Teraz obohatíme pojem algoritmu o volanie nejakého orákula. Budeme hovoriť, že funkcia f je **rekurzívna vzhľadom na orákulum** O, ak existuje algoritmus, ktorý počíta f a na výpočet niektorých hodnôt využíva orákulum O.

Príklad H.31 Videli sme už, že nerozhodnuteľnosť problému TEORÉMY možno dokázať redukciou na problém ZASTAVENIE. Môžeme si to preložiť do terminológie orákul. Keby sme mali orákulum O, čo rozhoduje problém TEORÉMY, problém ZASTAVENIE by bol rozhodnuteľný vzhľadom na orákulum O. Vieme už totiž, že v prvorádovom jazyku sa dá efektívne skonštruovať konečná množina viet Φ a veta ϕ také, že $\Phi \models \phi$ práve vtedy, keď daný algoritmus $\mathcal{A}$ zastaví pre daný vstup x. Ak by orákulum O rozhodovalo vzťah $\models$ (t.j. aj problém TEORÉMY), potom by musel byť rozhodnuteľný aj problém ZASTAVENIE. Problém zastavenie je teda rozhodnuteľný vzhľadom na orákulum O. ■

Vidíme, že konštrukt orákula je veľmi silný a veľmi idealizovaný.

Ak je orákulum O rekurzívna funkcia, potom funkcia rekurzívna vzhľadom na orákulum O je rekurzívna funkcia. Ak je f rekurzívna vzhľadom na orákulum g a g je funkcia rekurzívna vzhľadom na orákulum h, potom f je rekurzívna vzhľadom na orákulum h. Pojem relatívnej rekurzívnosti (rekurzívnosti vzhľadom na orákulum) teda generuje čiastočné usporiadanie na funkciách.

„Najľahšie“ vypočítateľné funkcie sú rekurzívne. Máme totiž algoritmy, ktoré ich počítajú. Ak je funkcia f rekurzívna vzhľadom na g, potom výpočet g nie je ľahší ako výpočet f. Ak g nie je ľahšie vypočítateľné ako f, potom výpočet f vzhľadom na g je tak ťažký ako výpočet g.

Príklad H.32 Uvedieme si príklad problému ťažšieho ako sú polorozhodnuteľné problémy. Ťažšie problémy ako polorozhodnuteľné nemajú ani jednostranný certifikát. Príkladom je: zastaví algoritmus $\mathcal{A}$ pre všetky vstupy? ■

Teraz zavedieme **aritmetickú hierarchiu**. Budeme hovoriť, že rekurzívne množiny sú množiny typu Σ_0 a Π_0. Ďalej definujeme:

- Množina M je typu Σ_{n+1}, ak $x \in M$ práve vtedy, keď $\exists y((x,y) \in R)$, kde R je množina typu Π_n.[16]
- Množina M je typu Π_{n+1}, ak $x \in M$ práve vtedy, keď $\forall y((x,y) \in R)$, kde R je množina typu Σ_n.[17]

[16] Množiny typu Σ_n sú zovšeobecnením rozpoznateľných (generovateľných, rekurzívne spočítateľných) množín – rozpoznateľná množina je množina typu Σ_1.

[17] Množiny typu Π_n ešte viac zovšeobecňujú predstavu o funkciách, počítateľných ťažšie

Samozrejme, už vieme, ako definovať najťažšie problémy v každej zo spomínaných tried:

Množina M je **úplná** množina typu Σ_n (resp. Π_n), ak je množina typu Σ_n (Π_n) a každú množinu typu Σ_n (Π_n) možno a-redukovať na M.

Doteraz sme sa pohybovali od rozhodnuteľných problémov k ťažším a ťažším problémom (alebo množinám, ktoré reprezentujú ich riešenia). Tento smer pozornosti vyústil v hierarchiu množín, uvedenú pred chvíľou (nazýva sa **aritmetickou hierarchiou**).[18] Už problémy (množiny) typu Σ_1 sú z hľadiska praktickej vypočítateľnosti veľmi nepríjemné. Analyzovali sme idealizácie, formalizácie predstáv o veľmi ťažkých (veľmi nerozhodnuteľných) problémoch. Vráťme sa však k problémom, ktoré principiálne možno riešiť výpočtami.

Ťažšie ako *NP*-úplnosť Najprv zadefinujeme pojem *NP*-ťažkého problému, potom prejdeme k výkladu, ktorý nás dovedie k polynomickej hierarchii.

Ak máme nejaký problém, ktorý nie je v *NP*, ale *nejako* ho možno redukovať na *NP*-úplný problém, podľa všetkého ide o ťažší problém ako *NP*-úplný. Budeme hovoriť o *NP*-ťažkých problémoch.

Hovoríme, že problém Π redukujeme na problém Π' pomocou polynomickej Turingovskej redukcie (T-redukujeme), ak existuje algoritmus $\mathcal{A}$, ktorý rieši problém Π tak, že volá nejaké orákulum O, riešiace problém Π'. Pritom platí: ak by O riešilo Π' v polynomickom čase, potom by $\mathcal{A}$ riešil Π v polynomickom čase.

Problém Π' je *NP*-ťažký, ak existuje *NP*-úplný problém Π, ktorý je T-redukovateľný na Π'.

Je známe, že existujú problémy, principiálne riešiteľné algoritmicky, ale ťažšie ako *NP*-úplné problémy. Ide o problémy, o ktorých je dokázané, že neexistuje algoritmus lepší ako exponenciálny, ktorý by ich riešil. Príkladmi takýchto problémov sú zovšeobecnený šach, zovšeobecnená dáma (na šachovniciach rozmerov $n \times n$), alebo splniteľnosť formúl dynamickej logiky[19] a ďalšie, pozri napríklad [Har 87].

Naším najbližším cieľom je dať metafore certifikátu presné vyjadrenie. Najprv dva dôležité pojmy – polynomická rozhodnuteľnosť a polynomická vyváženosť.

Relácia $R \subseteq U \times U$ je **polynomicky rozhodnuteľná** práve vtedy, keď existuje algoritmus, ktorý v polynomickom čase rozhoduje problém, či $(x, y) \in R$.

ako funkcie čiastočne rekurzívne: v tomto prípade „náročnosť výpočtu" rastie od existenčného kvantifikátora k všeobecnému (k danému x nestačí nájsť nejaké y také, že $(x, y) \in R$, treba nájsť všetky y s touto vlastnosťou).

[18] O aritmetickej hierarcii a o analytickej hierarchii, ktorej sa tu nevenujeme, pozri podrobnejšie v [Rog 67].

[19] Dynamická výroková logika popri výrokových premenných narába ešte so symbolmi reprezentujúcimi akcie (programy). Intuitívne, vyjadruje závislosť stavov na akciách. Ak obvyklá výrokovologická formula ϕ opisuje nejaký stav a aplikuje sa na ňu výraz reprezentujúci nejakú akciu (program) α, formula $\alpha\phi$ znamená, že po akcii α bude platiť ϕ.

Nech existuje rekurzívna injekcia $f : Nat \longrightarrow U$ taká, že $U = \{x : (\exists n \in Nat)\ f(n) = x\}$. Budeme hovoriť, že R je polynomicky vyvážená práve vtedy, keď pre každú dvojicu $(x, y) \in R$ takú, že $f(n) = x$ a $f(m) = y$ platí $m < n^k$ pre nejaké $k \geq 1$. Intuícia za vyváženosťou: ide o kompaktnú, hutnú reprezentáciu.

Tvrdenie H.33 ([Pap 93]) *Nech $M \subseteq U$ je množina riešení problému Π. Potom $\Pi \in NP$ práve vtedy, keď existuje polynomicky rozhodnuteľná a polynomicky vyvážená relácia R taká, že $M = \{x : \exists y\ (x, y) \in R\}$.*

Môžeme prejsť k objasneniu predstavy certifikátu: pre každú pozitívnu inštanciu NP-úplného rozhodovacieho problému Π (teda pre každé x na vstupe algoritmu riešiaceho problém Π, pre ktoré je odpoveď „áno") existuje y také, že $(x, y) \in R$. Keďže R je polynomicky rozhodnuteľná a (v istom zmysle slova) kompaktná, hutne reprezentovateľná (asi toľko cítime za polynomickou vyváženosťou), môžeme povedať, že y je certifikát, umožňujúci efektívne potvrdiť pozitívnu inštanciu daného problému.

Pochopiteľne, aj pre NP-úplné problémy máme iba jednostranné certifikáty. Rovnako pochopiteľné je, že nie sme nutne schopní nájsť (vypočítať) takýto certifikát v rozumnom (polynomickom) čase. Keď však už certifikát máme, inštanciu problému môžeme pohodlne vyriešiť.

Príklad H.34 Keď máme interpretáciu $\mathcal{I}$, ktorá spĺňa množinu výrokovologických formúl Φ, ľahko sa presvedčíme, že Φ je splniteľná. V takom prípade je $\mathcal{I}$ certifikátom splniteľnosti Φ.

Relácia $R = \{(\Phi, \mathcal{I}) : \Phi$ je množina výrokovologických formúl, $\mathcal{I}$ je výrokovologická interpretácia a $\mathcal{I}$ spĺňa $\Phi\}$ je rozhodnuteľná v polynomickom čase. Polynomickú vyváženosť možno zabezpečiť tak, že interpretácie reprezentujeme ako množiny literálov a množiny formúl vhodne kódujeme prirodzenými číslami. ■

Podobne ako sme definovali triedu *coRE*, budeme definovať aj triedu *coNP*. Vo všeobecnosti, ak máme zložitostnú triedu $\mathcal{C}$, potom triedu problémov $co\mathcal{C}$ môžeme vymedziť takto: nech $\overline{M}$ je množina všetkých riešení nejakého problému Π, ktorý patrí do triedy $\mathcal{C}$, potom komplementárny problém, ktorého množinou riešení je M, patrí do $co\mathcal{C}$.

Problém patrí do triedy *coNP*, ak problém k nemu komplementárny patrí do triedy *NP*. Zrejme, aj pre problémy z *coNP* existujú (polynomické) certifikáty – tieto certifikáty však slúžia ako falzifikácie inštancií daného problému: umožňujú v polynomickom čase rozhodnúť, že nejaké x je riešením komplementárneho problému, teda odpoveď pre túto inštanciu je „nie".

Príklad H.35 Výrokovologická formula je platná, ak je splnená každou interpretáciou. Problém PLATNOSŤ spočíva v rozhodovaní, či formula ϕ na vstupe je platná. Venujme sa teraz komplementárnemu problému, NE-PLATNOSTI.

Keďže ϕ je neplatná práve vtedy, keď $\neg\phi$ je splniteľná a splniteľnosť (SAT) je NP-úplný problém, aj NE-PLATNOSŤ je NP-úplný problém, PLATNOSŤ je teda $coNP$-úplný problém. ■

Opäť budeme používať pojem (metaforu, intuíciu) orákula. Algoritmus $\mathcal{A}$, ktorý využíva orákulum O, môže počas svojej činnosti klásť otázky orákulu O a to mu na ne odpovedá. Toto orákulum si v prvom kroku predstavme ako množinu odpovedí na prípustné otázky algoritmu $\mathcal{A}$. V druhom kroku už nie je ťažké túto množinu chápať ako množinu riešení nejakého (pod)problému.

Zavedieme nové zložitostné triedy: ak $\mathcal{C}$ je zložitostná trieda a O je orákulum, potom $\mathcal{C}^O$ je trieda, do ktorej patria problémy, rozhodnuteľné algoritmami, ktorých výpočtová zložitosť patrí do triedy $\mathcal{C}$, pričom tieto algoritmy môžu klásť otázky orákulu O.

Keďže orákulum chápeme ako množinu riešení nejakého (pod)problému, stačí nám uvažovať o O ako o reprezentantovi nejakej zložitostnej triedy (tej triedy, do ktorej daný podproblém patrí). Preto napríklad NP^{NP} bude zložitostná trieda, do ktorej patria problémy, riešiteľné nejakým nedeterministickým algorimom, ktorý kladie nejaké otázky orákulu a to orákulum odpovedá s využitím nedeterministického algoritmu, teda aj samotné orákulum patrí do triedy NP.

Považujme preto $\mathcal{C}$ a O za zložitostné triedy. $\mathcal{C}^O$ je trieda problémov, ktoré sú riešiteľné algoritmom z triedy $\mathcal{C}$, pričom tento algoritmus volá orákulum zložitostnej triedy O.

Sme pripravení definovať polynomickú hierachiu:

- $\Sigma_0^P = \Pi_0^P = \Delta_0^P = P$,
- $\Delta_{k+1}^P = P^{\Sigma_k^P}$,
- $\Sigma_{k+1}^P = NP^{\Sigma_k^P}$,
- $\Pi_{k+1}^P = coNP^{\Sigma_k^P}$.

Kumulatívna polynomická hierarchia sa definuje ako $PH = \bigcup_{i \geq 0} \Sigma_i^P$.

Príklady.

- $\Delta_1^P = P^P = P$ (Polynomický algoritmus, ktorý volá polynomický algoritmus, je polynomický.)
- $\Sigma_1^P = NP^P = NP$ (Polynomický nedeterministický algoritmus, ktorý volá polynomický algoritmus, je polynomický nedeterministický algoritmus.)
- $\Pi_1^P = coNP^P = coNP$
- $\Delta_2^P = P^{NP}$

- $\Sigma_2^P = NP^{\Sigma_1^P} = NP^{NP}$ (NP-algoritmus volá NP-orákulum.)
- $\Pi_2^P = coNP^{NP}$

Samozrejme, pre každú z tried týchto hierarchií možno obvyklým spôsobom definovať úplnosť, napríklad Σ_2^P-úplnosť alebo Π_2^P-úplnosť.

Zovšeobecnenie idey certifikátov (pre problém triedy Σ_i^P) prináša nasledujúce tvrdenie, zovšeobecnenie tvrdenia H.33.

Tvrdenie H.36 *Nech Φ je problém. $\Phi \in \Sigma_i^P$ práve vtedy, keď existuje polynomicky vyvážená relácia R taká, že problém rozhodovania členstva v R patrí do triedy Π_{i-1}^P a množina riešení problému Φ je $M = \{x : \exists y\ (x, y) \in R\}$.*

Príklad H.37 Ak $\Phi \in \Sigma_2^P = NP^{NP}$, existuje $coNP$-certifikát pre pozitívne inštancie Φ. ■

Prístupy k nezvládnuteľným problémom Keď je nejaký problém NP-úplný alebo ťažší, nie je možné jeho korektné a úplné algoritmické riešenie v rozumnom čase. Je preto produktívne hľadať spôsoby, ako ho riešiť menej uspokojujúcim spôsobom, ale v rozumnom čase.

Možno riešiť jeho špeciálne prípady, možno hľadať návody (heuristiky), ktoré nezaručujú, že problém vyriešime, ale ich úspešnosť je slušná. Môžeme sa usilovať o rozumné aproximácie riešenia (aproximačné algoritmy) alebo o riešenie s istou pravdepodobnosťou (pravdepodobnostné algoritmy), pozri napríklad [Pap 93, GaJ 79] alebo populárny výklad v [Har 87].

Základnú informáciu o flexibilných algoritmoch možno nájsť v časti 8.4.

Zaujímavý prístup k analýze výpočtovej zložitosti predstavuje fixovanie niektorých parametrov. Pri aplikáciách často niektorý z parametrov možno zafixovať. Potom možno hovoriť o tom, že problém so zafixovaným parametrom je zvládnuteľný (fp-zvládnuteľný). Teóriu parametrizovanej zložitosti vyvinuli [DoF 99]. Na problémy umelej inteligencie ju aplikovali [GSS 99].

Dodatok I

Autoepistemická logika

Na úvod si ilustrujme základnú myšlienku, ktorá viedla ku konštrukcii autoepistemických teórií. Túto myšlienku vlastne už poznáme, zoznámili sme sa s ňou v jednom z príkladov defaultových pravidiel. Pripomenieme si ho:

Príklad I.1 (Dôsledky neoboznámenosti) Ak neviem o tom, že by som mal sestru, mám vážny dôvod veriť tomu, že sestru nemám. (V doménach, ktoré dobre poznáme, je niekedy dôvodom pre prijatie nejakého záveru neinformovanosť o tom, že by platil jeho opak.)

$$\frac{: \neg m\acute{a}m\_sestru}{\neg m\acute{a}m\_sestru}$$

■

Teda, jedným zo spôsobov uvažovania podľa „zdravého rozumu" sú „inventúry" vlastných znalostí: Ak neviem o niečom z oblasti, ktorú dobre poznám, potom „toto niečo" podľa všetkého nie je pravdivé. Nemonotónnosť je zrejmá: ak dnes niečo neviem a z tohto predpokladu odvodzujem nejaké dôsledky, potom tieto dôsledky môžem v budúcnosti odmietnuť.

Moore [Moo 85] túto myšlienku sformalizoval („vyčistil" predchádzajúce pokusy McDermotta a Doyla [McD 80], [McD 82]). Moore sa rozhodol systematicky formalizovať introspekciu. Prejdime k jeho formalizácii:

Budeme používať výrokovologický jazyk, obohatený o nový operátor K. Intuícia, spätá s $K\phi$ je očakávaná: viem, že ϕ (prípadne – je známe, že ϕ).[1]

Pojem formuly možno zaviesť obvyklým spôsobom. Formuly výrokovej logiky nazveme objektívnymi formulami. Autoepistemická formula sa definuje rekurzívne:

[1] Presnejšie čítanie by bolo „myslím si, že viem ... " prípadne „verím (som presvedčený), že ... ". Teda, môžem „vedieť" niečo aj vtedy, keď to nie je pravda. V takomto zmysle slova tento operátor formalizoval Moore a v tom zmysle sa používa aj v každodennom uvažovaní. Aj o našich omyloch si myslíme, že sú poznatkami.

- každá objektívna formula je autoepistemická formula,
- ak ϕ je autoepistemická formula, potom $K\phi$ je autoepistemická formula,
- ak ϕ a ψ sú autoepistemické formuly, potom $\phi \wedge \psi$ a $\neg\phi$ sú autoepistemické formuly.

Formuly tvaru $K\phi$ a $\neg K\phi$ nazývame aj subjektívnymi formulami (autoepistemickými literálmi).

Autoepistemická teória je množina autoepistemických formúl.

Pokračujeme sémantikou. V adekvátnej sémantike pravdivostná hodnota formuly $K\phi$ nemôže závisieť na pravdivostnej hodnote formuly ϕ. To, že niečo je pravdivé (alebo nepravdivé), ešte neznamená, že je pravdivé (nepravdivé), že o tom viem.

Preto je rozumné definovať pravdivostnú hodnotu subjektívnych formúl vzhľadom na nejakú množinu „presvedčení“. Každú množinu autoepistemických formúl budeme nazývať *množinou presvedčení* (belief set). Autoepistemickú interpretáciu zadefinujeme tak, že subjektívne formuly budeme interpretovať pomocou množiny presvedčení a objektívne pomocou klasického, obvyklého výrokovologického ohodnotenia.

Definícia I.2 *Autoepistemická interpretácia* je dvojica (m, S), kde m je 2-interpretácia (konzistentná množina výrokovologických literálov)[2] a S je množina presvedčení.

Pre danú autoepistemickú interpretáciu $\mathcal{I} = (m, S)$ definujeme $val_{\mathcal{I}}$ ako zobrazenie z množiny autoepistemických formúl do množiny $\{0, 1\}$:

- ak p je výrokovologický atóm, potom $val_{\mathcal{I}}(p) = 1$ práve vtedy, keď $p \in m$,
- $val_{\mathcal{I}}(\neg\phi) = 1$ práve vtedy, keď $val_{\mathcal{I}}(\phi) = 0$,
- $val_{\mathcal{I}}(\phi \wedge \psi) = 1$ práve vtedy, keď $val_{\mathcal{I}}(\phi) = 1$ a $val_{\mathcal{I}}(\psi) = 1$,
- $val_{\mathcal{I}}(K\phi) = 1$ práve vtedy, keď $\phi \in S$.

■

Vidíme teda, že subjektívna formula je pravdivá vtedy, keď argument jej operátora patrí do špecifikovanej množiny presvedčení.

Autoepistemické interpretácie (AE-interpretácie) sa konštruujú obvykle pre pevné S (k danému S volíme rôzne a viaceré m).

Definícia I.3 Nech je daná množina presvedčení S a autoepistemická teória T. Budeme hovoriť, že AE-interpretácia $\mathcal{I} = (m, S)$ je modelom T práve

[2] Pripomeňme, že alternatívne a ekvivalentne môžeme hovoriť o dvojici množín atómov (T, F) takých, že $T \cap F = \emptyset$ a $T \cup F$ obsahuje všetky objektívne atómy jazyka.

vtedy, keď $\forall \phi \in T\ val_{\mathcal{I}}(\phi) = 1$. (Niekedy sa hovorí o S-modeli, zdôrazňuje sa tak, že množina presvedčení S je pevne zvolená.)

Ďalej budeme hovoriť, že formula ϕ S-vyplýva z autoepistemickej teórie T ($T \models_S \phi$), ak ϕ je pravdivá v každom S-modeli T. ■

Nie každá množina presvedčení (a teda nie každá autoepistemická interpretácia) je rozumná.

Príklad I.4 Predstavme si, že sme prijali autoepistemickú teóriu

$$T = \{p;\ \ \neg Kp \Rightarrow \neg p\}.$$

Keď akceptujeme túto teóriu, množina presvedčení $S = \{\neg p\}$ nebude rozumná, je v konflikte s prijatou teóriou T. Podobne nebude rozumná množina presvedčení, ktorá obsahuje ϕ spolu s $\phi \Rightarrow \psi$, ale neobsahuje ψ.

Ak berieme do úvahy hľadisko agenta s dokonalou introspekciou, množina presvedčení, ktorá obsahuje ϕ, ale neobsahuje $K\phi$ (neobsahuje ϕ a neobsahuje ani $\neg K\phi$), tiež nie je rozumná. ■

Na dôkladnú definíciu „rozumnej“ množiny presvedčení sa pripravíme definíciou dvoch dôležitých pojmov:

Definícia I.5 Nech je daná množina presvedčení S a autoepistemická teória T. Budeme hovoriť, že

- S je korektná vzhľadom na T, ak každý S-model T je aj modelom S,
- ak pre každú formulu ϕ, pravdivú v každom S-modeli, platí, že $\phi \in S$, potom je S úplná.

■

Podmienka korektnosti vyjadruje, že S rešpektuje autoepistemickú teóriu T. Podmienka úplnosti zasa hovorí, že S je tak nasýtená, že obsahuje každú formulu, ktorá je pravdivá na základe S.

Príklad I.6 $S = \{\neg p\}$ z príkladu I.4 nie je korektná (vzhľadom na dané T), pretože autoepistemická interpretácia $M = (\{p\}, S)$ je modelom T, ale nie je modelom S.

Ak S obsahuje ϕ, ale neobsahuje $K\phi$, potom nie je úplná: $K\phi$ je totiž pravdivá v každom S-modeli (podobne $KK\phi$, atď. sú pravdivé v každom S-modeli). Podobne, ak ϕ a $\phi \Rightarrow \psi$ patria do S a sú pravdivé v každom S-modeli, aj ψ je pravdivé v každom S-modeli. Preto by malo patriť do „rozumnej“ množiny presvedčení S. ■

Predstave o rozumnej množine presvedčení zodpovedá pojem expanzie:

Definícia I.7 Množina presvedčení S je *stabilná expanzia* (AE-expanzia) autoepistemickej teórie T, ak platí

- S je korektná vzhľadom na T,
- S je úplná,
- $T \subseteq S$.

■

Tvrdenie I.8 *Množina autoepistemických formúl S je AE-expanzia autoepistemickej teórie T práve vtedy, keď* $S = \{\phi : T \models_S \phi\}$.

Idea dôkazu:
$\Rightarrow$
Pre $\phi \in S$ dostávame $T \models_S \phi$ na základe korektnosti a pre ϕ spĺňajúce $T \models_S \phi$ platí $\phi \in S$ na základe úplnosti a $T \subseteq S$.
$\Leftarrow$
Korektnosť a inklúziu $T \subseteq S$ dostaneme ľahko. Na základe korektnosti preveríme aj úplnosť. ■

Po tejto sémantickej charakterizácii introspekcie (rozumnej množiny presvedčení, vyhovujúcej požiadavkám na introspekciu ideálne racionálneho agenta), prejdime k syntaktickej charakterizácii. Budeme hľadať také vyjadrenie, ktoré výlučne na základe tvaru jazykových výrazov splní sémantické podmienky, ktoré sme práve formulovali. Dostávame sa do situácie podobnej tej, ktorú sme už videli pri extenziách defaultových teórií (v časti 4.1). Vo všeobecnosti sú možné viaceré množiny presvedčení akceptovateľných pri danej autoepistemickej teórii, neexistuje konštruktívna metóda ich hľadania a ich definícia je založená na pevnom bode nejakého operátora.

Začneme defínicou stabilnej množiny presvedčení, ideu už poznáme.

Definícia I.9 Množina presvedčení S je stabilná práve vtedy, keď platí

- $S = Cn_{VL}(S)$,
- ak $\phi \in S$, potom $K\phi \in S$,
- ak $\phi \notin S$, potom $\neg K\phi \in S$.

■

Každú autoepistemickú teóriu môžeme automaticky pokladať za množinu presvedčení, preto budeme hovoriť aj o stabilných autoepistemických teóriách.

Tvrdenie I.10 *Ak S je stabilná a konzistentná množina presvedčení, potom platí:*

a) *ak* $K\phi \in S$, *potom aj* $\phi \in S$,

b) *ak* $\neg K\phi \in S$*, potom* $\phi \notin S$.

Idea dôkazu: **a)** – predpokladajme, že $K\phi \in S$ a $\phi \notin S$, potom $\neg K\phi \in S$. Narušili sme teda predpoklad konzistentnosti; podobne sa dokáže **b)**. ■

Vidno teda, že stabilné a konzistentné autoepistemické teórie sú nasýtené v tomto zmysle slova: pre každú formulu ϕ (objektívnu i subjektívnu) platí, že buď $K\phi$ alebo (vo vylučujúcom zmysle slova) $\neg K\phi$ patrí do tejto teórie.

Navyše, stabilné autoepistemické teórie sú determinované objektívnymi formulami:

Tvrdenie I.11 *Nech* T *a* T' *sú stabilné autoepistemické teórie,* T_0 *a* T_0' *sú množiny všetkých ich objektívnych formúl. Potom platí:*
Ak $T_0 = T_0'$*, potom* $T = T'$.

Dôkaz v [Moo 85] alebo v [Luk 90].

V predchádzajúcom tvrdení sa spomedzi všetkých formúl autoepistemickej teórie pozornosť venovala iba ich objektívnym formulám. Takéto presúvanie pozornosti možno zovšeobecniť tak, že zavedieme pojem modálnej hĺbky.

Definícia I.12 Autoepistemická formula ϕ má modálnu hĺbku

$$\begin{array}{rl} 0, & \text{ak } \phi \text{ je objektívna formula,} \\ m+1, & \text{ak } \phi \text{ je tvaru } K\psi \text{ alebo } \neg K\psi \text{ a } \psi \text{ má modálnu hĺbku } m. \end{array}$$

Fakt, že ϕ má modálnu hĺbku m značíme ako $d(\phi) = m$. Ak S je množina autoepistemických formúl, potom $S_m = \{\phi \in S : d(\phi) \leq m\}$. ■

Iteráciu cez modálnu hĺbku využijeme v nasledujúcej konštrukcii. Predpokladajme jazyk J, fixovaný výrokovologický jazyk obohatený o K a s pojmom autoepistemickej formuly, ktorý sme definovali vyššie. Pod J_m rozumieme množinu tých autoepistemických formúl z J, ktorých modálna hĺbka je nanajvýš m.

Tvrdenie I.13 *Nech* X *je deduktívne uzavretá množina objektívnych formúl (t.j.* $X = Cn_{VL}(X)$*). Definujme operátor* Z *tak, že* $Z(X) = \bigcup_{i=0}^{\infty} S^i$*, kde*

$$\begin{array}{rcl} S^0 & = & X, \\ S^{i+1} & = & \{\phi : \phi \in Cn_{VL}(S^i \cup \{K\psi : \psi \in S^i\} \cup \{\neg K\psi : \psi \notin J_i \setminus S^i\})\}_{i+1}. \end{array}$$

Potom $Z(X)$ *je stabilná množina presvedčení a* $Z(X)_0 = X$.

Zdá sa, že máme konštruktívnu metódu pre hľadanie stabilných množín presvedčení, rešpektujúcich danú autoepistemickú teóriu T. Nie je to však tak. Vo vyššie uvedenom tvrdení máme voľnosť pri výbere množiny objektívnych formúl X. Prejdeme k syntaktickej definícii autoepistemickej expanzie.

Tvrdenie I.14 *Nech T je autoepistemická teória. Potom množina autoepistemických formúl S je autoepistemickou expanziou T práve vtedy, keď*

$$S = Cn_{VL}(T \cup \{K\phi : \phi \in S\} \cup \{\neg K\phi : \phi \notin S\}).$$

Dôsledok I.15 *S je autoepistemickou expanziou teórie T práve vtedy, keď*

$$S = \{\psi : (T \cup \{K\phi : \phi \in S\} \cup \{\neg K\phi : \phi \notin S\}) \models \psi\}.$$

Teda, autoepistemická expanzia je množina autoepistemických formúl, generovaná danou teóriou T tak, že T sa dopĺňa o K-uzáver všetkých akceptovaných formúl a o $\neg K$-uzáver všetkých neakceptovaných formúl a o ich výrokovologické dôsledky.

Autoepistemická teória nemusí mať expanziu. Sú však triedy teórií, ktoré nutne majú autoepistemickú expanziu.

Tvrdenie I.16 *Každá konzistentná objektívna teória má jedinú stabilnú expanziu.*

Ďalším príkladom triedy teórií, ktoré nutne majú autoepistemickú expanziu, sú stratifikované autoepistemické teórie [Gel 87]. Zavedieme si ich:

Zúžime jazyk: budú nás zaujímať iba klauzy tvaru $H \leftarrow T$, kde T je (prípadne aj prázdna) konjunkcia literálov a H je (prípadne aj prázdna) disjunkcia atómov. Každý z disjunktov nazývame konzekventom.

Definícia I.17 Autoepistemická teória T je stratifikovaná, ak existuje jej rozklad $(T_0, \ldots, T_n)$,[3] pre ktorý platí:

- T_0 obsahuje iba objektívne formuly (môže byť aj prázdna),
- T_k nemôže obsahovať klauzy s prázdnym konzekventom, ak $k > 0$,
- ak sa výroková premenná p vyskytuje v konzekvente klauzy z T_k, potom
  - p ani $\neg p$ sa nevyskytujú v $T_0 \cup \cdots \cup T_{k-1}$ ako disjunkt alebo konjunkt,
  - nech ϕ je formula, ktorá obsahuje atóm p, potom $K\phi$ ani $\neg K\phi$ sa nevyskytujú v $T_0 \cup \cdots \cup T_k$ ako disjunkt alebo konjunkt.

■

[3] Pripomeňme si definúciu rozkladu: $T = T_0 \cup \cdots \cup T_n$, kde pre každé $i \neq j$ je $T_i \cap T_j = \emptyset$.

Stratifikáciou zavádzame syntaktické obmedzenie na autoepistemické formuly (podobne ako pri stratifikácii logických programov). Stratifikované autoepistemické teórie tvoria podmnožinu všetkých autoepistemických teórií. Po prvé berieme do úvahy iba klauzy tvaru $H \leftarrow T$, charakterizované vyššie. Po druhé, formuly takejto teórie sa musia dať roztriediť do vrstiev. V základnej vrstve sú objektívne formuly. Subjektívne formuly nemôžu mať prázdny konzekvent. Ak subjektívna formula vrstvy k obsahuje v konzekvente premennú p, potom sa premenná p (ani negovaná) nesmie vyskytnúť ako samostatný disjunkt alebo konjunkt v predchádzajúcich vrstvách. Ak je táto premenná p argumentom nejakého subjektívneho literálu, potom tento subjektívny literál sa nemôže vyskytnúť v predchádzajúcich vrstvách, ale ani vo vrstve k.

Budeme hovoriť, že stupeň výrokovej premennej p je k, ak sa p vyskytuje v konzekvente nejakej klauzy z T_k. Ak sa p nevyskytuje v konzekvente žiadnej klauzy, potom jeho stupeň je 0. Stupeň výrokovej premennej p budeme značiť ako $D(p)$. Formálne, D je zobrazenie z výrokových premenných do prirodzených čísel.

Očividne, každá výroková premenná má presne jeden stupeň. Stupeň objektívnej formuly ϕ sa definuje ako maximálny stupeň výrokových premenných, vyskytujúcich sa vo ϕ.

Stabilnú expanziu stratifikovanej autoepistemickej teórie možno definovať konštruktívnym spôsobom. Induktívna konštrukcia stabilnej expanzie:

$$
\begin{aligned}
K_0 &= Cn_{VL}(T_0), \\
K_{m+1} &= Cn_{VL}(K_m \cup \{Kp : D(p) = m \wedge p \in K_m\} \\
&\qquad\qquad \cup \{\neg Kp : D(p) = m \wedge p \notin K_m\} \cup T_{m+1}).
\end{aligned}
$$

Tvrdenie I.18 *Každá konzistentná a stratifikovaná autoepistemická teória T má jedinú stabilnú expanziu.*

Dodatok J

Cirkumskripcia

Ideu cirkumskripcie formalizoval McCarthy [McC 80, McC 86] a k jej kryštalizácii výrazne prispel Lifschitz [Lif 88, Lif 95].

Motiváciou pre túto formalizáciu je nutnosť riešiť nasledujúci problém: Predstavme si, že nejakým spôsobom je špecifikovaná úloha pre inteligentný systém. Ak je táto úloha zasadená do reálneho prostredia, môže vzniknúť potreba uistiť sa, že žiadna abnormalita tohto prostredia neskomplikuje úlohu (napríklad, v známej úlohe o prievozníkovi nebudú chýbať veslá). Samozrejme, takýchto komplikácií môže byť neohraničené množstvo. Preto treba nejako zaznamenať, že nič iné sa o úlohe a danom prostredí nepredpokladá, iba to, čo je explicitne sformulované. Inými slovami, treba nejako stanoviť, že zamýšľaný model opisu situácie a úlohy je minimálny model.

Cieľom cirkumskripcie je určiť práve minimálne modely. Intuitívne, cirkumskripcia predikátu p vzhľadom na databázu DB (množinu viet) je nejaká formula, ktorá má tú vlastnosť, že keď ju dodáme k DB, modelmi výslednej množiny viet budú iba minimálne modely DB. Minimalizácia, minimálny model v úlohe štandardného modelu, sú obvyklým spôsobom sémantickej charakterizácie nemonotónneho usudzovania. Ide pri nej o to, aby sa na základe viet nejakej dátabázy nepredpokladalo viac, než je nevyhnutne treba.

Začneme najjednoduchším prípadom: Cieľom je minimalizovať interpretáciu jedného predikátu. Nech sa predikát p vyskytuje v teórii T. Nech interpretácia M je modelom teórie T. Nech p je interpretovaný reláciou ρ v M. Ak chceme minimalizovať interpretáciu p, usilujeme sa o to, aby sa p nedalo v žiadnom „rozumnom“ modeli teórie T interpretovať reláciou, ktorá by bola podmnožinou ρ. To znamená, pripustíme iba také modely teórie T, v ktorých by sa p intepretovalo minimálnou možnou reláciou.

Ako príklad možno spomenúť predikát ab, pomocou ktorého sme formalizovali viaceré príklady v jadre textu. Formulácie týchto príkladov mlčky predpokladali, že abnormality treba pripustiť v minimálnom potrebnom počte (nechceme prijať abnormalitu vtedy, ak nie sú explicitné dôvody pre jej prijatie). Nasledujúca definícia ukáže techniku, pomocou ktorej možno minimalizovať interpretáciu nejakého predikátu.

Definícia J.1 ($CIRC(DB;p)$) Nech p^* je predikátový symbol rovnakej arity ako p. Nech $DB(p^*)$ získame z DB nahradením každého výskytu p výskytom p^*. Potom

$$\begin{aligned} CIRC(DB;p) \quad \equiv \quad & (DB \wedge \forall p^*(DB(p^*) \\ & \wedge \forall X(p^*(X) \Rightarrow p(X)) \Rightarrow (\forall X p(X) \Rightarrow p^*(X)))). \end{aligned}$$

■

Formula $CIRC(DB;p)$ sa nazýva *cirkuskripčnou* formulou. $CIRC(DB;p)$ nie je definovaná pomocou prvorádovej formuly.[1] Kvantifikovaný je predikátový symbol p^*.

Intuitívne, cirkumskripčná formula hovorí: Platí to, čo tvrdí DB. Navyše, ak v DB predikát p nahradíme iným predikátom tej istej arity tak, aby databáza po tejto substitúcii, $DB(p^*)$, ostala naďalej pravdivá, interpretácia nového predikátu nemôže byť podmnožinou interpretácie pôvodného predikátu p.

Zhutnená verzia zápisu tejto formuly využíva tieto skratky:

$$\begin{aligned} p \leq p^* \quad & \equiv \quad \forall X\ p(X) \rightarrow p^*(X), \\ p = p^* \quad & \equiv \quad \forall X\ p(X) \leftrightarrow p^*(X), \\ p < p^* \quad & \equiv \quad (p \leq p^*\ \wedge\ p \neq p^*. \end{aligned}$$

Teraz syntaktické varianty cirkumskripčnej formuly:

$$\forall p^*(DB(p^*) \wedge p^* \leq p \Rightarrow p \leq p^*),$$

$$\forall p^*(DB(p^*) \Rightarrow \neg(p^* < p)).$$

Prirodzený krok od minimalizácie jediného predikátu je minimalizácia množiny predikátov.

Definícia J.2 Nech $P = \{p_1, \ldots, p_n\}$ je množina predikátov. *Paralelná cirkumskripcia* sa definuje pomocou formuly:

$$CIRC(DB, P) \equiv DB(P) \wedge \neg(\exists P^* DB(P^*) \wedge P^* < P),$$

kde P^* je množina predikátov $\{p_1^*, \ldots, p_n^*\}$ a pre každé i majú p_i a p_i^* rovnakú aritu. Relácia $<$ porovnáva množiny predikátov po dvojiciach tak, že $P^* < P$ práve vtedy, keď pre každé i je $p_i^* < p_i$. ■

[1] ⟨ predikátový počet vyšších rádov : dodatok B ⟩

Doteraz sme minimalizovali niektoré predikáty a o ostaných sme vyžadovali, aby boli pevne, fixne interpretované. Niektoré aplikácie však vyžadujú, aby sa v „rozumných“ modeloch teórie T spolu s minimalizáciou niektorých predikátov povolila variabilnosť interpretácie iných predikátov. Rozhodnutie, ktoré predikáty minimalizovať, ktoré fixovať a u ktorých dovoliť variablitu, sa nazýva *cirkumskričnou stratégiou* (circumscription policy). Takéto rozhodnutie je nevyhnutné pre adekvátnu formalizáciu pri každej konkrétnej aplikačnej úlohe.

Definícia J.3 Paralelná cirkumskripcia s predikátmi Z, ktorých interpretácia je variabilná, sa vyjadruje formulou

$$CIRC(DB; P; Z) \equiv DB(P; Z) \wedge \neg(\exists P^* \exists Z^* DB(P^*; Z^*) \wedge P^* < P).$$

■

V súvislosti s negáciou v logickom programovaní je zaujímavá cirkumskripcia po bodoch a s prioritami [Lif 88], pozri časť 6.5.

Definícia J.4 *Cirkumskripcia po bodoch*: Nech $P = \{p_1, \cdots, p_m\}$ je množina predikátov. Potom formula

$$\begin{aligned} CIRC(DB; p_i \in P; P) \quad \equiv \quad & DB(P) \wedge \forall x, p'_1, \cdots, p'_m \neg(p_i(x) \wedge \neg p'_i(x) \wedge \\ & DB(p'_1, \cdots, p'_m)) \end{aligned}$$

vyjadruje cirkumskripciu po bodoch. ■

Táto formula minimalizuje p_i, jeden z predikátov z množiny predikátov, ktorých interpretácia sa môže meniť. Podľa definície neexistujú predikáty p'_i spĺňajúce DB, ktorých interpretácia by bola podmnožinou interpretácie p_i.

Nasledujúci príklad ilustruje cirkumskripciu po bodoch a motivuje aj potrebu tento pojem nejak modifikovať (skomplikovať).

Príklad J.5 Budeme analyzovať logický program DB a predpokladať, že jedinými objektami v univerze sú tie, ktorých mená sa vyskytujú v programe (v našom príklade to sú 1 a 2). Nech $DB =$

$$\begin{aligned} \{p(1) \quad & \leftarrow \quad , \\ q(2) \quad & \leftarrow \quad , \\ r(X) \quad & \leftarrow \quad \neg q(X)\}. \end{aligned}$$

Budeme minimalizovať p.

$$CIRC(DB; p; \{p, q, r\}\} \equiv DB \wedge \forall X(p(X) \equiv X = 1).$$

Podľa toho, ak platí $p(X)$ a platí aj DB, tak X musí byť 1. Podobne:

$$CIRC(DB; q; \{p, q, r\}\} \equiv DB \wedge \forall X(q(X) \equiv X = 2).$$

Minimalizácia r však spôsobuje problémy.

$$CIRC(DB; r; \{p, q, r\}\} \equiv DB \wedge \forall X \neg r(X).$$

Dôvod, prečo je posledná cirkumskripcia ekvivalentná konjunkcii dátabázy a formuly $\forall X \neg r(X)$ je v tom, že q má variabilnú interpretáciu, možno teda nájsť q' také, že $q'(1) \wedge q'(2)$. Ak chceme minimalizovať r, môžeme q ľubovoľne meniť.

Teda, model všetkých troch uvedených cirkumskripčných formúl je

$$\{p(1), q(2)\},$$

ten však nie je modelom DB. Túto nepríjemnosť zapríčiňuje konflikt medzi minimalizáciou q a minimalizáciou r. Je zrejmé, že s minimalizáciou q by malo „rásť“ r a naopak. Vzniká teda prirodzená otázka: ak vzniká konflikt medzi dvojicou minimalizácií, ktorú z nich uprednostniť?

Hľadajme riešenie, ktoré by dovoľovalo prijať $DB \wedge \forall X(r(X) \equiv X = 1)$, teda chceme chápať r ako komplement q, presne podľa definície r v programe DB. Ak chceme minimalizovať r, potrebujeme mať predtým minimalizované q. Inými slovami: q je preferovanejšie ako r vzhľadom na minimalizáciu. ■

Definícia J.6 Majme reflexívnu a tranzitívnu reláciu preferencie $\preceq$. *Cirkumskripciu po bodoch s prioritami* vyjadruje formula

$$CIRC(DB; p; \{q : p \preceq q\}),$$

definovaná ako v definícii J.4, pričom množina variabilných predikátov $\{q : p \preceq q\}$ je množina tých predikátov, ktoré sú preferovanejšie než predikát p, ktorý máme minimalizovať. ■

Príklad J.7 Vráťme sa k príkladu J.5. Preferenciu medzi predikátmi zaveďme podľa závislostného grafu: $p \preceq q$ práve vtedy, keď v závislostnom grafe ide cesta od A ku B (A je v hlave nejakého pravidla a B v tele toho istého pravidla). Keďže požadujeme reflexívnu reláciu, uvažujeme jej reflexívny uzáver (pre každé A platí $A \preceq A$).

V našom príklade platí (okrem reflexívnych prípadov) iba $q \preceq r$. To znamená, keď chceme rešpektovať priority, budeme vyhodnocovať cirkumskripčné formuly $CIRC(DB; p; p)$, $CIRC(DB; q; \{q, r\})$ a $CIRC(DB; r; r)$. Pri minimalizácii p nedovolíme variabilitu žiadneho iného predikátu. Podobne pre minimalizáciu r: q tu bude musieť byť fixované. Toto q zafixujeme na základe

$$\begin{aligned} CIRC(DB; q; \{q, r\}) &\equiv DB(\{q, r\}) \wedge \forall X, q, q' \neg(q(X) \wedge \neg q'(X) \wedge DB(q')) \\ &\equiv (\forall X(q(X) \equiv X = 2)). \end{aligned}$$

■

Literatúra

[ABM 98] Antoniou, G., Billington, D., Maher, M.J. *Sceptical logic programming based default reasoning - Defeasible logic rehabilitated.* Proc. of Commonsense'98, 1-19

[ABo 94] Apt, K.R., Bol, R.N. *Logic Programming and Negation: A Survey.* J.Logic Programming, 1994:19,20:5-71

[ABW 87] Apt, K.R., Blair, H.A., Walker, A. *Towards a Theory of Declarative Knowledge.* In [Min 87].

[AFr 99] Anthony, S., Frisch, A. *Using meta-languages for learning.* P.Flach, N.Lavrac (eds.): Area Meeting of CompulogNet: Computational Logic and Machine Learning, Prague 1999, 4-7

[AGG 99] Antoniou, G., Ghose, A., Goebel, R. *On the Dynamics of Default Reasoning.* Proc. 5th European Conf. on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQUARU'99), LNAI 1638, Springer 1999

[AKG 91] Abiteboul, S., Kanellakis, P., Grahne, G. *On the representation and querying of sets of possible worlds.* Theoretical Computer Science 78 (1991), 159-187

[Ali 00] Aliseda, A. *Abductive reasoning as epistemic change.* [FlK 00]

[AnW 97] Antoniou, G., Williams, M.A. *Reasoning with Incomplete and Changing Information.* J. of Information Science, 99, 1 and 2:83-99, 1997

[APe 96] Alferes, J., Pereira, L. *Update-programs can update programs.* Springer, 1996

[APP 96] Alferes, J., Pereira, L., Przymusinski, T. *Belief Revision in Non-Monotonic Reasoning and Logic Programming.* Fundamenta Informaticae 1(1996)

[APP 98] Alferes, J., Leite, J., Pereira, L., Przymusinska, H., Przymusinski, T. *Dynamic logic programming.* Proc. of KR'98, Morgan Kaufmann, 1998

[Apt 88] Apt K.R. *Introduction to Logic Programming.* Technical Report, Amsterdam 1988

[BaG 99] Baumgartner, R., Gottlob, G. *On the complexity of model checking for propositional default logics: new results and tractable cases.* In

[PAI 99], 64-69

[BaM 90] Bain, M., Muggleton, S. *Non-monotonic learning.* In: Hayes, Michie, Tyugu (eds.): Machine Intelligence 12, Oxford, 1990

[Bar 98] Baral, C. *Abductive reasoning through filtering.* `http://www.cs.utep.edu/chitta/papers/filter.ps`

[BDK 97] Bondarenko, A., Dung, P., Kowalski, R., Toni, F. *An abstract, argumentation-theoretic framework for default reasoning.* Artificial Intelligence 93 (1997), No. 1-2, 63-101

[BDP 97] Brass, S., Dix, J., Przymusinski, T. *Super Logic Programs.* Technical Report, Universität Hannover, Institut für Informatik, 1997

[Bel 76] Belnap, N.D. *How a computer should think.* In: Contemporary Aspects of Philosophy. Proceedings of the Oxford International Symposium, 1976.

[BeS 94] Besnard, P., Schaub, T. *Possible worlds semantics for default logics.* Fundamenta Informaticae 21 (1994), 39-66

[BGe 94] Baral, C., Gelfond, M. *Logic Programming and Knowledge Representation.* J.Logic Programming, 1994:19,20: 73-148

[BGe 00] Baral, C., Gelfond, M. *Reasoning agents in dynamic domains.* `http://earth.cs.ttu.edu/~mgelfond/papers/lbai.ps`

[BiF 87] Bidoit, N., Froidevaux, C. *General logic databases: default logic semantics and stratification.* Journal of Information and Computation

[BFL 85] Brachman, R.J., Fikes, Levesque, H.J. *KRYPTON: a functional approach to knowledge representation.* In [BLe 85]

[BLe 85] Brachman, R.J., Levesque, H.J. (eds.) *Readings in Knowledge Representation.* Morgan Kaufmann Publishers, 1985.

[BLR 89] Brachman, R.J., Levesque, H.J., Reiter, R. *Principles of Knowledge Representation and Reasoning.* Morgan Kaufmann Publishers, 1989

[Boc 98] Bochman, A. *Belief contraction as nonmonotonic inference.* KR'98, Belief revision workshop

[Boc 99] Bochman, A. *Credulous nonmonotonic inference.* Proc. of the IJ-CAI'99

[BoJ 80] Boolos, G., Jeffrey, R. *Computability and Logic.* Cambridge University Press, 1980

[Bon 97] Bonner, A. *A logical semantics for hypothetical rulebases with deletion.* The Journal of Logic Programming, vol. , No. , 1997, 119-170

[Bou 94] Boutilier, C. *Unifying default reasoning and belief revison in a modal framework.* Artificial Intelligence 68 (1994), 33-85

[Bou 95] Boutilier, C., Becher, V. *Abduction as Belief Revision.* Artificial Intelligence 77 (1995), 43-94

[Bra 85] Brachman, R.J. *On the epistemological status of semantic networks.* In [BLe 85].

[Bre 89] Brewka, G. *Preferred subtheories: an extended logical framework for default reasoning.* Proc. of IJCAI-89, Morgan Kaufman Publishers, Inc., 1989

[Bre 91] Brewka, G. *Cumulative default logic.* Artificial Intelligence 50 (1991), 183-205

[Bry 90] Bry, F. *Intesional updates: abduction via deduction.* Proc. of 7th Logic Programming Int. Conf. 1990

[Buc 97] Buccafurri, F., Eiter, T., Gottlob, G., Leone, N. *Enhancing symbolic model checking by AI techniques.* IFIG Research Report 9701, September 1997

[Buk 85] Bukovský, L. *Množiny a všeličo okolo nich.* Alfa, Bratislava, 1985

[Bun 88] Buntine, W. *Generalized subsumption and its application to induction and redundancy.* Artificial Intelligence 36 (1988), 149-176

[Byl 91] Bylander, T. et al. *The computational complexity of abduction.* Artificial Intelligence 49 (1991), 25-60

[CaS 93] Cadoli, M., Schaerf, M. *A survey of complexity results for nonmonotonic logics.* The Journal of Logic Programming 1993:17:127-160

[CDS 95] Cadoli, M., Donini, F.M., Schaerf, M. *Is intractability of nonmonotonic reasoning a real drawback?* Artificial Intelligence 88, 1-2, 215-251

[CES 86] Clarke, E., Emerson, E., Sistla, A. *Automatic verification of finite-state concurrent systems using temporal logic specifications.* ACM Transactions on Programming Languages and Systems, vol. 8, No. 2, April 1986, 244-263

[ChH 80] Chandra, A.K., Harel, D.. *Computable queries for relational databases.* Journal of Computer and System Sciences 21, 156-178 (1980)

[ChH 82] Chandra, A.K., Harel, D. *Structure and complexity of relational queries.* Journal of Computer and System Sciences, 25, 99-128 (1982)

[ChH 85] Chandra, A.K., Harel, D. *Horn clause queries and generalizations.* J.Logic Programming 1985:1:1-15

[ChL 73] Chang, Ch., Lee, R. *Symbolic Logic and Mechanical Theorem Proving.* Academic Press, New York, 1973

[Cho 94] Cholewiński, P. *Stratified deafault theories.* Proceedings of CSL, 1994

[Cho 95] Cholewiński, P. *Reasoning with stratified default theories.* In [LPN 95]

[Chr 00] Christiansen, H. *Abduction and induction combined in a metalogical framework.* In [FlK 00]

[CKu 89] Crawford, J.M., Kuipers, B. *Towards a theory of access-limited knowledge for knowledge representation.* In [BLR 89].

[CMC 87] Cercone, N., McCalla, G. *The knowledge frontier. Essays in the representation of knowledge.* Springer, 1987

[CMM 95] Cholewiński, P., Marek, W., Mikitiuk, A., Truszczynski, M. *Experimenting with nonmonotonic reasoning.* Proc. of 12th Int. Conf. on Logic Programming, MIT Press, 1995

[CMT 95] Cholewiński, P., Marek, W., Truszczynski, M. *Deafult reasoning system DeReS.*

[Cod 81] Codd, E.F. *A relational model of data for large shared data banks.* CACM, 24, 1981

[Coh 89] Cohn, A.G. *On the appearance of sortal literals: a non substitutional framework for hybrid reasoning.* In [BLR 89]

[CPa 99] Chopra, S., Parikh, R. *An inconsistency tolerant model for belief representation and belief revision.* Proc. of the IJCAI'99, 192-197

[Del 99] Delgrande, J. *Considerations on a similarity-based approach to belief change.* Proc. of the IJCAI'99, 180-185

[Den 94] Denecker, M. *A declarative perspective on abductive logic programming.* `http://www.cs.engr.uky.edu/~lpnmr/papers.html`

[dRB 92] De Raedt, L., Bruynooghe, M. *Belief updating from integrity constraints and queries.* Artificial Intelligence 53 (1992), 291-307

[DGa 84] Dowling, W., Gallier, J. *Linear-time algorithms for testing the satisfiability of propositional Horn formulae.* J.Logic Programming 3 (1984), 267-284

[DiK 96] Dimopoulos, Y., Kakas, A. *Abduction and Induction: an AI perspective.* ECAI'96 workshop on abduction and induction

[Dix 92] Dix, J. *Deafult theories of Poole-type and a method for constructing cumulative versionof default logic.* Proc. ECAI 92, John Wiley & sons, 1992

[Dix 95a] Dix, J. *A classification theory of semantics of normal logic programs. I. Strong Properties.* Fundamenta Informaticae 22 (1995), No. 3, 227-255

[Dix 95b] Dix, J. *A classification theory of semantics of normal logic programs. II. Weak Properties.* Fundamenta Informaticae 22 (1995), No. 3, 257-288

[dKl 86] deKleer, J. *An Assumption Based Truth Maintenance System.* Artificial Intelligence 28 (1986)

[DLe 82] Davis, R., Lenat,D.B. *Knoweldge-Based Systems in Artificial Intelligence.* McGraw-Hill, 1982

[DMy 86] Delgrande, J., Mylopoulos, J. *Knowledge representation: features of knowledge.* In: Fundamentals of Artificial Intelligence, eds. Bibel, Jorrand, Springer Verlag 1986

[DoF 99] Downey, R., Fellows, M. *Parametrized Complexity.* Springer, 1999

[DoP 91] Doyle, J., Patil, R.S. *Two dogmas of knowledge representation: language restrictions, taxonomic classification, and the utility of representation services.* Artificial Intelligence, vol. 48, No 3, April 1991, 261-297

[Doy 79] Doyle, J. *A truth maintenance system.* Artificial Intelligence 12 (1979), 231-272

[DSJ 94] Delgrande, J., Schaub, T., Jackson, W. *Alternative approaches to default logic.* Artificial intelligence 70 (1994), 167-237

[Dun 91] Dung, P. *Negations as hypotheses: an abductive foundation for logic programming.* Proc. of Int. Conf. on Logic Programming, 1991, 3-17

[EBB 89] Etherington, D., Borgida, A., Brachman, R., Kautz, H. *Vivid knowledge and tractable reasoning: preliminary report.* In: Proceedings of IJCAI-89. Morgan Kaufmann Publ, 1989, 1146-1152

[EGL 97] Eiter, T., Gottlob, G., Leone, N. *Abduction from logic programs: Semantics and complexity.* Theoretical Computer Science 189 (1997), 129-177

[EiG 95] Eiter, T., Gottlob, G. *The complexity of logic-based abduction.* Journal of the Association for Computing Machinery, vol. 42, No. 1, January 1995, 3-42

[EKo 89] Eshghi, K., Kowalski, R. *Abduction compared with negation by failure.* Proc. 6th Logic Programming International Conference, 1989, 234-255

[Elk 90] Elkan, C. *A rational reconstruction of nonmonotonic truth maintenance systems.* Artificial Intelligence 43 (1990), 219-234

[Eng 96] Engelfriet, J. *Minimal temporal epistemic logic.* Notre Dame Journal of Formal Logic vol. 37, No 2, Spring 1996, 233-259

[EnT 95] Engelfriet, J., Treur, J. *Temporal theories of reasoning.* J. of Applied and Non-Classical Logics vol. 5, No. 1, 1995, 97-119

[EnT 96] Engelfriet, J., Treur, J. *Executable temporal logic for non-monotonic reasoning.* J. Symbolic Computation 22 (1996), 615-625

[Eth 88] Etherington, D. *Reasoning with incomplete information.* Morgan Kauffman Publ., 1998

[Fah 79] Fahlman, S.E. *NETL: a system for representing and using real-world knowledge.* MIT Press, 1979

[FGh 99] Fisher, M., Ghidini, C. *Programming resource-bounded deliberative agents.* In [PAI 99], 200-205

[FiG 96] Finger, M., Gabbay, D. *Combining temporal logic systems.* Notre Dame Journal of Formal Logic vol. 37, No. 2, Spring 1996, 204-232

[Fla 98] Flach, P. *Comparing consequence relations.* Proc. of the Int. Conf. on the Principles of the Knowledge Representation and Reasoning, KR'98, Morgan Kaufmann, 1998

[FlK 00] Flach, P., Kakas, A. (eds.) *Abduction and Induction. Essays on their relation and integration.* Kluwer Academic Publishers, 2000

[FPL 99] Faber, W., Pfeifer, G., Leone, N. *Pushing goal derivation in DLP computations.* In [LPN 99], 177-191

[FaV 84] Fagin, R., Vardi, M. *The theory of data dependencies – an overview.* ICALP 1984, 1-22

[FPM 92] Frawley, W.J., Piatetsky-Shapiro, G., Matheus,C.J., *Knowledge Discovery in Databases: An Overview.* AI Magazine, Fall 1992, vol.13, No 3, 57-70

[Fri 89] Frisch, A.M. *A general framework for sorted deduction: fundamental results on hybrid reasoning.* In [BLR 89].

[Gab 85] Gabbay, D. *Theoretical foundations for non-monotonic reasoning in expert systems.* K.R.Apt (ed.), Logics and models of concurrent systems, Springer, 1985, 439-457

[GaH 93] Gabbay, D., Hunter, A. *Making inconsistency respectable: Part 2 – Meta-level handling of inconsistency.* Proc. of ESCQARU, Springer, LNCS 747, 129-136

[GaJ 79] Garey, M.R., Johnson, D.S. *Computers and intractability.* W.H.Freeman and comp., 1979

[GaR 84] Gabbay, D., Reyle, U. *N-Prolog: an extension of Prolog with hypothetical implications. I.* The Journal of Logic Programming, 1984, 4, 119-155

[Gaš 97] Gašo, J. *Artificial Mathematician – analýza a reimplementácia systému na odvodzovanie matematických hypotéz.* Diplomová práca, MFF UK Bratislava, 1997

[Gel 87] Gelfond, M. *On stratified autoepistemic theories.* Proceedings AAAI-87, 207-211

[GeL 90] Gelfond, M., Lifschitz, V. *Logic programs with classical negation.* Proc. of the 7th Int. Conf. on Logic Programming, ICLP'90, 579-597

[GeL 91] Gelfond, M., Lifschitz, V. *Classical negation in logic programs and deductive databases.* New Generation Computing, 365-387, 1991

[Gel 94] Gelfond, M. *Logic programming and reasoning with incomplete information.* Annals of Mathematics and Artificial Intelligence, 12 (1994), 89-116

[GeN 87] Genesereth, M.R., Nilsson, N.J. *Logical foundations of artificial intelligence.* Morgan Kaufmann Publishers 1987.

[Geo 96] Georgatos, K. *Ordering-based representations of rational inference.* Proc. of JELIA'96

[Geo 97] *Entrenchment relations: a uniform approach to nonmonotonicity.* Proc. of ESCQARU/FAPR'97

[GeP 92] Geffner, H., Pearl, J. *Conditional entailment: bridging two approaches to default reasoning.* Artificial Intelligence 53 (1992), 209-244

[GiM 90] Giordano, L., Martelli, A. *Generalized stable models, truth maintenance and conflict resolution.* Proc. 7th Int. Conf. on Logic Programming, MIT Press, 1990, 427-441

[Gin 86] Ginsberg, M. *Counterfactuals.* Artificial Intelligence, 30, 35-79,1986

[GKP 95] Gogic, G., Kautz, H., Papadimitriou, C., Selman, B. *The comparative linguistics of knowledge representation.* Proc. of IJCAI'95

[GLi 88] Gelfond,M., Lifschitz,V. *The stable model semantics for logic programming.* In: Proc. of the Fifth Int. Conf. and symposium on Logic Programming, eds. Kowalski,R.A., Bowen, K.A. MIT Press, 1988, 1070-1080.

[GMa 88] Gärdenfors, P., Makinson, D. *Revisions of knowledge systems using epistemic entrenchment.* Proc. of the 2nd Conf. on Theoretical

Aspects of Reasoning about Knowledge, Morgan Kaufamann, 1988, 83-95

[GMa 94] Gärdenfors, P., Makinson, D. *Nonmonotonic inference based on expectations.* Artificial Intelligence 65 (1994), 197-245

[Got 92] Gottlob, G. *Complexity results in non-monotonic logics.* J. of Logic and Computation, June 1992

[GPP 89] Gelfond, M., Przymusinska, H., Przymusinski, T. *On the relationship between circumscription and negation as failure.* Artificial Intelligence 38, 1989, 75-94

[Gra 96] Grass, J. *Reasoning about computational resource allocation. An introduction to anytime algorithms.* `http://www.acm.org/crossroads/xrds3-1-racra.html`

[GRo 95] Gärdenfors, P., Rott, H. *Belief Revision.* In: Handbook of Logic in Artificial Intelligence and Logic Programming, vol.4. Oxford 1995.

[Gro 97] Grosof, B. *Courteous Logic Programs: Prioritized Conflict Handling for Rules.* IBM Research Report RC 20836, 1997

[Gru 97] Gruska, J. *Foundations of Computing.* International Thomson Computer Press. 1997

[GSS 99] Gottlob, G., Scarcello, F., Sideri, M. *Fixed-Parameter Complexity in AI and Nonmonotonic Reasoning.* in [LPN 99]

[GuC 90] Guerreiro, R., Casanova, M. *An alternative semantics for default logic* Preprint, The Third International Workshop on Nonmonotonic Reasoning, South Lake Tahoe, 1990

[Guh 95] Guha, R.V. *Contexts: A Formaliztion and Some Applications.* Tech.Rep., 1995

[Haj 93] Hájek, P. *Epistemic entrenchment and arithmetical hierarchy.* Artificial Intelligence 62 (1993), 79-87

[Har 87] Harel, D.: *Algorithmics. The Spirit of Computing.* Addison-Wesley Publ. Comp., 1987

[HaV 91] Halpern, J., Vardi, M. *Model checking vs. theorem proving: a manifesto.* V.Lifschitz (ed.): Artificial Intelligence and Mathematical Theory of Computation (Papers in Honor of John McCarthy), Academic Press, 1991, 151-176

[Hay 79] Hayes, P.J. *The logic of frames.* In: Frame Conceptions and Text Understanding (ed. D.Metzing), deGruyter, 1979

[Hel 89] Helft, N. *Induction as nonmonotonic inference.* In: Proc. of the 1. International Conference on Principles of Knowledge Rerpresentation and Reasoning. Morgan Kaufmann1989, 149-156

[HmD 87] Hanks, S., McDermott, D. *Nonmonotonic logic and temporal projection.* Artificial Intelligence 33 (3), 1987, 379-412

[HMV 94] Halpern, J., Moses, Y., Vardi, M. *Algorithmic knowledge.* Proc. of the 5th Conf. on Theoretical Aspects of Reasoning about Knowledge (TARK 94), 255-266

[HoU 69] Hopcroft, J., Ullman, J. *Formálne jazyky a automaty.* Alfa 1978 (anglický originál 1969).

[HTo 95] Hinkle,D., Toomey, C. *Applying case-based reasoning to manufacturing.* AI magazine, Spring 1995, 65-73

[HTT 90] Horty,J.F., Thomason,R.H., Touretzky,D.S. *A Skeptical Theory of Inheritance in Nonmonotonic Semantic Networks.* Artificial Intelligence 41 (1990), 311 - 348

[ImL 84] Imieliński, T., Lipski, W. *Incomplete information in relational Databases.* Journal of the ACM, vol. 31, No. 4, October 1984, 761-791

[Imm 82] Immerman, N. *Relational queries computable in polynomial time.* 14th Ann. ACM Symp. on Theory of Computing. San Francisco, CA, May 1982, 147-152

[ImM 96] Imielinski, T., Mannila, H. *A database perspective on knowledge discovery.* Communications of the ACM, vol. 39, No. 11, Nov 1996, 58-64

[InS 96] Inoue, K., Sakama, C. *A fixpoint characterization of abductive logic programs.* Logic Programming, vol. 27 (1996), No. 2, 107-136

[KaK 97] Kanai, T., Kunifuji, S. *Extending inductive generalization with abduction.* IJCAI'97 workshop on abduction and induction

[KaM 90] Kakas, A., Mancarella, P. *On the relation between truth maintenace and abduction.* Proc. Conf. on Very Large Databases, 1990, 385-391

[KaM 90a] Kakas, A., Mancarella, P. *On the relation between truth maintenance and abduction.* Proc. of PRICAI'90, 438-443

[KaM 90b] Kakas, A., Mancarella, P. *Generalized stable models: a semantics for abduction.* Proc. of ECAI'90, 385-391

[Ker 99] Kern-Isberner, G. *Postulates for conditional belief revision.* Proc. of the IJCAI'99, 186-191

[KLo 89] Kifer, M., Lozinskii, E. *RI: A logic for reasoning with inconsistency.* Proc. of LICS 1989, IEEE Computer Society Press

[KKS 95] Kautz,H., Kearns, M., Selman, B. *Horn approximations of empirical data.* Artificial Intelligence 74 (1995), 129-145

[KKT 93] Kakas, A., Kowalski, R., Toni, F. *Abductive Logic Programming.* Journal of Logic and Computation, vol. 2 (1993), No. 6, 719-770

[KLM 90] Kraus, S., Lehmann, D., Magidor, M. *Nonmonotonic reasoning, preferential models and cumulative logics.* Artificial Intelligence 44 (1990), 167-207

[Knu 93] Knuth, D. *The Stanford GraphBase: a platform for combinatorial computing.* Addison-Wesley, 1993

[KoL 99] Koch, C., Leone, N. *Stable model checking made easy.* In [PAI 99], 70-75

[KSe 91] Kautz, H., Selman, B. *Hard problems for simple default logics.* Artificial Intelligence 49 (1991) 243-279

[KSf 90] Kalaš, I., Šefránek, J. *Reprezentačné schémy v znalostných systémoch.* Zborník SOFSEM'90, 227-252

[LeB 84] Lenat, D.B., Brown, J.S. *Why AM and EURISKO appear to work.*

Artificial Intelligence, vol. 23, No.3, 269-294

[LeB 85] Levesque, Brachman *A fundamental tradeoff in knowledge representation and reasoning.* In [BLe 85], 41-70

[LeM 92] Lehmann, D., Magidor, M. *What does a conditional knowledge base entail ?* Artificial Intelligence 55 (1992), 1-60

[Len 76] Lenat,D.B. *An artificial intelligence approach to to discovery in mathematics as heuristic search.* Memo AIM-286, Dept. of Computer Science, Stanford University, 1976

[Lev 84] Levesque, H.J. *Foundations of a Functional Approach to Knowledge Representation.* Artificial Intelligence 23 (1984), 155-212

[Lev 86] Levesque, H.J. *Making believers out of computers.* Artificial Intelligence 30 (1986), 81-108

[Lev 90] Levesque, H.J. *All I know: a study in autoepistemic logic.* Artificial Intelligence 42 (1990), 263-309

[Lif 88] Lifschitz, V. *On the declarative semantics of logic programs with negation.* In [Min 87], 177-192

[Lif 90] Lifschitz, V. *On open defaults.* Computational logic (ed. J.W.Lloyd), Springer 1990

[Lif 94] Lifschitz, V. *Minimal belief and negation as failure.* Artificial Intelligence 70 (1994), 53-72

[Lif 94a] Lifschitz, V., Turner, H. *Splitting a Logic Program.* Proc. of 11th Conf. on Logic. Programming, 1994, 23-37

[Lif 95] Lifschitz, V. *Circumscription.* Handbook of Logic in AI and Logic Programming, vol. 3

[Lif 99] Lifschitz, V. *Answer set planning.* Proc. of ICLP, 1999

[Lip 77] Lipski, W. *On the logic of incomplete information.* Mathematical Foundations of the Computer Science, 1977

[Lip 81] Lipski, W. *On databases with incomplete information.* Journal of the ACM, vol. 28, No. 1, January 1981, pp.41-70

[Llo 87] Lloyd, J.W. *Foundations of logic programming.* Springer-Verlag, 1987

[LoU 96] Lobo, J., Uzcátegui, C. *Abductive change operators.* Fundamenta Informaticae, vol. 27 (1996), No 4, 385-411

[LoU 97] Lobo, J., Uzcátegui, C. *Abductive consequence relation.* Artificial Intelligence, vol. 89 (1997), No 1-2, 149-171

[LPN 91] LPNMR'91. Proceedings of the 1th Int. Workshop Logic Programming and Non-Monotonic Reasoning, Washington, MIT Press, 1991

[LPN 93] LPNMR'93. Proceedings of the 2nd Int. Workshop Logic Programming and Non-Monotonic Reasoning, Lisbon, MIT Press, 1993

[LPN 95] LPNMR'95. Proceedings of the 3rd Int. Conference Logic Programming and Non-Monotonic Reasoning, Lexington, KY, Springer, 1995

[LPN 97] LPNMR'97. Proceedings of the 4th Int. Conference Logic Programming and Non-Monotonic Reasoning, Dagstuhl, Springer, 1997

[LPN 99] LPNMR'99. Proceedings of the 5th Int. Conference Logic Programming and Non-Monotonic Reasoning, El Paso, TX, Springer, 1999

[Luk 88] Lukasiewicz, W. *Considerations on default logic – an alternative approach.* Computational Intelligence, 4 (1988), 1-16

[Luk 90] Lukasiewicz, W. *Non-monotonic reasoning. Formalization of commonsense reasoning.* Ellis Horwood Ltd., 1990

[MaG 91] Makinson, D., Gärdefors, P. *Relations between the logic of theory change and nonmonotonic logic.* A.Fuhrmann, M.Morreau (eds.) The Logic of Theory Change, Springer, 1991

[Mak 89] Makinson, D. *General theory of cumulative inference.* In: Nonmonotonic reasoning. Reinfrank, de Kleer, Ginsberg, Sandewall (eds.), Springer, LNAI 465, 1989, 1-18

[Mal 65] Malcev, A. *Algoritmy i rekursivnyje funkcii.* Moskva, 1965

[Man 97] Mannila, H. *Methods and problems in data mining.* Proc. of Int. Conf. Of Database Theory, 1997

[Mar 97] Marek, W., Treur, J., Truszczynski, M. *Representation theory for default logic.* Annals of Mathematics and Artificial Intelligence 21 (2-4): 343-358, 1997

[Mar 00] Marek, W., prednáška, Viedeň, marec 2000

[MaT 93] Marek, W., Truszczynski, M. *Nonmonotonic logic: context-dependent reasoning.* Springer, 1993

[MaT 94] Marek, W., Truszczynski, M. *Revision specifications by means of programs.* In Proc. od JELIA'94, Springer 1994

[MaT 99] Marek, W., Truszczynski, M. *Stable models and an alternative logic programming paradigm.* In The Logic Programming Pardigm: a 25-Year Perspective, 375-398, Springer 1999

[McC 77] McCarthy, J. *Epistemological problems of Artificial Intelligence.* Proceedings of IJCAI77, Cambridge, Massachusets, 1977, pp.1038-1044

[McC 80] McCarthy, J. *Circumscription – a form of non-monotonic reasoning.* Artificial Intelligence 13 (1980), 27-39

[McD 80] McDermott, D., Doyle, J. *Non-monotonic logic I.* Artificial intelligence 13 (1, 2), (1980), 41-72

[McD 82] McDermott, D. *Non-monotonic logic II. Nonmonotonic modal theories.* JACM 29 (1) (1982), 33-57

[MCH 69] McCarthy, J., Hayes, P. *Some philosophical problems from the standpoint of Artificial Intelligence.* Machine Intelligence 4, eds. Meltzer, B., Michie, D., Edinburgh University Press, 1969, 463-502

[Min 82] Minker, J. *On indefinite databases and the closed world assumption.* In: Proc. of the 6th Conf. on Automated Dedudction. LNCS, 138, Springer-Verlag, 1982, 43-59

[Min 87] Foundations in Deductive Databases and Logic Programming. Ed. J. Minker. Morgan Kaufmann Publishers, 1987

[MiT 93] Mikitiuk, A., Truszczynski, M., Rational default logic and disjunctive logic programming. Logic Programming and Non-monotonic Reasoning, ed. L.M.Pereira, A. Nerode, MIT Press, 1993

[MiT 95] Mikitiuk, A., Truszczynski, M., Constrained and rational default logic. Proc. of IJCAI-95, Morgan-Kaufmann Publ., Inc.

[MNR 90] Marek, W., Nerode, A., Remmel, J. *A theory of nonmonotonic rule systems I, II*

[McC 86] McCarthy, J.,*Applications of circumscription to formalizing commonsense knowledge.* Artificial Intelligence, 28 (1986), 89-116

[Moi 88] Morris, P. *The anomalous extension problem in default reasonig.* Artificial Intelligence 35 (1988), 383-399

[Moo 85] Moore, R.C. *Semantic considerations on nonmonotonic logic.* Artificial Intelligence, 25 (1985), 75-94

[Mor 96a] Morgenstern, L. *Inheriting well-formed formulae in a formula-augmented semantic network.* In: Principles of Knowledge representation and Reasoning, 1996, 268-279

[Mor 96b] Morgenstern, L. *New problems for inheritance theories.* In: Third Symposium on formal theories of commonsense reasoning. 1996

[Mor 97] Morgenstern, L. *Inheritance comes of age: applying nonmonotonic techniques to problems in industry.* In: Proc. of IJCAI'97.

[Mor 99] Morgenstern, L. *Practical nonmonotonic reasoning: extending inheritance techniques to solve real-world problems.* In [LPN 99]

[MoS 97] Morgenstern, L., Singh, M. *An expert system using nonmonotonic techniques for benefits inquiry in the insurance industry.* In: Proc. of IJCAI'97.

[MuC 90] Muggleton, S., Cao Feng *Efficient induction of logic programs.* Proc. of the 1st Conf. on Algorithmic Learning Theory. Tokyo, 1990

[Mug 92] Muggleton, S. *Inductive logic programming.* Academic Press. 1992

[Mug 99a] Muggleton, S. *Inductive logic programming: issues, results and the challenge of learning language in logic.* Artificial Intelligence, 114 (1-2) (1999) 283-296

[Mug 99b] Muggleton, S. *Scientific knowledge discovery using inductive logic programming.* CACM 42 (11): 42-46, November 1999

[MuR 94] Muggleton, S., de Raedt, L. *Inductive logic programming: theory and methods.* Journal of logic programming 1994, 19, 20, 629-679

[NaA 89] Nait Abdallah, M.A. *An extended framework for default reasoning.* Proc. of the Int. Conf. Fundamentals of Comptation Theory, 339-348, 1989

[Nie 98] Niemelä, I. *Logic programs with stable model semantics as a constraint programming paradigm.* Workshop on computational aspects of nonmonotonic reasoning, Trento, 1998

[NSS 00] Nicolas, P., Saubion, F., Stéphan, I. *Genetic algorithms for extension search in default logic.* Workshop NMR'2000, Breckenridge

[PAI 99] *Proceedings of IJCAI'99.* Stockholm 1999

[Pap 93] Papadimitriou, C. *Computational complexity.* Addison-Wesley Publ. Comp. 1993

[PaS 92] Papadimitriou, C., Sideri, M. *On finding extensions of default theo-*

ries. Proc. Int. Conf. in Database Theory, Springer 1992, 276-281

[PaS 89] Patel-Schneider, P.F. *Undecidability of subsumption in NIKL.* Artificial Intelligence 39 (1989), 267-272

[PeU 98] Pérez, R., Uzcátegui, C. *Abduction vs. deduction in nonmonotonic reasoning.* Proc. of the Workshop on Formal Aspects and Applications of Nonmonotonic Reasoning, Trento, 1998

[PGA 87] Poole, D., Goebel, R., Aleliunas, R. *Theorist: a logical reasoning system for defaults and diagnosis.* In: [CMC 87].

[PiC 89] Pimentel, S., Cuadrado, J. *A truth maintenace system based on stable models.* Proc. of the North Am. Conf. on Logic Porgramming, vol. 1, 1989, 274-290

[Plo 70] Plotkin, G. *A note on inductive generalization.* In Machine intelligence 5, eds. B. Meltzer, D.Michie, Edinburgh, 1970, 153-163

[Plo 71] Plotkin, G. *A further note on inductive generalization.* In Machine intelligence 6, eds. B. Meltzer, D.Michie, Edinburgh, 1971, 101-124

[Poo 88] Poole, D. *A logical framework for default reasoning.* Artificial Intelligence 36 (1988), 27-47

[Pop 59] Popper, K. *The Logic of Scientific Discovery.* 1959

[Prz 87] Przymusinski, T. *On the declarative semantics of logic programs with negation.* In [Min 87], 193-216

[Prz 89a] Przymusinski, T. *On the declarative and procedural semantics of logic programs.* Journal of Automated Reasoning 5, 1989, 167-205

[Prz 89b] Przymusinski, T. *Non-monotonic formalisms and logic programming.* Proc. of 6th Int. Conf. on Logic Programming. MIT Press 1989, 655-674

[Prz 91] Przymusinski, T. *Three-valued nonmonotonic formalisms and semantics of logic programs.* Artificial Intelligence 49 (1991), 309-343

[Prz 97] Przymusinski, T. *Autoepistemic logic of knowledge and belief.* Artificial intelligence 95(1), 1997

[Prz 97b] Przymusinski, T., Turner, H. *Update by inference rules.* The Journal of Logic Programming, 1997

[ReC 81] Reiter, R., Criscuolo, G. *On interacting defaults.* Proc. of IJCAI'81, 270-276

[Rei 80] Reiter, R. *A logic for default reasoning.* Artificial Intelligence, 13 (1980), 81-132

[Rei 86] Reiter, R. *A sound and sometimes complete query evaluation algorithm for relational databases with null values.* Journal of the ACM, vol. 33, No. 2, April 1986, pp.349-370

[ReK 87] Reiter, R., deKleer, J. *Foundations of assumption-based Truth Maintenance Systems.* Proc. of the Conference of AAAI 1987, Morgan Kaufmann, 183-188

[RHa 84] Ritchie, G.D., Hanna, F.K. *AM: A case study in AI methodology.* Artificial Intelligence, vol. 23 (1984), No.3, 249-268

[Rog 67] Rogers, H. *Theory of recursive functions and effective computability.*

McGraw-Hill, 1967

[Ros 99] Rosatti, R. *Model checking for nonmonotonic logics: algorithms and complexity.* In [PAI 99], 76-81

[RoZ 95] Rounds, W., Zhang, G.Q. *Domain theory meets default logic.* Journal of Logic and Computation, vol. 5, No. 1, 1995, 1-25

[RoZ 97a] Rounds, W., Zhang,G.Q. *Nonmonotonic consequences in default model theory.* Annals of Mathematics and Artificial Intelligence, vol. 20, 1997, 227-265

[RoZ 97b] Rounds, W., Zhang,G.Q. *Defaults in domain theory.* Theoretical Computer Science, vol. 177, 1, 1997, 155-182

[RoZ 97c] Rounds, W., Zhang,G.Q. *Logical considerations on default semantics.* Annals of Mathematics and Artificial Intelligence, vo. 20, 1997, 195-226

[RoZ 97d] Rounds, W., Zhang,G.Q. *Complexity of power default reasoning.* Proc. of LICS'97, pp- 328-339

[RWe 91] Russell, S., Wefald, E. *Principles of metareasoning.* Artificial intelligence 49 (1991), 361-395

[SaI 91] Satoh, K., Iwayama, N. *Computing abduction by using the TMS.* Proc. of the Int. Conf. on Logic Programming, 1991, 505-518

[Sat 00] Satury, P. *Konštrukcia defaultových teórií z extenzií.* Diplomová práca, MFFUK Bratislava, 2000

[Sef 91] Šefránek, J. *Inconsistencies handling and non-monotonic reasoning.* Fundamentals of Artificial Intelligence Research, Springer, 1991

[Sef 97] Šefránek, J. *Dynamic Kripke structures.* CAEPIA'97. Actas de la VII Conferencia de la Associación Española para la Inteligencia Artificial, Malaga 1997, 271-283

[Sef 99] Šefránek, J. *Belief, knowledge, revisions, and a semantics of non-monotonic reasoning.* In [LPN 99]

[Sef 00] Šefránek, J. *A Kripkean semantics for dynamic logic programming.*

[Sha 83] Shapiro, E. *Algorithmic program debugging.* MIT Press 1983

[She 87] Shepherdson, J.C. *Negation in Logic Programming.* In [Min 87].

[Shh 88] Shoham, Y. *Reasoning about change: time and causation from the standpoint of artificial intelligence.* MIT Press, Cambridge, MA, 1988

[Sho 71] Shoenfield, J. *Degrees of unsolvability.* North-Holland Publ. Comp. 1971

[Sim 95] Simon, H.A. *Artificial intelligence: an empirical science.* Artificial Intelligence, vol. 77 (1995), 95-127

[SKa 90] Selman, B., Kautz, H. *Model-preference default theories.* Artificial Intelligence 45 (1990), 287-322

[SKa 96] Selman, B., Kautz, H. *Knowledge compilation and theory approximation.* Journal of the ACM, vol. 43, No. 2, March 1996, 193-224

[SLe 93] Selman, B., Levesque, H. *The complexity of path-based defeasible inheritance.* Artificial Intelligence 62 (1993), 303-339

[SMS 96] Srinivasan, A., Muggleton, S.H., Sternberg, M.J.E., King, R.D.

Theories for mutagenicity: a study in first-order and feature-based induction. Artificial Intelligence, 85 (1-2) (1996) pp. 277-299

[Sms 99] *Extending the stable model semantics with more expressive rules.* In [LPN 99], 305-316

[Smu 79] Smullyan, R.M.: *Logika prvého rádu.* Alfa, Bratislava, 1979.

[Ste 92] Stein, L.A. *Resolving ambiguity in nonmonotonic inheritance hierarchies.* Artificial Intelligence 55 (1992), 259-310

[StS 87] Sterling, L., Shapiro, E. *The art of Prolog.* MIT Press, 1987

[Tar 56] Tarski, A. *Logic, Semantics, Metamathematics.* Papers from 1923-1938. Oxford 1956

[Tou 86] Touretzky, D.S. *The mathematics of inheritance systems.* Pitman, London, 1986

[Ull 87] Ullman, J. *Database theory: Past and Future.* Principles of Database Systems, 1987

[Ull 88] Ullman, J. *Principles of database and knowledge-based systems.* Computer Science Press, 1988

[Val 95] Valdés-Pérez, R. *Machine discovery in chemistry: new results.* Artificial Intelligence 74 (1995), 191-201

[Val 96] Valdés-Pérez, R. *A new theorem in particle physics enabled by machine discovery.* Artificial Intelligence 82 (1996), 331-339

[Var 86] Vardi, M. *Querying logical databases.* Journal of Computer and System Sciences, vol. 33, No. 2, October 1986

[vBe 85] van Benthem, J. *The variety of consequence, according to Bolzano.* Studia logica 44, 1985, 389-403

[vBe 87] van Benthem, J. *Semantic parallels in natural language and computation.* Logic Colloquium'87, North Holland, 1989, 331-375

[vGR 91] van Gelder,A., Ross, K., Schlipf,J. *The well-founded semantics for general logic programs.* The Journal of ACM, 38 (3): 620-650,(1991)

[VHT 00] Verberne, A., van Harmelen, F., ten Teije, A. *Anytime diagnostic reasoning using approximate boolean constraint propagation.* Proc. KR 2000

[Wal 85] Walther, C. *A mechanical solution of Schubert's streamroller by many-sorted resolution.* Artificial Intelligence, vol. 26 (1985), 217-224

[Wec 92] Wechler, W. *Universal Algebra for Computer Scientists.* Springer, 1992

[WiA 98] Williams, M.-A., Antoniou, G. *A strategy for revising default theories extensions.* Proc. of the 6th Int. Conf. on Principles of Knowledge Representation. Morgan Kaufmann 1998

[WiB 93] Witteveen, C., Brewka, G. *Skeptical reason maintenance and belief revision.* Artificial Intelligence 61 (1993), 1-36

[Wil 97] Williams, M.-A. *Anytime belief revision.* Proc. IJCAI'97, 74-81

[Win 93] Winston, P. *Artificial Intelligence.* Addison-Wesley, Reading, Massachusets, 1993

[Woj 88] Wójcicki, R. *An axiomatic treatment of non-monotonic arguments.* Bulletin of the Section of Logic, Warszawa 1988, 56-61

[YuA 88] Yuasa, H., Arikawa, S. *Pseudo extensions in default reasoning and belief revision by model inference.* In: Logic Programming, Proc. of 7th Conf., Springer, LNCS, vol. 383, 27-37

[ZZC 98] Zhang, D., Zhu, Z., Chen, S. *Default reasoning and belief revision: a syntax-independent approach.* KR'98, Belief revison workshop

[Zil 95] Zilberstein, S. *Models of bounded rationality.* AAAI Fall Symposium on Rational Agency, November 1995

Ján Šefránek

Inteligencia ako výpočet

Vydal PhDr. Milan Štefanko – Vydavateľstvo IRIS

Tlač: Tlačiareň IRIS, Bodvianska 21, 821 07 Bratislava
Tel./fax: 07/4525 9552

ISBN: 80-88778-96-4