Multivalued Logics: A Uniform Approach to Inference in Artificial Intelligence

Matthew L. Ginsberg

Department of Computer Science
Stanford University
Stanford, California 94305
(415) 723-1239

Multivalued Logics: A Uniform Approach to Inference in Artificial Intelligence

Abstract

This paper describes a uniform formalization of much of the current work in AI on inference systems. We show that many of these systems, including first-order theorem provers, assumption-based truth maintenance systems (ATMS's) and unimplemented formal systems such as default logic or circumscription can be subsumed under a single general framework.

We begin by defining this framework, which is based on a mathematical structure known as a *bilattice*. We present a formal definition of inference using this structure, and show that this definition generalizes work involving ATMS's and some simple nonmonotonic logics.

Following the theoretical description, we describe a constructive approach to inference in this setting; the resulting generalization of both conventional inference and ATMS's is achieved without incurring any substantial computational overhead. We show that our approach can also be used to implement a default reasoner, and discuss a combination of default and ATMS methods that enables us to formally describe an "incremental" default reasoning system. This incremental system does not need to perform consistency checks before drawing tentative conclusions, but can instead adjust its beliefs when a default premise or conclusion is overturned in the face of convincing contradictory evidence. The system is therefore much more computationally viable than earlier approaches.

Finally, we discuss the implementation of our ideas. We begin by considering general issues that need to be addressed when implementing a multivalued approach such as that we are proposing, and then turn to specific examples showing the results of an existing implementation. This single implementation is used to solve a digital simulation task using first-order logic, a diagnostic task using ATMS's as suggested by de Kleer and Williams, a problem in default reasoning as in Reiter's default logic or McCarthy's circumscription, and to solve the same problem more efficiently by combining default methods with justification information. All of these applications use the same general-purpose bilattice theorem prover and differ only in the choice of bilattice being considered.

Keywords: knowledge representation, nonmonotonic reasoning, defaults, circumscription, commonsense reasoning, truth maintenance, theorem proving

Contents

1 Introduction

A substantial fraction of the artificial intelligence community is concerned with inference of one sort or another. Within this community, there is a variety of approaches taken to the long-term goal of constructing programs with interesting behavior. Members of one broad group, often referred to as "proceduralists," attack the problem directly, writing programs that succeed in performing some specific sort of inference task. Another "declarative" approach proceeds by attempting to find a precise formalization of the problems the proceduralists are considering.

Ted Shortliffe, creating MYCIN [50], is a typical proceduralist; John McCarthy, working on circumscription [34, 35], a typical declarativist. Drew McDermott, upon concluding that the Yale shooting problem [27] does not admit a straightforward formal solution, appears to have jumped from one camp to the other [37].

The unfortunate thing about this split is that the proponents of each approach have become increasingly uninterested in the results of the other. Formalists argue that proceduralists have no objective way in which to describe what it is that their programs are doing, and are therefore unimpressed by the successes of these programs. Proceduralists argue that the formal work is too far removed from problems of practical interest to be of any use. Instead of concern for or interest in the field of inference as a whole, the proceduralists consider principally the progress being made by their colleagues writing increasingly sophisticated programs; the declarativists concern themselves with the latest wrinkle in circumscription or related issues. I myself am hardly innocent in this regard.

This sort of methodological schism is a familiar scientific phenomenon. In *The Structure of Scientific Revolution* [30], T.S. Kuhn argues that resolving the difficulties between the two camps depends upon one approach solving not its own problems, but problems *of interest to the other approach*:

> Probably the single most prevalent claim advanced by the proponents of a new paradigm is that they can solve the problems that have led the old one to a crisis. When it can legitimately be made, this claim is often the most effective one possible. In the area for which it is advanced the paradigm is known to be in trouble. That trouble has repeatedly been explored, and attempts to remove it have again and again proved vain. [30, p. 153]

Kuhn is suggesting that a formal paper will be of interest to the proceduralists only if it enables them to write their programs more easily, to write more efficient or effective programs, or to write programs that could not previously be written.

My intention in this paper is to make just this sort of a contribution. What I hope to do is to formalize in precise but very general terms just what it is that the proceduralists' programs are doing.

In order to argue that such a formal description will lead to results of interest to the proceduralists, let me begin by describing *informally* what it is that these programs are doing.

Very informally, what they are doing is inference. PROLOG, for example, is a resolution theorem prover. MYCIN is a depth-first backward chainer.

MYCIN and PROLOG differ from first-order theorem provers in at least two significant ways. The first is that the inference procedures on which they are based are in some sense incomplete. Backward chaining is well known to be incomplete as a proof method, for example. The resolution strategy underlying most PROLOG implementations is incomplete as well, as is PROLOG's restriction to Horn clauses, with quantification handled implicitly and by Skolemization [4].

This is not a contentious issue. Both formalists *and* proceduralists have a clear interest in understanding the nature and computational properties of the various "approximations" to first order inference. The question of under what circumstances any particular proof strategy is or is not complete is of clear interest to both the procedural and formal camps, and cooperation here is fairly commonplace.

What *is* contentious is what a proceduralist program does with the results of its proving efforts. MYCIN, for example, labels each of the premises in any particular proof with a "certainty factor;" these certainty factors are combined in some *ad hoc* (but extremely effective!) fashion in order to assign a certainty factor to the proposition being proved.

PROLOG also uses the results of its first-order prover in unusual ways. The treatment of default rules via negation as "failure to prove" uses the results of a proof of p to trim the inference space involved in the proof of a second sentence q.

Both PROLOG and MYCIN are doing some sort of "bookkeeping" with the results returned by the first-order inference engine; it is through this sort of bookkeeping that the proceduralists' programs achieve much of their effectiveness.

In some cases, this is due to pruning of the search space. MYCIN, for example, has a heuristic that prunes any proof with an "overall" certainty factor of .2 or less. In other cases, the effectiveness is a result of a specialized combination rule. I have already remarked on the combination of certainty factors in MYCIN; other systems use other methods. Bayes' rule is used in the PROSPECTOR system [10], while more recent systems have investigated the possibility of using Dempster's rule, which combines probability ranges instead of specific values [8, 48].

Probabilistic systems are not the only ones that do bookkeeping of this sort. Truth maintenance systems [9] and assumption-based truth maintenance systems [5] collect the results of many proofs of a single sentence into a justification for that sentence. Default inference systems involving negation as failure draw conclusions by combining the results of attempts to prove a sentence with the results of an attempt to prove its negation.

My basic intention in this paper is to formalize this bookkeeping. The good news is that this formalization is of substantial value to both proceduralists and declarativists, as it leads to answers of interest to each; the bad news is that the formalism itself rests on mathematical results that are simple in concept but fairly difficult in practice. Let me discuss the bad news first.

The proceduralists' bookkeeping manipulates information that would be lost if the knowledge or data base were simply a list of valid sentences. The additional information can be represented in a variety of ways – certainty factors, rule order, pointers to justifications, and so on – but the information is always tangibly present. The approach I will be presenting works by giving this extra information a formal structure. The power of the approach lies in the fact that the *same* formal structure is shared by a wide variety of the procedural approaches.

The formal structure is based on the realization that the bookkeeping labels are capturing two sorts of information: that pertaining to how true or false a given statement is believed to be, and that relating to how much or little is known about it. The structure we are introducing – a *bilattice* – is little more than a set equipped with two partial orders, corresponding respectively to truth/falsity and to amount of evidence.

Throughout this paper, we will be describing the knowledge or beliefs held by an agent or inference system. Thus, we may say that one sentence p is "more true" than another sentence q; when we do so, we will mean only that the inference system has more reason to believe in the truth of p than it does to believe in the truth of q. In a purely Tarskian sense, of course, a sentence is either true or it isn't; the finer distinctions that will interest us can only be drawn when discussing an agent's beliefs, as opposed to the actual state of the world in which the agent finds itself.

We will return to these issues at much greater length shortly; at this point, let me note merely that these notions are not new ones. The idea of using a partial order to describe the truth or falsity of a sentence underlies the mathematical structure known as a *Boolean algebra* [26]; more recently, Sandewall [46] has suggested a partial order on probability intervals that corresponds to amount of underlying evidence. Similar ideas can also be found in [18, 40].

The approach I am proposing leads to substantial new insights for both the procedural and formal camps. On the formal side, a uniform treatment of existing procedural systems (default logics, truth maintenance systems, etc.) leads to a new description of default inference that, unlike its predecessors, has attractive computational properties. A uniform framework also ensures that it will be possible to obtain formal descriptions of prioritized default inference or other procedurally attractive systems.

As we will see, the procedural advantages are more substantial. The principal advantages resulting from the formal description are the following:

1. A formal description of inference allowing us to describe precisely what the behavior of an inference system should be and thereby test such a system for correctness,
2. Modularity of code, and
3. Efficiency of algorithm.

The first of these is probably of the least interest to the proceduralists (who verify their programs by using them, by and large); the impact of the remaining two points will be more significant.

The separation of the inference problem into "pure" first-order inference and "bookkeeping" leads to the usual benefits resulting from splitting a problem into smaller ones. Changing from one bookkeeping method to another is straightforward. Thus, if a knowledge engineer develops a knowledge base intending to solve a problem using default logic, but subsequently discovers that he must incorporate truth maintenance as well, he needs to modify only the "bookkeeping" portion of his inference engine, instead of replacing it in its entirety.

It is also possible to easily change from one first-order inference engine to another. The depth-first backward chainer used in many expert systems may be inadequate in some cases, perhaps because recursive inference paths require the use of breadth-first methods or depth-bounded ones, or perhaps because the completeness of a resolution theorem prover is needed. If the inference has been separated from the bookkeeping, modifying the expert system shell to change the underlying inference method is straightforward.

The final advantage of the separation is that it allows the development of general-purpose inference algorithms that do not depend upon the details of the bookkeeping rules. In principle, this may reduce the effectiveness of the developed system, since it will not be possible to use optimizations that depend on the bookkeeping procedures. As we will see, however, the uniform description of inference that we will present is the "right" one, in the sense that any such optimizations can apparently always be described *in terms of the mathematical structures we will discuss.* In actual fact, the inference algorithms developed without considering details of other problems appear to be *more* efficient than those developed otherwise, for the simple reason that the programming problem being solved is somewhat easier to understand in the general case.

As an example, the ideas in these papers have been implemented as part of the MRS expert system shell developed at Stanford [16, 45]. The overhead for the bookkeeping system generally was approximately 60%, in that inference tasks that did not need to make use of the multivalued facility ran 60% more slowly than they had originally.

The enhanced version of MRS was then used to develop an assumption-based truth maintenance system. The code took something under a day to develop (since only a few simple bookkeeping functions needed to be written). The resulting ATMS ran approximately three times faster than the justification procedure normally used with MRS, and the results computed by the two systems were equivalent. Additional details are in Section 16.

2 Overview

Once again, that's the good news. The bad news is that a lot of mathematics underlies these results. I will do my best to present examples of interest to an AI audience, but my main initial intention will be to lay the mathematical foundation needed later in the paper. This work makes up sections 3 through 7.

Following this, in sections 8 through 12 we discuss in theoretical terms the construction of an automated inference engine working in the setting developed here. The work in sections 3 through 7 presents a formal definition of inference using bilattices, but the description given is not computable. The work in the later sections develops effective procedures that are provably correct, in that they model the formal construction. The material in sections 13 through 18 is more practical still, discussing the implementation of these ideas within the MRS system, and describing in detail the use of the resulting system in implementing a variety of inference schemes.

There is an analogy here with work on first-order inference itself. Initially, we define entailment in a noncomputable way, saying that $p \models q$ if q holds in every model of p (and there are infinitely many such models). This sort of a description corresponds to the material in the first part of this paper.

The next step in a first-order approach, corresponding to the second section of this paper, is to consider a somewhat more constructive definition of entailment. One common method (see [11]) is to show that application of *modus ponens* to p together with all tautologies is sufficient to determine whether or not $p \models q$. If q follows from p using a construction of this sort, one generally writes $p \vdash q$; the construction is called *sound* if $p \vdash q$ implies that $p \models q$, so that any sentence proved from p does indeed follow from it; it is called *complete* if $p \models q$ implies that $p \vdash q$, so that all potential proofs can indeed be captured within the constructive framework. The procedures developed in the second paper are sound and complete in this sense.

Unfortunately, the "constructive" description of the last paragraph is generally not computationally effective. Classical first-order entailment is only semi-decidable, for example: Any algorithm that will eventually succeed in proving q from p if $p \models q$ may fail to terminate if q does not follow from p. So actually getting from a sound and complete description of inference to a useful program can be rather a large step. Different search procedures will have different properties. Perhaps completeness should be sacrificed for speed in some situations. Perhaps soundness should be sacrificed for speed, although this is less likely.

The final portion of the paper addresses issues of this sort. I discuss the current implementation of the ideas being presented, describe its use in a variety of settings, and analyze the efficiency of the algorithms being used. Finally, Section 19 summarizes the work and makes suggestions for future research.

In an (unsuccessful) attempt to keep the body of the paper to a manageable length, proofs of results can be found in Appendix A. Examples of LISP code describing the bilattices used in the various applications discussed in the final portion of the paper are contained in Appendix B.

3 Formalism: Introduction

Before beginning, let me present an outline of the first portion of the paper (sections 4 through 7) in somewhat greater depth. I begin in Section 4 by discussing issues in representation, as opposed to inference. Given that the bookkeeping of interest to us involves the manipulation of labels attached to the sentences in the database, what information should these labels contain? Section 4 also contains an example that has provided the fundamental motivation for a great deal of this work, and includes some necessary mathematical preliminaries. The basic representational scheme of interest to us, what I have chosen to call a *bilattice*, is introduced at this point.

An extended example of this construction is discussed in sections 4.5 and 5.2. Here we consider bilattices that can be generated simply by considering a collection of "possible worlds." We discuss the notion of inference using these bilattices, and use it to show that the approach we present does indeed generalize both first-order logic and truth maintenance systems that do not incorporate default rules.

Inference in a more general setting is discussed in Sections 5 through 7 and is compared to Reiter's default logic [41] in Section 7.3.

4 Representation

4.1 A motivating example

Although a principal reason for our introducing bookkeeping functions of the sort described in the introduction is to aid with the efficient implementation of a variety of inference schemes, there are representational advantages as well. As an example, consider Tweety, AI's well-known bird.

By default, birds fly; any of the standard formalizations of default reasoning (such as [35] or [41]) will allow us to conclude from this that Tweety can also fly. Suppose that we do so, adding this conclusion to our knowledge base. Only now do we learn that Tweety is in fact a penguin.

The difficulty is that this new fact is in contradiction with the information just added to our knowledge base. Having incorporated into our knowledge base the fact that Tweety can fly, we are unable to withdraw it gracefully.

The reason, loosely speaking, is that we have not recorded the fact that our belief in Tweety's flying is a consequence of *a default rule*, and is therefore subject to revision when new information is obtained.

Of course, truth maintenance systems [9] provide a way around this difficulty; the idea is to mark a statement not as merely "true" or "false," but as true or false *for a reason.* Thus our belief in Tweety's flying may depend on Tweety's being a penguin *not* being in the knowledge base; having recorded this, it is straightforward to adjust our knowledge base to record the consequences of the new information.

The truth maintenance approach, however, provides us with a great deal more power than is needed to solve this particular problem. We drew a default conclusion that was subsequently overturned by the arrival of new information. Surely we should be able to deal with this without recording the justification for the inference involved; it should be necessary merely to record the fact that the conclusion never achieved more than default status.

In this particular example, we would like to be able to label the conclusion that Tweety can fly not as true, but as true by default. The default value explicitly admits to the possibility of new information overturning the tentative conclusion it represents.

4.2 Lattices

The idea of labelling sentences as more than simply "true" or "false" is not a new one. There is an extensive literature discussing the ramifications of choosing the truth value assigned to a given statement from a continuum of possibilities instead of simply the two-point set $\{t, f\}$. Typical examples are a suggestion of Scott's in 1982 [47] and one of Sandewall's in 1985 [46].[1]

Scott notices that we can partially order statements by their truth or falsity, and looks at this as corresponding to an assignment to these statements of truth values chosen from some set L that is partially ordered by some relation $\leq_t$ (the reason for the subscript will be apparent shortly).

He goes on to note that if we can associate with the partial order $\leq_t$ a greatest lower bound operation glb and a least upper bound operation lub, the set L is what is known to mathematicians as a *lattice* [26]. Essentially, a lattice is a triple $\{L, \wedge, \vee\}$ where $\wedge$ and $\vee$ are binary operations from $L \times L$ to L that are *idempotent, commutative and associative*:

$$a \wedge a = a \vee a = a$$

$$a \wedge b = b \wedge a; \quad a \vee b = b \vee a$$

$$(a \wedge b) \wedge c = a \wedge (b \wedge c); \quad (a \vee b) \vee c = a \vee (b \vee c).$$

In terms of the partial order mentioned earlier, we have $a \wedge b = \mathrm{glb}(a, b)$ and $a \vee b = \mathrm{lub}(a, b)$. $\wedge$ is called the *meet* operation of the lattice; $\vee$ is called the *join.* Thus the first equality above states that the greatest lower bound of a with itself is a again; the other two equalities state that the greatest lower bound (meet) and least upper bound (join) operations are commutative and associative.

We also require that if $a \leq c$, then $a \wedge c = \mathrm{glb}(a, c) = a$ and $a \vee c = \mathrm{lub}(a, c) = c$. Since the elements c of the lattice with $a \leq c$ are all of the form $c = a \vee b$ for some b, this is captured by the *absorption identities*:

$$a \wedge (a \vee b) = a; \quad a \vee (a \wedge b) = a.$$

Lattices can be represented graphically. Given such a representation, we will take the view that $x \leq_t y$ if a path can be drawn on the graph from x to y that moves uniformly from left to right on the page.

In the lattice in Figure 1, f is the minimal element of the lattice, and t is the maximal element. We also have $a \leq_t b$; a and c are incomparable since there is no unidirectional path connecting them.

[1] Once again, we emphasize that we are really labelling not the sentence in question, but our degree of belief in that sentence. Although we will refer to the labels assigned to sentences as "truth values," they are to be distinguished from the truth values appearing in fuzzy logic [51]. There, a label of .7 for the sentence, "Fred is bald," means that Fred is fairly bald. We will take the label of .7 to mean that we have a certain level of confidence in the absolute truth of the sentence in question.

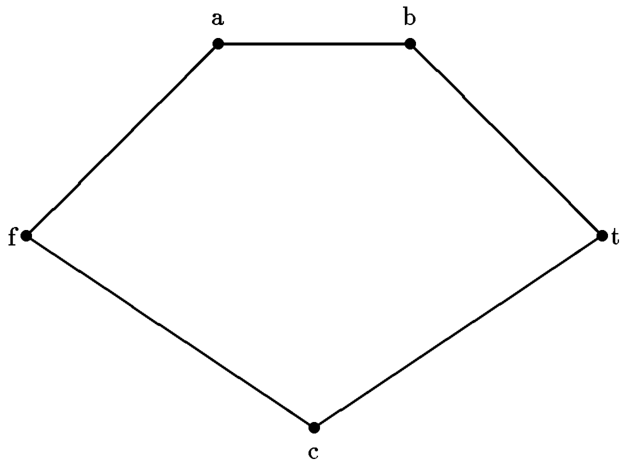

Figure 1: A lattice

Figure 2: The two-point lattice

A lattice is called *distributive* if $\vee$ and $\wedge$ distribute with respect to one another, so that

$$a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c), \tag{1}$$

for all a, b and c or, equivalently,

$$a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c).^2 \tag{2}$$

If a lattice has a maximal element 1 and a minimal element 0, it is called *bounded.* If the least upper bound and greatest lower bounds exist for arbitrary subsets of the lattice (as opposed to only finite subsets), the lattice is *complete.* Every complete lattice is bounded, since the 1 and 0 are given by the least upper bound and greatest lower bound of the entire lattice.

An element a of a bounded lattice has a *complement* b if $a \wedge b = 0$ and $a \vee b = 1$. A bounded lattice in which every element has a complement is called a *complemented* lattice. For a fuller description of these and many other lattice-related issues, the reader is referred to [26].

Up to isomorphism, there is a unique two-point lattice, shown in Figure 2. The truth values in first-order logic are chosen from this lattice; all we are saying here is that $f \leq_t t$; "true" is more true than "false."

4.3 Uncertainty

Sandewall's proposal, although also based on lattices, is different. Instead of ordering truth values based on truth or falsity, he orders them based on the completeness of the information they represent. Specifically, Sandewall suggests that the truth values be subsets of the unit interval $[0,1]$, the truth value indicating that the probability of the statement in question is known to lie somewhere in the associated probability interval. This proposal also appears in [19] and [40] and underlies Dempster-Shafer theory [8, 18, 48].

Given this interpretation of the truth values, if we learn more about some statement, this increase in our knowledge is reflected in a contraction of the interval of possible probabilities assigned to it. It follows that the lattice operation suggested by Sandewall is that of set inclusion. Thus *true*, corresponding to the singleton set $\{1\}$, is *incomparable* to *false*, which corresponds to the singleton $\{0\}$. (And each is in turn incomparable with any other point probability, such as $\{0.4\}$.) Instead, as remarked above, the inclusion of one truth value in another relates to our acquiring more information about the statement in question. The minimal element of the lattice is the full unit interval $[0,1]$; the fact that the probability of some statement lies in this interval contains no real information at all.

This is in sharp contrast with knowing, for example, that the probability of the statement in question is .5. If the probability of a coin's coming up heads is .5, the coin is fair; if nothing is known about the probability, it may well not be.

It is clear that the partial order corresponding to Sandewall's notion relates not to how confident we are that a particular statement p is true or false, but to how much information we have about p. To describe this distinct idea, we introduce a second partial order $\leq_k$ onto our lattice of truth values, interpreting $x \leq_k y$ to mean loosely that the evidence underlying an assignment of the truth value x is subsumed by the evidence underlying an assignment of the truth value y. Informally, more is known about a statement whose truth value is y than is known about one whose truth value is x.

If f and t in the two-point lattice corresponding to first-order logic are to be incomparable with respect to this second partial order, there is no way to introduce this second lattice structure onto the lattice in Figure 2. Instead, we need to introduce two additional truth values $\text{glb}_k(t,f)$ and $\text{lub}_k(t,f)$, as shown in Figure 3. Just as $x \leq_t y$ if x is to the left of y in a graphical representation, we will adopt the convention that $x \leq_k y$ if x is *below* y on the page.

The two new values are given by u (unknown) and $\perp$ (contradictory). The latter indicates a truth value subsuming both true *and* false; this truth value will be assigned to a given statement just in case it is possible to prove it true using one method and false using another.

We will denote the two lattice operations corresponding to $\leq_k$ by $+$ (lub_k) and $\cdot$ (glb_k) respectively.

[2] That either (1) or (2) is sufficient to guarantee the other is not obvious. See [26] for a proof.

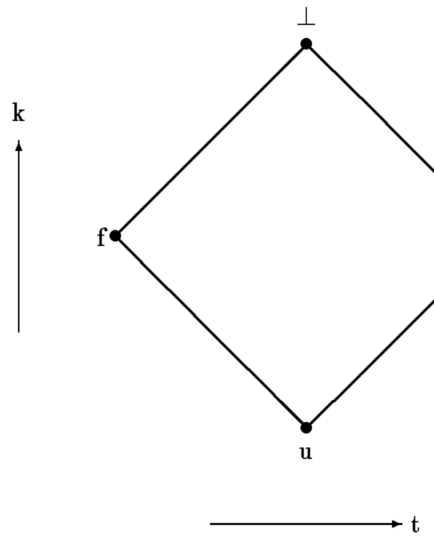

Figure 3: F, the smallest nontrivial bilattice

4.4 Negation

The two partial orders discussed thus far are related via the notion of *negation.* In the simple example of Figure 3, we would like to be able to say that the "negation" of t is f and vice versa, while the negations of u and $\perp$ are u and $\perp$ respectively (since if we know nothing about some statement, we also know nothing about its negation).

If B is the set of truth values underlying the two partial orders, we would like to think of negation as a map $\neg$ from B to B. If $x \geq_t y$, then we will expect $\neg x \leq_t \neg y$: Negation reverses the sense of the t partial order. Equivalently, if some sentence p is more true than a sentence q, we expect $\neg p$ to be *less* true (i.e., more false) than $\neg q$.

Somewhat less transparent is that we should have $\neg x \geq_k \neg y$ if and only if $x \geq_k y$: if we know less about p than about q, we also know less about the negation of p than about that of q. Finally, we require that $\neg\neg x = x$.

If a set B has two partial orders and a negation mapping satisfying the above conditions, we will call B a *bilattice.* Formally:

Definition 4.1 *A* bilattice *is a sextuple* $(B, \wedge, \vee, \cdot, +, \neg)$ *such that:*

1. $(B, \wedge, \vee)$ *and* $(B, \cdot, +)$ *are both complete lattices, and*
2. $\neg : B \rightarrow B$ *is a mapping with:*
   (a) $\neg^2 = 1$, *and*
   (b) $\neg$ *is a lattice homomorphism from* $(B, \wedge, \vee)$ *to* $(B, \vee, \wedge)$ *and from* $(B, \cdot, +)$ *to itself.*

The bilattice will be called k-distributive *if the* k *lattice* $(B, \cdot, +)$ *is distributive, and* t-distributive *if* $(B, \wedge, \vee)$ *is. It will be called* cross distributive *if each of* $\wedge$ *and* $\vee$ *distributes with respect to* $\cdot$ *or* $+$. *A bilattice that is* k*-distributive,* t*-distributive and cross distributive will simply be called* distributive.

Note that in condition (2b), we have reversed the order of $\wedge$ and $\vee$ while retaining the order of $\cdot$ and $+$. This corresponds to our earlier observations with regard to negation.

Note also that we are using the symbols $\wedge$ and $\vee$ to denote the greatest lower bound and least upper bound operations in the t lattice, even though these symbols are generally reserved for the conjunction and disjunction of logical sentences. The reason we have chosen to overuse notation in this fashion is that, as

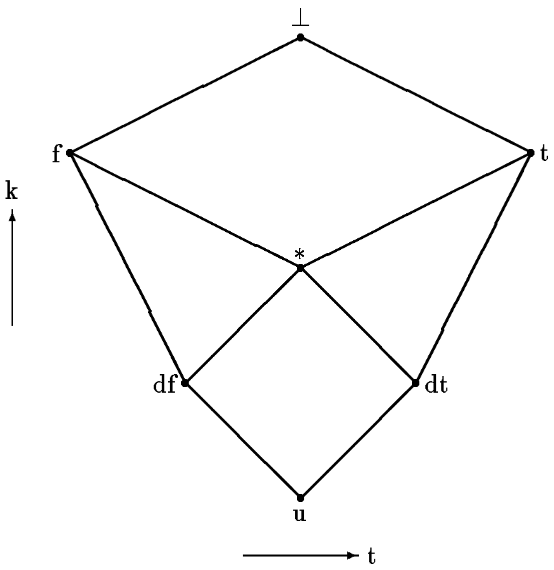

Figure 4: D, the bilattice for default logic

we will see in Section 6 and Definition 6.1, there is a close connection between the linguistic and bilattice operations.

The smallest bilattice is the trivial bilattice, in which B consists of a single point and the bilattice operations map that point to itself. In Section 7.3.3, we will briefly consider the possibility of performing inference using *infinite* bilattices.

The bilattice discussed at the end of the previous section is depicted in Figure 3; graphically, negation corresponds to reflection around the axis joining u and $\perp$. Another example appears in Figure 4; this is the bilattice of truth values used in Section 4.1's informal treatment of default logic. In addition to the old values of t, f, u and $\perp$, a sentence can also be labelled as dt (true by default) or df (false by default). The additional value $* = dt + df$ labels sentences that are both true *and* false by default. This is of course distinct from u (indicating that no information at all is available) or $\perp$ (indicating the presence of a proven contradiction). Once again, negation corresponds to reflection around the u-$\perp$ axis. We will discuss this bilattice in greater detail in Section 7.3.2.

Before proceeding, however, note that this bilattice shares the elements t, f, $\perp$ and u with the previous one. In fact, *any* bilattice will have four distinguished elements, corresponding to the maximal and minimal elements under the two partial orders. We will denote these distinguished elements in this fashion throughout the paper.

We immediately have:

Lemma 4.2 *In any bilattice,* $\neg t = f$ *and* $\neg f = t$.

We also have the following result, which tells us that neither t nor f subsumes the information corresponding to the other:

Lemma 4.3 *In any nontrivial bilattice,* t *and* f *are* k*-incomparable.*

In other words, neither $t \leq_k f$ nor $f \leq_k t$.

It follows from this that the bilattice shown in Figure 3 is indeed the smallest nontrivial bilattice. Belnap [2, 3] has considered the possibility of selecting truth values from this bilattice.

4.5 World-based bilattices

A fairly wide class of bilattices can be obtained by considering a collection of *worlds*. Informally, by a world we mean some possible way things might be.

Suppose that W is some set of worlds, and let (U, V) be a pair of subsets of W. Then if we say that some sentence p has truth value (U, V), we will mean that p is true in the worlds in U, and false in those in V. We do not require $U \cap V$ to be empty, although any world w in $U \cap V$ will be inconsistent, since p will be both true and false in w. We also do not require $U \cup V = W$; the truth or falsity of p will simply be undetermined in any world in $W - (U \cup V)$.

Given these truth values, what does it mean for a sentence p with truth value (U, V) to be less true than a sentence q with truth value (S, T)? Clearly, p will be less true than q just in case it is true no more frequently than q is, and false *at least as frequently.* This gives us:

$$(U, V) \leq_t (S, T) \quad \text{iff} \quad U \subseteq S \text{ and } T \subseteq V.$$

In an analogous way, we obtain

$$(U, V) \leq_k (S, T) \quad \text{iff} \quad U \subseteq S \text{ and } V \subseteq T,$$

which says that we know more about q than about p if we know q to be true whenever p is, and false whenever p is.

It is fairly clear that the four bilattice operations associated with $\leq_t$ and $\leq_k$ are:[3]

$$\begin{aligned}
(U, V) \wedge (S, T) &= (U \cap S, V \cup T) \\
(U, V) \vee (S, T) &= (U \cup S, V \cap T) \\
(U, V) \cdot (S, T) &= (U \cap S, V \cap T) \\
(U, V) + (S, T) &= (U \cup S, V \cup T).
\end{aligned}$$

The four preferred elements of the bilattice are given by $u = (\emptyset, \emptyset)$, $t = (W, \emptyset)$, $f = (\emptyset, W)$ and $\perp = (W, W)$.

Negation on this bilattice is given by $\neg(U, V) = (V, U)$. If some sentence p is known to be true in the worlds in U and false in those in V, then $\neg p$ is known to be true in the worlds in V and false in those in U.

We denote the set of pairs of subsets of W by B_W, so that $B_W = \mathcal{P}(W) \times \mathcal{P}(W)$, where $\mathcal{P}(W)$ is the power set of W. We now have:

Lemma 4.4 *For any set W, $(B_W, \wedge, \vee, \cdot, +, \neg)$ is a bilattice.*

Let me end this section with two comments about this construction. Firstly, note that if there is only a single world in the set W being considered, then B_W collapses to F, the bilattice corresponding to first-order logic. This is exactly as it should be: Logic deals simply with "the way things are." No alternatives are considered. But many alternatives to logic are based on considerations of "ways the world might be," and the world-based bilattices will enable us to understand these ideas in a formal way.

But we should also note that as we have presented these ideas, the worlds being considered are all indistinguishable. Most formalizations of nonmonotonic reasoning within AI are based on some sort of partial order among these worlds themselves (this is made explicit in [12, 22, 25, 49] and is implicit in [35, 41]). We will address this possibility in Section 7.

5 Inference

In order to apply the representational ideas we have been discussing, it is insufficient merely to give a framework in which to describe the truth values associated with the sentences of our language. We must also be able to perform inference using these truth values. We now turn to the issue of describing logical operations in a multivalued setting.

[3] This particular bilattice appears to have been discovered independently by Delgrande [7].

5.1 Framework

Let L be the set of all well-formed formulae in our language. We will define a *truth assignment* to be any mapping that assigns some truth value in the bilattice B to each formula in L:

Definition 5.1 *A* truth assignment *is a function* $\phi : L \to B$.

In first-order logic, inference is described in terms of a closure operation which, given a set of sentences, produces the complete set of consequences of those sentences. This is merely a formal description, providing a theoretical background against which to measure a specific implementation.

We propose to take a similar approach, capturing the notion of inference by defining a closure operation on truth assignments. Given a truth assignment ϕ and a sentence p, ϕ contains explicit information about p in the truth value $\phi(p)$, together with a variety of implicit information in the form of truth values on sentences logically related to p. Both of these will ideally be captured via the closure operation, which will produce a new truth assignment $\mathrm{cl}(\phi)$ that captures all of the explicit *and implicit* information about p in the new value $\mathrm{cl}(\phi)(p)$.

We can expect the closure operation to have the following properties:

1. It should be a bilattice construction, depending only on the partial orders and negation defining the bilattice being investigated.
2. It should accurately reflect the notions of inference already formalized in existing work on extensions to first-order logic.
3. $\mathrm{cl}(\mathrm{cl}(\phi)) = \mathrm{cl}(\phi)$ for all ϕ.
4. $\mathrm{cl}(\phi)(p) \geq_k \phi(p)$ for all p.

The first of these properties reflects our belief that the information already available in the bilattice structure should be sufficient to enable us to define a suitable notion of inference. Recall that we expect to be able to describe a wide variety of types of inference in terms of conventional first-order inference, combined with some "bookkeeping" functions. First-order inference is a well-defined notion, and the bookkeeping functions are simply those describing the bilattice. It follows that the notion of inference we are formalizing should incorporate information derived from the bilattice structure, but should not involve any other appeals to potentially domain-dependent information or techniques.

We also will expect the multivalued closure operation to give results that can be easily interpreted in terms of the existing work that has been done on formal descriptions of logical or extra-logical inference. We will see examples of this in Sections 6.3 and 7.3, for example.

The remaining two conditions are technical, stating respectively that the closure operation need only be performed once in order to extract all of the information implicit in the truth values given by ϕ, and that if ϕ takes the value x at some sentence p, performing inference on ϕ should never retract this explicit information.

In order to deal with these issues formally, we make the following definition:

Definition 5.2 *A truth assignment ϕ will be called an* extension *of a truth assignment ψ if $\phi(p) \geq_k \psi(p)$ for all $p \in L$. If ϕ is an extension of ψ, we will write $\phi \geq_k \psi$. If the inequality is strict for at least one $p \in L$, we will write $\phi >_k \psi$ and say that the extension is* proper.

Informally, an extension of a truth assignment is what is obtained upon the acquisition of more information about some sentence or sentences in L. The extension will be proper if and only if the new information was not explicitly stated in the original truth values.

5.2 Closure for world-based bilattices

It would be possible at this point for me to simply present a definition of closure for multivalued truth assignments, but let me spend some time developing it in order that it may be a little better motivated. We will start by defining closure for truth assignments whose image is some world-based bilattice B_W.

In this case, it should be possible to use the existing notion of logical closure to determine the "closure" of any particular truth assignment ϕ. For recall that ϕ, given a sentence p, associates with p those worlds in which it is known to be true or false.

An alternative way to view ϕ is by assuming that, given some world w, it associates with w a set of sentences ϕ_w known to be true (with $\neg p$ known to be true if p is known to be false). We can now take the logical closure of this set of sentences; this new set of sentences $\mathrm{cl}(\phi_w)$ should be the set of sentences assigned to w by the "closure" of ϕ, $\mathrm{cl}_W(\phi)$ (the subscript indicates that this definition is drawn from an intuition about world-based bilattices and applies only to them).

Formalizing this will be somewhat simpler if we begin by restricting our attention to the first component of ϕ. Recall that a truth assignment for the bilattice B_W is a mapping

$$\phi : L \to \mathcal{P}(W) \times \mathcal{P}(W).$$

We can project this mapping onto the first component to get another truth assignment

$$\phi_+ : L \to \mathcal{P}(W).$$

ϕ_+ takes a sentence p and maps it to the collection of worlds where it is known to hold. (We will discuss the information in the second component of ϕ presently.)

As a function from L to $\mathcal{P}(W)$, ϕ_+ is nothing more than a relation on L and W, with $\phi_+(p, w)$ if and only if p is given as true in w. As suggested above, we can easily interpret this as a function $\tilde{\phi}_+ : W \to \mathcal{P}(L)$ which, given a world w, produces the set of sentences known to be true in it.

Rather than define the closure of a truth assignment directly, we will first define what it means for a truth assignment to be *closed*, so that $\phi = \mathrm{cl}_W(\phi)$. It follows from the remarks of the previous paragraph that one requirement we should make is that $\tilde{\phi}_+(w)$ be closed (in the usual logical sense) for each $w \in W$.

This has left unaddressed the information contained in the second component of ϕ. To handle this, we simply require $\phi(\neg p) = \neg\phi(p)$ for all $p \in L$. Thus if p is given as true in the worlds in U and false in those in V, so that $\phi(p) = (U, V)$, we must have $\phi(\neg p) = (V, U)$ if ϕ is to be closed. This gives us the following:

Definition 5.3 *A truth assignment $\phi : L \to B_W$ will be called* W-closed *if and only if:*

1. *$\tilde{\phi}_+(w)$ is deductively closed (as a subset of L) for each $w \in W$, and*
2. *$\phi(\neg p) = \neg\phi(p)$ for each $p \in L$.*

The approach we are following is quite close to that used to define the closure operation in first-order logic itself. There, an initial definition is made of what it is for a set to be closed (that $p \wedge q$ be in the set if p and q are, for example), and the closure of some fixed set S of sentences is defined to be the intersection of all of the closed supersets of S. This definition makes sense because the intersection of closed sets is itself closed, and in fact we have:

Lemma 5.4 *Let ϕ and ψ be two W-closed truth assignments on a bilattice B_W. Then $\phi \cdot \psi$ is also W-closed. In general, if $\{\phi_i\}$ are all W-closed, then so is*

$$\prod_i \phi_i = \phi_1 \cdot \ldots \cdot \phi_n.$$

It follows from the lemma that we can mimic the usual definition of closure:

Definition 5.5 *The* W-closure *of a truth assignment* ϕ *defined on a world-based bilattice is given by*

$$\mathrm{cl}_W(\phi) = \prod\{\psi|\psi \geq_k \phi \text{ and } \psi \text{ is } W\text{-closed}\}. \tag{3}$$

Corollary 5.6 *If* ϕ *is a truth assignment defined on a world-based bilattice, then* $\mathrm{cl}_W(\phi)$ *is* W*-closed.*

6 Closure on balanced bilattices

6.1 The bilattice construction

One of the conditions on a truth assignment ϕ being W-closed in the last section was that $\tilde{\phi}_+$ be closed. This notion, however, is not bilattice-theoretic: It required us to introduce the mapping $\tilde{\phi}_+$, which in turn depended upon the precise structure of the bilattice $B_W = \mathcal{P}(W) \times \mathcal{P}(W)$. Our aim in this section is to give an equivalent definition of closure that does not suffer from this deficiency.

Definition 6.1 *A truth assignment* ϕ *will be called* apparently closed *if:*

1. *If* $p \models q$*, then* $\phi(q) \geq_t \phi(p)$*,*
2. $\phi(\wedge_i p_i) \geq_k \wedge_i \phi(p_i)$*, and*
3. $\phi(\neg p) = \neg\phi(p)$ *for all* p*.*

Unlike the previous definition, this one *is* dependent only upon the partial orders $\geq_t$ and $\geq_k$ and the negation defining the bilattice. The motivation for the various clauses in the definition is as follows:

1. If $p \models q$, then we would expect q to be at least as true as p is; in other words, we would expect $\phi(q) \geq_t \phi(p)$.
2. We should know *at least as much* about a conjunction as we do about each of the conjuncts. In some cases, however, we may know more.

   Consider a situation where $\phi(p) = \phi(\neg p) = u$. Now we have $\phi(p) \wedge \phi(\neg p) = u$, but we should still have $\phi(p \wedge \neg p) = f$. Because of the special logical connection between p and $\neg p$, we know more about the conjunction $p \wedge \neg p$ that we can conclude from the truth values assigned to the conjuncts p and $\neg p$ individually. In this case, since $t \models \neg(p \wedge \neg p)$, we can conclude from the first clause in the definition that $\phi[\neg(p \wedge \neg p)] \geq_t t$ for any p, so that $\phi[\neg(p \wedge \neg p)] = t$, and $\phi(p \wedge \neg p) = f$ because of the third clause.
3. For the third clause in the definition, we have already discussed negation, arguing that negation should map the truth value assigned to p to that assigned to $\neg p$.

We also have the following two results:

Lemma 6.2 *A truth assignment is apparently closed if and only if it is* W*-closed.*

Theorem 6.3 $\mathrm{cl}_W(\phi) = \prod\{\psi|\psi \geq_k \phi$ *and* ψ *is apparently closed*$\}$.

This construction, as opposed to that in (3), depends only upon the bilattice in question.

In addition to the above results, we will also have occasion to consider the third clause of Definition 6.1 in isolation. To this end, we make the following definitions:

Definition 6.4 *We will call two sentences* p *and* q negation equivalent*, writing* $p \equiv_\neg q$*, if there exists a nonnegative integer* n *such that* $p = \neg^{2n} q$ *or vice versa.*

Thus p and $\neg\neg p$ are negation equivalent, and so on. We also define a truth assignment to be $\neg$-closed if and only if it satisfies the final condition of Definition 6.1:

Definition 6.5 *A truth assignment ϕ will be called* ¬-closed *if $\phi(\neg p) = \neg\phi(p)$ for all p.*

Lemma 6.6 *A truth assignment ϕ is ¬-closed if and only if*

$$\phi(p) = \sum\{\phi(q) | q \equiv_\neg p\} + \sum\{\neg\phi(q) | q \equiv_\neg \neg p\}$$

for all p.

6.2 The independence result

The results of the previous section gives us a formally satisfactory way in which to calculate closure for world-based bilattices. An attractive feature of the multivalued approach, however, is that it does *not* necessarily require us to keep track of all of the individual worlds that underlie the truth values. Consider probabilities, for example: The truth values here contain very streamlined information about the relative frequency with which some sentence holds among the various possible worlds, as opposed to any sort of enumeration of them.

This is a fairly general situation:

Definition 6.7 *A bilattice B will be called* balanced *if there exists a collection of worlds W and a surjective bilattice homomorphism h from B_W to B. We will say that h and W* balance *B in this case.*

The intuition behind the definition of balance is that the bilattice treats all of the world in W uniformly, not preferring some of them over others. The default bilattice of Figure 4, for example, treats unconditional information preferentially over default information, and this bilattice is not balanced. (This is a consequence of Theorem 9.21 of Section 9.2.)

Note also that as a consequence of the definition, we must have $h(B_W) = B$ and $h(\neg x) = \neg h(x)$ for all $x \in B_W$. We also have:

Lemma 6.8 *Any balanced bilattice is distributive.*

Now let ϕ be a truth assignment on some balanced bilattice B, where B is balanced by h and W, and suppose that χ is a truth assignment on B_W with $h(\chi) = \phi$. It would be attractive if it were possible to take the closure of χ on B_W, and to define the closure of ϕ to be the image of this closure under h. The potential difficulty with this is that we have made a variety of choices (in W, h and χ) in defining the closure of ϕ; the closure construction should be independent of such choices.

In fact, the construction we have presented *is* independent of the choices of h, W and χ made in the previous paragraph. For reasons that will be apparent in Section 7, we will only call this the *balanced closure* of ϕ, denoting it $\mathrm{cl}_B(\phi)$. We have:

Theorem 6.9 *Let (h_1, W_1) and (h_2, W_2) both balance a bilattice B. Then if ϕ is a truth assignment on B, and χ_i are truth assignments on B_{W_i} for $i = 1, 2$, with $h_i(\chi_i) = \phi$,*

$$h_1[\mathrm{cl}(\chi_1)] = h_2[\mathrm{cl}(\chi_2)]$$

(and both are equal to $\mathrm{cl}_B(\phi)$). In fact,

$$\mathrm{cl}_B(\phi) = \prod\{\psi | \psi \geq_k \phi \text{ and } \psi \text{ is apparently closed}\}. \tag{4}$$

As a consequence, we can show that the balanced closure operation satisfies the technical conditions suggested in Section 5.1:

Corollary 6.10 *For any truth assignment ϕ, $\mathrm{cl}_B(\phi) \geq_k \phi$ and $\mathrm{cl}_B(\mathrm{cl}_B(\phi)) = \mathrm{cl}_B(\phi)$.*

We can also show that the balanced closure of a truth assignment on a balanced bilattice is itself apparently closed. (This result does not hold in general; see Section 7.2.)

Corollary 6.11 *If ϕ is a truth assignment defined on a balanced bilattice, $\mathrm{cl}_B(\phi)$ is apparently closed.*

Finally, we have the following:

Corollary 6.12 *cl_B is k-monotonic. In other words, if $\phi \leq_k \psi$, then $\mathrm{cl}_B(\phi) \leq_k \mathrm{cl}_B(\psi)$.*

What this corollary tells us is that if ϕ is "more informed" than ψ is, then $\mathrm{cl}(\phi)$ will necessarily be more informed than $\mathrm{cl}(\psi)$. In other words, the sort of nonmonotonicity where we must retract some conclusion in the face of new evidence cannot arise if the notion of inference being used is that of balanced closure for some bilattice.

6.3 Examples

The definition of balanced closure implicit in (4) can, of course, be applied to bilattices that are not themselves balanced. We will see in Section 7 that there is a more useful definition for unbalanced bilattices, but the existing definition is sufficiently powerful to capture both the standard notion of inference and that used in assumption-based truth maintenance systems, or ATMS's [5, 44].

Investigating this requires that we "translate," in some sense, the construction used by (for example) a reason maintenance system into our framework. This translation is essentially a three step process, requiring that we:

1. Describe a bilattice that captures the extra information considered by the inference method, and indicate the fashion in which information available before any inference is attempted is encoded in this bilattice. An ATMS, for example, associates a set of "contexts" with each sentence in the language, indicating what assumptions need to be made (if any) in order for the sentence to be valid. This set of contexts will need to be encoded in the truth value of the sentence in question.
2. Investigate the properties of the closure operation for this bilattice.
3. Demonstrate that the bilattice closure operation corresponds naturally to the inference mechanism or description being modelled. See, for example, the following Theorems 6.13 and 6.15.

6.3.1 First-order logic

We have already remarked that first-order logic corresponds closely to the four-point bilattice F, which we repeat in Figure 5.

To see that this bilattice can indeed be used to capture the usual notion of inference, suppose that we have some set S of sentences, and are interested in the consequences of these sentences. We first define a truth assignment ϕ_S by:

$$\phi_S(q) = \begin{cases} t, & \text{if } q \in S; \\ u, & \text{otherwise.} \end{cases}$$

Essentially, ϕ_S is the truth assignment that "knows" the facts in S, but nothing else.

The equivalence result is now given by the following:

Theorem 6.13 *Let S be a set of sentences, and q a sentence. Then:*

$$\mathrm{cl}_B \phi_S(q) = \begin{cases} \bot, & \textit{if } S \textit{ is inconsistent;} \\ t, & \textit{if } S \textit{ is consistent and } S \models q; \\ f, & \textit{if } S \textit{ is consistent and } S \models \neg q; \\ u, & \textit{otherwise.} \end{cases} \tag{5}$$

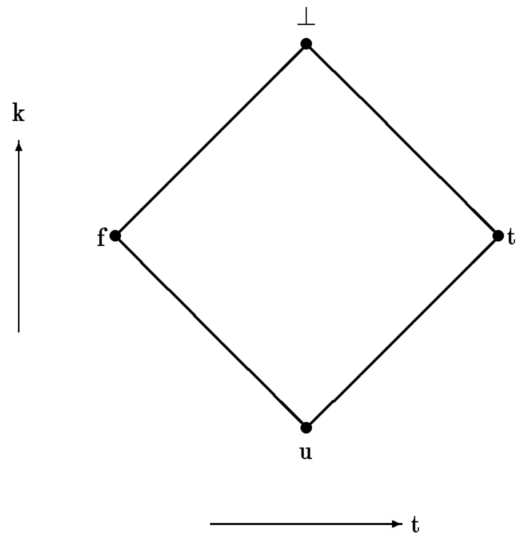

Figure 5: F, the bilattice for first-order logic

Alternatively, we can view the correspondence between first-order inference and the bilattice F in terms of the following corollary to the above result:

Corollary 6.14 *Let S and T be two sets of sentences. Then the following two conditions are equivalent:*

1. $T \subseteq \mathrm{cl}(S)$
2. $\phi_T \leq_k \mathrm{cl}_B(\phi_S)$.

6.3.2 ATMS's

In an ATMS, the truth value assigned to some sentence contains information concerning potential reasons for its truth or falsity. We can capture this in a multivalued setting by using a bilattice in which the truth values consist of pairs $[\ a\ .\ b\ ]$ where a and b are respectively justifications for the truth or falsity of the statement in question. In order to make this notion precise, we need to formalize the justifications themselves.

Justifications In deKleer's original ATMS [5], the justification associated with any particular sentence was a list of contexts in which the sentence was valid; each context consisted of a list of sentences that characterized that context.

An example will make this clearer. Suppose that p is some statement, and that p will hold if q does, or if r and s both hold. Then there are two known contexts for p, one given by $\{q\}$ and the other by the pair $\{r, s\}$. The context given by $\{r, s\}$ corresponds to the set of situations in which the conjunction $r \wedge s$ is valid.

We see from this that we can view a justification as a collection of subsets of our language L, so that the set of all justifications is simply $\mathcal{P}(\mathcal{P}(L))$ (the power set of the set of subsets of L). We will tend to write justifications as *lists* of sets, in order to keep the notation manageable. Thus the justification for p in the previous paragraph would be given by

$$(\{q\}\{r, s\}).$$

Given these notational conventions, it is not hard to see that one can partially order the set of justifications, with the "maximal" justification (i.e., that requiring the fewest premises) being $(\{\})$: A sentence with this justification will hold in any context in which the sentences in $\{\}$ hold. (In other words, the sentence will hold in any context at all.) The "minimal" justification of the empty set $()$ gives us no information whatsoever about contexts in which some sentence holds.

Formally, the partial order on justifications is given as follows: Suppose that j_1 and j_2 are two justifications. Then we will take $j_1 \leq j_2$ to mean that every conjunctive subclause in j_1 contains some subclause in j_2 as a subset:

$$(a_1 \ldots a_n) \leq (b_1 \ldots b_m)$$

if for each a_i, there is some b_j with $b_j \subseteq a_i$. It is not hard to see that the empty justification () is minimal under this partial order, while ({}) is maximal; this is in keeping with the remarks of the previous paragraph.[4]

The ATMS bilattice We saw in Section 4.5 that given a set W of worlds, it was possible to construct a bilattice from the partial order on the subsets of W. In exactly the same way, we can construct a bilattice from the lattice of justifications just discussed. The elements of the bilattice are pairs $[\, j_1 \,.\, j_2 \,]$ of justifications. Just as the first component of a truth value in a world-based bilattice listed those worlds in which some sentence was known to hold, the justification j_1 is a justification for the truth of the sentence in question. The justification j_2 gives those contexts in which a particular sentence is known to be false, and is therefore a justification for the *negation* of the sentence being considered.

The partial orders on the ATMS bilattice are given by:

$$[\, j_1 \,.\, k_1 \,] \leq_t [\, j_2 \,.\, k_2 \,] \qquad \text{iff} \qquad j_1 \leq j_2 \text{ and } k_1 \geq k_2, \tag{6}$$

and

$$[\, j_1 \,.\, k_1 \,] \leq_k [\, j_2 \,.\, k_2 \,] \qquad \text{iff} \qquad j_1 \leq j_2 \text{ and } k_1 \leq k_2. \tag{7}$$

The analog to Theorem 6.13 is now the following:

Theorem 6.15 *Let S be the set of assumptions in our knowledge base, and suppose that ϕ is given by*

$$\phi(q) = \begin{cases} [\ (\{q\}) \,.\, \text{nil}\ ], & \textit{if } q \in S; \\ u, & \textit{otherwise.} \end{cases}$$

Then if $\{q_1, \ldots, q_m\} \subseteq S$ and p is an arbitrary sentence,

$$[\ (\{q_1, \ldots, q_m\}) \,.\, \text{nil}\ ] \leq_k \text{cl}_B \phi(p)$$

if and only if the q_i's entail p. In addition,

$$[\ \text{nil} \,.\, (\{q_1, \ldots, q_m\})\ ] \leq_k \text{cl}_B \phi(p)$$

if and only if the q_i's entail $\neg p$.

Intuitively, the theorem begins by labelling each sentence in S as "self-justified;" for any q, q holds in any context in which q holds. The conclusion tells us that if some collection of these q's justifies p, in the sense that $q_1, \ldots, q_m \models p$, then the truth value of p in the closure will reflect this in that

$$[\ (\{q_1, \ldots, q_m\}) \,.\, \text{nil}\ ] \leq_k \text{cl}_B \phi(p).$$

The justification for p in the closure requires no more premises than a justification depending simply on the q_i's.[5]

[4] We are being somewhat sloppy here. The partial order just defined is in fact only a *preorder*, since we can have $x \leq y$ and $y \leq x$ without having $x = y$. As an example, suppose that x is the justification $(\{p\})$, and that y is the justification $(\{p\}, \{p, q\})$. The second element of y is irrelevant, since the associated context is subsumed by the first one. In actual fact, the justification lattice consists of the equivalence classes under this preorder on $\mathcal{P}(\mathcal{P}(L))$. This is the usual way in which partial orders are constructed from preorders.

[5] This result is clearly related to Reiter and de Kleer's description of an ATMS in terms of what they refer to as *prime implicants* [44].

Nogoods In many cases, a single sentence p will be true in some contexts but false in others. It is clear from Theorem 6.15 that this is handled easily with the ATMS bilattice, and simply corresponds to a justification $[\ j_1 \ . \ j_2\ ]$ where neither j_1 nor j_2 is empty.

More interesting is the fact that as we have described the ATMS bilattice, it is possible for a single justification to appear in *both* halves of some statement's truth value, so that we may have

$$[\ j \ . \ j\ ] \leq_k \phi(p)$$

for some sentence p and justification j.

In this case, j justifies not only p but also its negation. It follows that j is inconsistent with the known facts in the database; deKleer refers to such an assumption set as a *nogood.* He goes on to indicate that it is possible to "simplify" the justifications labelling the various sentences so as to eliminate nogoods, since any justification involving a nogood is effectively useless. It is upon this idea that the practical value of the ATMS appears to rest.

As an example, suppose that $a \wedge b$ is inconsistent with the facts in the database, so that $\{a, b\}$ is a nogood. DeKleer would now have us rewrite the justification

$$(\{a, b\}\{c\}) \tag{8}$$

as simply $(\{c\})$, since the first justification depends on a nogood. Note that we would also rewrite

$$(\{a, b, d\}\{c\}) \tag{9}$$

as $(\{c\})$, as the first justification continues to be a superset of the nogood $\{a, b\}$.

It is remarkably easy to capture this notion within the bilattice framework. We need the following:

Theorem 6.16 *Let B be a distributive bilattice. Then if x is any element of B with $x = \neg x$, the subset of B given by*

$$B^x = \{y \in B | y \geq_k x\}$$

is a bilattice. In addition, any extension of a truth assignment with values in B^x will also take values in B^x.

The theorem tells us that the set of points "above" x in a distributive bilattice also constitutes a bilattice. In addition, since this new bilattice is closed under extension, any truth assignment taking values in it will continue to take values in it when the closure is taken.

Now suppose that j is a nogood, so that the conjunction of the facts in each term in the justification is provably false. In this case, *any* sentence p will follow from the justification, as will $\neg p$. It follows that

$$\mathrm{cl}_B \phi(p) \geq_k [\ j \ . \ j\ ]$$

for any sentence p. We can therefore apply the above theorem to conclude that we can restrict all of our subsequent analysis to the bilattice $B^{[\ j \ . \ j\ ]}$. DeKleer's observation is simply that, as in (8) and (9), it is possible to simplify the *labelling* of the points in $B^{[\ j \ . \ j\ ]}$ by removing j from all of the justification lists. This relabelling, too, is a bilattice operation:

Theorem 6.17 *If B is a distributive bilattice such that the k-lattice of B is complemented, there is a natural bilattice homomorphism $i^x : B^x \to B$ such that $i^x(x) = u$ and i^x maps B^x isomorphically onto its image.*

In other words, the "relabelling" given in the theorem corresponds to deKleer's: It is an isomorphism that takes a justification corresponding simply to the nogood j into "unknown." That the mapping be an isomorphism is important because it guarantees that information about justifications other than j is preserved.

7 Closure on arbitrary bilattices

7.1 Motivation

It is well known that in any particular situation involving nonmonotonic reasoning, there may be conflicts among the default rules that can be applied. Consider Reiter's well-known political example [43]: Republicans are, by default, nonpacifists. Quakers are, by default, pacifists. Nixon is a Republican and a Quaker. Is he a pacifist?

In Reiter's description, the above default theory admits two extensions. In one, the Republican default rule is valid, and Nixon is a hawk. In the other, the Quaker rule is valid and Nixon is a dove. Reiter's construction gives us no way to determine which of the two extensions actually should be believed.

Circumscription, as described in [35], is similar. If we modify our description to say that people are hawks, doves, or apathetic, then the circumscriptive approach will enable us to conclude that Nixon is either a hawk or a dove, but will give us no indication at all as to which.

Hanks and McDermott [27] have demonstrated that some solution to this problem is a necessary precursor to an adequate formal description of actions and their consequences.[6] They note that actions only succeed by default (this observation is what McCarthy [33] has dubbed the *qualification* problem), and that it is possible for the consequences of one action to invalidate another. If this happens, which action do we expect to fail?

The particular example given by Hanks and McDermott involves loading a gun, waiting, and then firing it at someone (generally named Fred). According to the frame axiom, the gun's being loaded should persist through the waiting action, *and* Fred's being alive should persist through the shooting action. Given that these default conclusions conflict with one another, how are we to conclude that it is the former that is in fact valid?

One possible solution to this problem is to *prioritize* the default rules in question; Shoham, for example, proposes in [49] a prioritization based on a temporal order. Lifschitz describes a similar prioritization using pointwise circumscription in [31]. Etherington and Reiter [13] also deal with the issue of prioritized default rules, although restricting their attention to problems involving conflicting defaults on inheritance hierarchies.

Informally, these solutions are capturing the idea that we may prefer worlds in which one default rule fails over those in which another fails. In inheritance hierarchies, for example, we prefer the rule that birds fly to the rule that animals don't.

Even more simply, imagine that we have a rule indicating that all flying things are unafraid of heights. Given that we only know *by default* that Tweety can fly, we now have a choice: We can either abandon this default conclusion, or abandon our uncertainty about Tweety's being acrophobic. We choose the latter, and conclude (by default) that Tweety is indeed unafraid of heights.

7.2 Bounded extensions

It is possible to understand this choice in terms of the underlying bilattices. Let us suppose that we are working in a system using the default bilattice D, which we repeat in Figure 6.

Let us denote by *bird* the statement the Tweety is a bird, by *flies* the statement that he can fly, and by *acro* the statement that he is acrophobic. Our initial truth assignment ϕ is given by:

$$\begin{aligned}
\phi(\mathit{bird}) &= t, \\
\phi(\mathit{flies}) &= u, \\
\phi(\mathit{acro}) &= u, \\
\phi(\mathit{bird} \rightarrow \mathit{flies}) &= dt, \\
\phi(\mathit{flies} \rightarrow \neg \mathit{acro}) &= t.
\end{aligned}$$

[6] Recent investigations by Lifschitz [32] and Ginsberg and Smith [23, 25] indicate that Hanks and McDermott may be overly pessimistic here. But that is another tale.

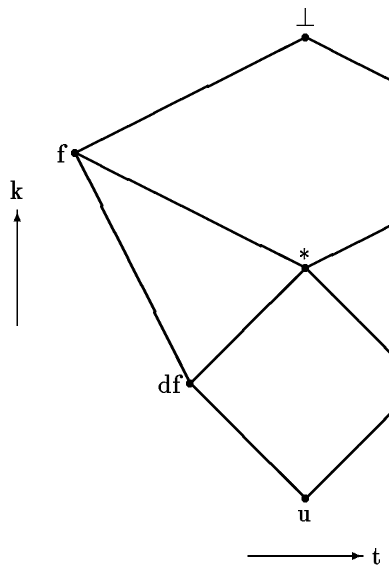

Figure 6: D, the bilattice for a simple default logic

Tweety is a bird, birds fly by default, and flying things are not afraid of heights. Nothing is known at this point about Tweety's flying abilities or mental condition.

In order to compute the closure of the above truth assignment, we need to evaluate the product appearing in (4). In fact, it suffices to take the product of only the *k-minimal* apparently closed extensions of ϕ, since any apparently closed extension of ϕ that is not k-minimal will make no contribution when the product is evaluated.

In any apparently closed extension of the truth assignment we are considering, the application of the default rule $bird \rightarrow flies$ will allow us to conclude that the truth value of *flies* is $\geq_k dt$. At this point we have a choice. If we take $\psi(flies) = dt$, then the rule saying that flying things are not acrophobic can be applied to conclude $\psi(\neg acro) = dt$ as well.

On the other hand, if we do not take $\psi(flies) = dt$, but instead consider the possibility that Tweety in actuality *cannot* fly, we can avoid concluding anything about the sentence *acro*.

We see from this that the given truth assignment has two k-minimal apparently closed extensions. In the first, we simply apply the two rules of inference, getting:

$$\begin{aligned}
\psi_1(bird) &= t, \\
\psi_1(flies) &= dt, \\
\psi_1(acro) &= df, \\
\psi_1(bird \rightarrow flies) &= dt, \\
\psi_1(flies \rightarrow \neg acro) &= t.
\end{aligned}$$

In the second, we abandon the default conclusion that Tweety can fly:

$$\begin{aligned}
\psi_2(bird) &= t, \\
\psi_2(flies) &= f, \\
\psi_2(acro) &= u, \\
\psi_2(bird \rightarrow flies) &= f, \\
\psi_2(flies \rightarrow \neg acro) &= t.
\end{aligned}$$

It follows from this that the balanced closure of ϕ is given by:

$$\begin{aligned}
\mathrm{cl}_B\phi(\mathit{bird}) &= t,\\
\mathrm{cl}_B\phi(\mathit{flies}) &= dt,\\
\mathrm{cl}_B\phi(\mathit{acro}) &= u,\\
\mathrm{cl}_B\phi(\mathit{bird} \rightarrow \mathit{flies}) &= dt,\\
\mathrm{cl}_B\phi(\mathit{flies} \rightarrow \neg \mathit{acro}) &= t.
\end{aligned}$$

In the balanced closure, all of the possible worlds are treated uniformly, and we are unable to decide whether to use the default conclusion or abandon it. Note that $\mathrm{cl}_B\phi$ is not apparently closed in this case.

The difference between the two extensions ψ_1 and ψ_2 is that ψ_1 adjusts the truth value of $\neg acro$ from u to dt, drawing a default conclusion, while ψ_2 adjusts the truth value of *flies* from dt to f, drawing a "certain" one. Since the certainty of this conclusion is not justified by the available evidence, we should prefer ψ_1 to ψ_2.

What does it mean in general for one truth assignment to draw stronger conclusions than another? The comparison drawn between ψ_1 and ψ_2 above was not a pointwise one; the pointwise definition of an extension appearing in Definition 5.2 is unable to distinguish between ψ_1 and ψ_2.

The distinction we are trying to draw is a *global* one. The reason that ψ_2 draws stronger conclusions than ψ_1 does is because the conclusion that Tweety *certainly* cannot fly is stronger than *any* conclusion drawn by ψ_1. By "any conclusion" here, we mean the truth values taken by ψ_1 at points where they differ from truth values taken by ψ_2. In other words, ψ_2 is drawing stronger conclusions than ψ_1 because there is some sentence p (that Tweety can fly, in this case) such that for *every* sentence q with $\psi_1(q) \neq \psi_2(q)$, we have:

$$\psi_2(p) >_k \psi_1(q).$$

To formalize this, we make the following definition:

Definition 7.1 *Let ϕ and ψ be two truth assignments. We will say that ψ is* bounded by *ϕ, or that ϕ* bounds *ψ, writing $\psi \lhd \phi$, if either $\psi <_k \phi$ or there is some x with $\phi(x) \neq \psi(x)$ and*

$$\sum\{\psi(z) | \psi(z) \neq \phi(z)\} <_k \phi(x). \tag{10}$$

Once again, the intent of the definition is to capture the sense in which the two extensions in our example differ. The definition formalizes that way in which ψ_2 (which drew an unjustifiable conclusion that *flies* was false) is "greater" than ψ_1 (which drew the intended default conclusion).

In our example, the sentences in question are *flies*, *acro* and *bird* $\rightarrow$ *flies*. Since ψ_2 takes values f, u and f at these sentences, while ψ_1 takes values whose least upper bound is $\mathrm{lub}_k\{dt, df, dt\} = *$, we conclude that ψ_2 has drawn stronger conclusions than ψ_1 has, and thus that $\psi_1 \lhd \psi_2$.

Lemma 7.2 *The relation $\lhd$ induces a partial order on the set of truth assignments from L to B.*[7]

We now define the *closure* of a truth assignment ϕ, defined on an arbitrary bilattice B, to be the greatest lower bound of all of its apparently closed extensions *that are $\lhd$-minimal.* In other words, if ψ_1 and ψ_2 are two apparently closed extensions of ϕ with $\psi_1 \lhd \psi_2$ (as in our example), only ψ_1 is considered when constructing the closure:

Definition 7.3 *Let ϕ be an arbitrary truth assignment, and denote by $A(\phi)$ the set of ϕ's apparently closed extensions. Then the* closure *of ϕ, to be denoted* $\mathrm{cl}(\phi)$*, is given by:*

$$\mathrm{cl}(\phi) = \prod\{\psi \in A(\phi) | \chi \in A(\phi) \Rightarrow \chi \not\lhd \psi\}. \tag{11}$$

[7] Although $\lhd$ corresponds to a partial order on the set of truth assignments, it does not lead to a lattice structure for this set. The reason is that the greatest lower bound and least upper bound operations with respect to $\lhd$ need not be unique.

Lemma 7.4 *For any ϕ, $\mathrm{cl}(\phi) \geq_k \mathrm{cl}_B(\phi)$.*

Once again, we can show that the closure operation satisfies the technical conditions suggested in Section 5.1:

Theorem 7.5 *For any truth assignment ϕ, $\mathrm{cl}(\phi) \geq_k \phi$ and $\mathrm{cl}(\mathrm{cl}(\phi)) = \mathrm{cl}(\phi)$.*

7.3 Examples

7.3.1 Balanced bilattices

We have already seen several examples of balanced closure for world-based bilattices in Section 6.3. Those examples remain valid in the general case because of the following result:

Theorem 7.6 *Closure and balanced closure are identical for balanced bilattices.*

Clearly world-based bilattices are balanced.

Corollary 7.7 *The closure operator is k-monotonic on balanced bilattices.*

7.3.2 Default reasoning

Reiter's construction Reiter defines default reasoning in terms of a *default theory* (R, T) where T is a collection of first-order sentences, and R is a collection of *defaults*, each of the form

$$\frac{\alpha : \beta_1, \ldots, \beta_m}{w},$$

indicating roughly that if α holds and all of the β's are consistent, then w holds. If every default rule is of the form

$$\frac{\alpha : w}{w},$$

so that we infer w from α in the absence of information to the contrary, the default theory is called *normal.*

Reiter goes on to define an operator Γ such that for any collection S of sentences, $\Gamma(S)$ is the smallest set of sentences such that:

1. $T \subseteq \Gamma(S)$,
2. $\Gamma(S)$ is deductively closed, and
3. If $(\alpha : \beta_1, \ldots \beta_m / w) \in R$, $\alpha \in \Gamma(S)$ and $\neg\beta_1, \ldots, \neg\beta_m \notin S$, then $w \in \Gamma(S)$.

Reiter now defines an *extension* of (R, T) to be any set E such that $E = \Gamma(E)$, and shows that the extensions of a default theory correspond to the collections of facts derivable from such a theory. Since there may be conflicting default rules, it is possible that the extension E may not be unique.

The connection between default logic and a multivalued approach is clearest if all of the default rules are of the form

$$\frac{: w}{w}.$$

We will call a default theory of this form *supernormal.*

The intuition here is that w holds in the absence of information to the contrary; this is similar to the approach taken in circumscription, where the extent of some "abnormality" predicate is minimized. Thus, to express the default rule, "If Tweety is a bird, then Tweety can fly," in a supernormal default theory, we would write:

$$b \wedge \neg ab \rightarrow f$$

$$\frac{: \neg ab}{ab},$$

as opposed to the more usual[8]

$$\frac{b:f}{f}.$$

Defaults and multivalued logic The bilattice for default logic was shown in Figure 6.

Theorem 7.8 *Let (R,T) be a supernormal default theory where T is consistent. Associate with (R,T) a truth assignment ϕ given by*

$$\phi(p) = \begin{cases} t, & \textit{if } p \in T; \\ dt, & \textit{if } p \textit{ is } r_i \textit{ for some } \frac{:r_i}{r_i} \in R \textit{ but } p \notin T; \\ u, & \textit{otherwise.} \end{cases}$$

Then for any i,

$$\mathrm{cl}(\phi)(r_i) = \begin{cases} t, & \textit{iff } T \models r_i; \\ f, & \textit{iff } T \models \neg r_i; \\ *, & \textit{iff } r_i \textit{ is true in some extensions of } (R,T) \textit{ and false in others;} \\ dt, & \textit{iff } T \not\models r_i \textit{ but } r_i \textit{ is true in all extensions of } (R,T). \end{cases} \tag{12}$$

Note that Theorem 7.8 only discusses the truth values of the default facts themselves, and not of their consequences. As an example, suppose that we have two conflicting default rules stating, for example, that since Nixon is a Quaker, he is a dove (p_1), but since he is a Republican, he is a hawk (p_2). Now there is a fact in the database stating also that hawks are not doves: $p_2 \rightarrow \neg p_1$.

Given this, we will be led to conclude that the truth value assigned by $\mathrm{cl}(\phi)$ to both p_1 and p_2 is $*$. The interesting thing is that the truth value assigned to the disjunction $p_1 \vee p_2$ is *also* $*$, and not dt as one would expect (since the disjunction holds in all extensions of the default theory).

Using the simple bilattice D, there is no way to "keep track" of which sentences are true in which extensions. It is true that in this example there are only two extensions, with p_1 true in one and p_2 true in the other, but this information is not recorded anywhere in the truth assignment ϕ.

It is possible to correct this by including reason maintenance information in the default bilattice. In the proof of Theorem 7.8 appearing in Appendix A, we note that D is in fact the simple four-point bilattice F, with the point u replaced with the collection of truth values dt, df, $*$ and u (corresponding to a "default" copy of the original bilattice F). Instead of replacing u with a copy of F, it is possible to replace it with a copy of the ATMS bilattice presented in Section 6.3.2. The result is a bilattice that handles interactions between conflicting default rules in a more complete fashion. This bilattice is discussed at some length (although not in these terms) in [24], where it is shown that it can be used to construct a circumscriptive theorem prover. The general procedure of constructing a bilattice by splicing u out of some bilattice and replacing it with another bilattice is described in [21].

7.3.3 Prioritized defaults

In this section, we give a simple illustration of a bilattice that can be used to handle prioritized defaults. The bilattice in question is shown in Figure 7. We have taken the default bilattice D and expanded the point u to reflect the existence of a "second order default."

[8] The two theories being presented here are in fact different. If we know $\neg f$, so that Tweety doesn't fly, then the "abnormality" formalism will conclude that Tweety is not a bird, while the more conventional one will not. Thus the default rule does not contrapose in the original formulation. I am indebted to Karen Myers for pointing this out to me. Imielinski also discusses this and related issues at some length in [28]; his results appear to indicate that it is only seminormal defaults without preconditions that can be captured in a multivalued setting.

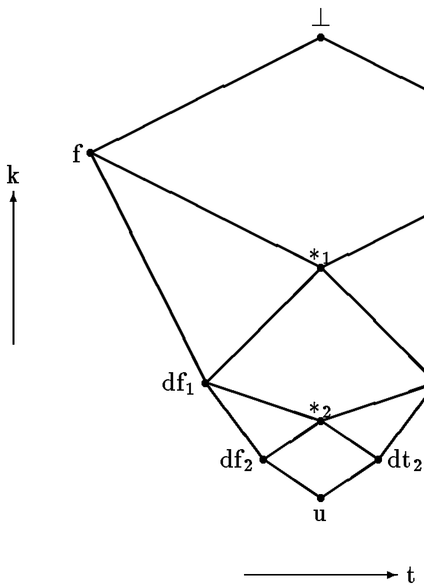

Figure 7: The bilattice for a prioritized default logic

Theorem 7.9 *Suppose we fix some set of sentences T, together with two default sentences p_1 and p_2, and that ϕ is given by:*

$$\phi(p) = \begin{cases} t, & \text{if } p \in T; \\ dt_1, & \text{if } p = p_1; \\ dt_2, & \text{if } p = p_2; \\ u, & \text{otherwise.} \end{cases}$$

Then for any sentence p,

$$\mathrm{cl}(\phi)(p) = \begin{cases} t, & \text{iff } T \models p; \\ f, & \text{iff } T \models \neg p; \\ dt_1 & \text{iff } p \text{ is independent of } T \text{ but } T \cup \{p_1\} \models p; \\ df_1 & \text{iff } p \text{ is independent of } T \text{ but } T \cup \{p_1\} \models \neg p; \\ dt_2 & \text{iff } p \text{ is independent of } T \cup \{p_1\} \text{ but } T \cup \{p_1, p_2\} \models p; \\ df_2 & \text{iff } p \text{ is independent of } T \cup \{p_1\} \text{ but } T \cup \{p_1, p_2\} \models \neg p; \\ u, & \text{otherwise.} \end{cases}$$

Extensions of this idea can obviously be used to handle hierarchies of defaults, such as those discussed by Etherington and Reiter in [13] or by Shoham in [49]. Note in particular that there is no reason to assume that the "levels" of default information are discrete. Thus we might construct a bilattice where the default levels corresponded to a time chosen from a continuous range.

7.4 Nonmonotonicity

We have shown in Corollary 7.7 that if the bilattice in question is either world-based or the image of a world-based bilattice, inference is k-monotonic. In this case, we will see in Section 10.2 that it is possible to "incrementally" calculate the closure, determining what changes can be directly attributed to a new piece of incoming information.

This need not be true in general. The reason is that, although $A(\phi) \subseteq A(\psi)$ if $\phi \geq_k \psi$, the $\lhd$-minimal elements of $A(\phi)$ may not all be $\lhd$-minimal elements of $A(\psi)$, so that it is possible that $\text{cl}(\phi) \not\geq_k \text{cl}(\psi)$ even if $\phi \geq_k \psi$.

We have already seen an example of this in Section 7.2. Suppose that we extend our original example there, adding the fact that penguins do not fly. Now consider the following two truth assignments:

p	$\phi(p)$	$\psi(p)$
bird(Tweety)	t	t
penguin(Tweety)	u	t
flies(Tweety)	u	u
acrophobic(Tweety)	u	u

Clearly $\phi <_k \psi$. But the closures of ϕ and ψ are given by:

p	$\text{cl}(\phi)(p)$	$\text{cl}(\psi)(p)$
bird(Tweety)	t	t
penguin(Tweety)	df	t
flies(Tweety)	dt	f
acrophobic(Tweety)	df	u

We do not have $\text{cl}(\phi) \leq_k \text{cl}(\psi)$. The point is that the fact that Tweety is now known not to fly blocks the rule about acrophobia.

This will make the calculational requirements of an inference scheme using the bilattice D tremendously greater than those needed in a k-monotonic situation. The problem is that we must essentially rederive all of our earlier conclusions at every step; indeed, this *is* a necessity in a default inference system. (Perhaps this explains why there are so few of them.)

We will show in Theorem 10.1 that it is possible to understand this k-monotonicity (or the lack thereof) simply in terms of the underlying bilattices. For a wide class of bilattices which we will refer to as *stratified*, for example, inference is k-monotonic if and only if $t \cdot f = u$ (see Theorem 11.9).

7.5 Properties of inference

We conclude this section by demonstrating that certain typical properties of inference are preserved in the multivalued approach, independent of the bilattice being considered. For example, the truth values assigned to two logically equivalent sentences should always be identical after the closure is taken.

In order to describe this and similar properties, we define a truth assignment ϕ to be *closed* if $\phi = \text{cl}(\phi)$, and have:

Lemma 7.10 *If p and q are logically equivalent, $\phi(p) = \phi(q)$ for any closed truth assignment ϕ.*

Next we show that if q is true, then the truth value of a conjunction $p \wedge q$ is the same as the truth value of p for closed truth assignments.

Theorem 7.11 *Let ϕ be a closed truth assignment, and suppose that $\phi(q) = t$ and $\phi(r) = f$. Then*

$$\begin{aligned} \phi(p \wedge q) &= \phi(p) \\ \phi(p \vee r) &= \phi(p) \\ \phi(p \rightarrow r) &= \neg\phi(p) \\ \phi(q \rightarrow p) &= \phi(p) \end{aligned}$$

for all p.

A variety of familiar results follow from this:

Corollary 7.12 (Modus Ponens) *Let ϕ be a closed truth assignment. Then for any p and q,*

$$\phi(q) \geq_t \phi[p \wedge (p \rightarrow q)].$$

In addition, if $\phi(p \rightarrow q) = t$, then $\phi(q) \geq_t \phi(p)$.

Lemma 7.13 *If ϕ is a closed truth assignment, $\phi(p \wedge q) \geq_k \phi(p) \wedge \phi(q)$. In addition, $\phi(p \vee q) \geq_k \phi(p) \vee \phi(q)$.*

Theorem 7.14 *Let ϕ be a closed truth assignment, and for any sentence p containing a free variable x, denote by p_x^t the result of substituting the term t for x. Then for any p and q:*

$$\begin{aligned}
\phi(p \wedge q) &\leq_t \phi(p) \wedge \phi(q) \\
\phi(p \vee q) &\geq_t \phi(p) \vee \phi(q) \\
\phi(p \rightarrow q) &\geq_t \neg\phi(p) \vee \phi(q) \\
\phi(\forall x.p) &\leq_t \mathrm{glb}_t\{\phi(p_x^t) | t \text{ is substitutable for } x \text{ in } p\} \\
\phi(\exists x.p) &\geq_t \mathrm{lub}_t\{\phi(p_x^t) | t \text{ is substitutable for } x \text{ in } p\}
\end{aligned}$$

Corollary 7.15 (Unification) *Let ϕ be a closed truth assignment. Then if t is substitutable for x in p,*

$$\phi(\exists x.p) \geq_t \phi(p_x^t) \geq_t \phi(\forall x.p).$$

Theorem 7.16 (Resolution) *Let ϕ be a closed truth assignment. Then*

$$\begin{aligned}
\phi[(a \wedge d) \rightarrow (b \vee e)] &\geq_t \phi\{[a \rightarrow (b \vee c)] \wedge [(c \wedge d) \rightarrow e]\} \\
&\geq_k \phi[a \rightarrow (b \vee c)] \wedge \phi[(c \wedge d) \rightarrow e].
\end{aligned}$$

8 Inference: Introduction

The work we have presented thus far is very much at odds with the motivation given for the multivalued approach. Our principal argument was that many practical AI programs performing inference could be profitably viewed as the combination of a first-order inference engine and a "bookkeeping" system that combined the answers in some previously unformalized fashion.

We argued that formalizing this bookkeeping would lead to a formalization of just what it was that the programs were doing, and also that the results would allow us to develop effective procedures for a variety of formal systems (such as nonmonotonic inference schemes) that had previously not been implemented. Unfortunately, the nonconstructive nature of the results presented thus far are of little use in this regard. We now address these difficulties.

An implementation of the ideas presented in Section 7 should have the following properties:

1. It should be general-purpose, accepting as inputs the bilattice operators as well as an initial set of facts and a query. The facts and query would then be interpreted using inference on the associated bilattice. A single multivalued inference engine could be used for a wide variety of reasoning tasks.

2. It should be efficient, in the sense that the overhead required by the multivalued approach should be small. If the bilattice operators corresponding to first-order logic are provided, the resulting calculation should be similar to the associated first-order derivation, in order to ensure that the multivalued approach does not introduce computational complexities not otherwise associated with the problem. The same comment holds for ATMS's, default logics, and so on.

3. It should enable us to identify in advance bilattices that will lead to effective computation. Many formal inference schemes (such as default reasoning) are computationally unwieldy. The approach presented in this paper allows us to identify bilattices corresponding to intractable logics.

4. Ideally, the approach will help us to deal with computational intractability in two ways:

   - First, it should be "incremental," so that we can make use of partial analyses or derivations. The need to do a consistency check before asserting a default conclusion is not acceptable in other than toy problem domains. We therefore need to be able to draw tentative conclusions and later retract them if needed.
   - Second, it should point the way toward modifications of the logic that correspond to somewhat more tractable inference.

The construction we are about to present meets all of these requirements.

We begin in Section 9 by discussing some general considerations on multivalued inference. We show that calculating multivalued closure is semi-decidable at best, and investigate the question of under what circumstances we can even hope to find a computationally useful procedure for it. We also continue the discussion begun in Section 7.4 of those situations in which the closure operation is monotonic.

We turn to issues of substance in Section 10, where we discuss inference on *monotonic* bilattices. (Monotonic here refers to an easily identified property of the bilattice itself.) The construction presented generalizes both first-order logic and deKleer's ATMS work, and we show that the computational overhead of the multivalued approach is small in these cases.

We then go on to consider the nonmonotonic case. We begin by presenting a fixed-point description of nonmonotonic inference in Section 11.1. This description is related to existing fixed-point work such as that appearing in [36, 38, 39, 41], but is more robust in the sense that fixed-point extensions always exist in a multivalued setting.

The fixed-point solution will be unique if the bilattice is *stratified*, a point we investigate in Section 11.2. Once again, our notion is related to the more conventional one introduced into the logic programming community by Apt et al. in [1]; once again, the multivalued description is somewhat more general.

A procedure for computing closure in the nonmonotonic case is developed beginning in Section 11.3. We begin by presenting a multivalued analog of the usual approach, which checks consistency before applying any default rules. A specialization to default reasoning is discussed.

In Section 12, we discuss the possibility of improving the efficiency of nonmonotonic inference by using information from the ATMS bilattice to avoid retracting conclusions that remain valid even if some default inferences become unjustified.

9 General considerations

As we have remarked, our current intention is to present a description of inference in a multivalued setting that is suitable for a machine implementation. The difficulty is that the definition of closure given by (11) is hardly suited for this purpose.

A similar problem is faced in implementing first-order inference. Suppose that we consider the following definition of the deductive closure of a set S of sentences:

$$\mathrm{cl}(S) = \bigcap \{T \supseteq S | T \text{ is deductively closed}\}. \tag{13}$$

Since it is impractical to enumerate all of the supersets in (13) when calculating the closure, a proof theory for first-order inference must be developed. This proof theory generally involves axiom schemata such as the tautologies and rules of inference, such as resolution or modus ponens. The proof theory is then shown to correspond to the formal notion of entailment.

We will take a similar approach. If ϕ is some truth assignment and p is some query, so that we are interested in the value of $\mathrm{cl}(\phi)(p)$, we will find a collection of x_i corresponding to the various derivations of

p from other sentences in our language, and such that $\mathrm{cl}(\phi)(p) \geq_k x_i$ for each i. In fact, we will be able to select the x_i in such a way that

$$\mathrm{cl}(\phi)(p) = \sum_i x_i.$$

As we will see in (14) of Section 9.2, this is the multivalued analog to a rule of inference such as modus ponens.

Before proceeding, however, we note the following disheartening result:

Theorem 9.1 *Let B be a nontrivial bilattice, and ϕ a truth assignment defined on B. Then if $p \in L$ is a sentence in our language, determining the value of $\mathrm{cl}(\phi)(p)$ is in general not decidable.*

An examination of the proof of this result shows that the reason for the undecidability is that multivalued inference is a generalization of its first-order counterpart, and first-order inference is itself undecidable.

As we will see, the undecidability of first-order inference is the "only" reason the theorem holds. In other words, it is possible to describe a procedure for calculating multivalued closure that is "algorithmic" in the sense that it relies on a (necessarily nonalgorithmic) sound and complete theorem prover for first-order logic but is otherwise guaranteed to terminate. A sharp description of this procedure, together with a proof of its correctness, is the principal aim of this portion of the paper.

An analogy can be drawn here with Reiter's recent work on diagnosis [42]. There, he presents a formal "algorithm"

> ... for computing all diagnoses for a given faulty system. This algorithm has a number of virtues, not the least of which is its relative independence of the particular logic representing the system being diagnosed. By "relative independence" here we mean that the algorithm assumes the availability of a sound and complete theorem prover for the logic being used, but in all other respects is unconcerned with the underlying logic. A nice consequence of this decomposition is that special purpose theorem provers can be designed for particular diagnostic applications, for example, Boolean equation solvers for switching circuits. Such a special purpose theorem prover can then "hook into" the general purpose algorithm to yield a domain specific diagnostic algorithm. [42, pp. 58–59]

Reiter's remarks apply equally well to the approach we are presenting here.

9.1 Grounded bilattices

Before continuing, there are a variety of formal definitions that will be of use to us in the following material. These are presented in this section and the next; I will attempt to motivate them as well as possible.

The first is the notion of distributivity that was introduced in Definition 4.1. Balanced bilattices are distributive, but the default bilattice D shown in Figure 4 is not, since $(f \cdot *) \vee u = (* \vee u) = dt$, but $(f \vee u) \cdot (* \vee u) = u \cdot dt = u$. A weaker assumption than that of distributivity is the following:

Definition 9.2 *A bilattice will be called* k-respectful *if, whenever $x \geq_k z$ and $y \geq_k z$, then $x \wedge y \geq_k z$ and $x \vee y \geq_k z$. It will be called* t-respectful *if, whenever $x \geq_t z$ and $y \geq_t z$, then $x \cdot y \geq_t z$ and $x + y \geq_t z$. If a bilattice B is k-respectful, t-respectful and k-distributive, it will be called* respectful.

Consider the definition of t-respectfulness. What this is telling us, in part, is that if $x \leq_t z$ and $y \leq_t z$, then $x + y \leq_t z$. Now suppose that $\phi(q) = z$, and that we conclude for two different reasons that $\phi(p) \geq_k x$ and $\phi(p) \geq_k y$. According to each of these separate conclusions, we have $\phi(p) \leq_t \phi(q)$, so that p is less true than q is. The condition of t-respectfulness now ensures that we will continue to hold this belief when we replace the truth value assigned to p with the combined value $x + y$.

The default bilattice, repeated once again in Figure 8, is respectful. Throughout this paper, we will assume that all bilattices under consideration are respectful unless indicated otherwise.

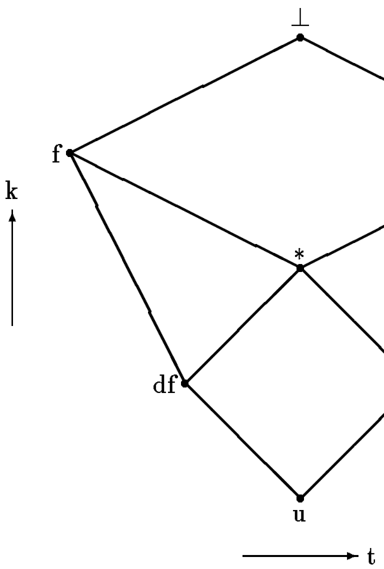

Figure 8: D, the bilattice for a simple default logic

A more interesting notion is that of *groundedness.* It stems from the fact that there are some elements of a bilattice that correspond to "primitive" bits of information; since we are intending to construct the closure of an arbitrary truth assignment by accumulating information from a variety of sources, this is a useful notion to formalize.

Consider again the default bilattice of Figure 8. One can easily imagine concluding, from a single piece of evidence, that some sentence is either true or false, or that it is default true or default false. It is harder to imagine concluding that the sentence has truth value $*$ or $\perp$, however, since these two values correspond to a *combination* of evidence. What distinguishes these two points from the remainder of the bilattice?

The intuitive answer to this is that they are not "on the bottom edge" of the bilattice. Note that this "bottom edge" in fact divides into two portions: that connecting u and t, and that connecting u and f. We will call an element x *t-grounded* if it is on the edge joining u and t, and call x *f-grounded* if it is on the edge joining u and f:

Definition 9.3 *An element x of a bilattice B will be called* t-grounded *if, for any $y \in B$,*

$$y \geq_t x \Rightarrow y \geq_k x.$$

It will be called f-grounded *if, for any $y \in B$,*

$$y \leq_t x \Rightarrow y \geq_k x.$$

x will be called grounded *if it is either t-grounded or f-grounded.*

The following results are all straightforward:

Lemma 9.4 *A point x of a bilattice is t-grounded if and only if $\neg x$ is f-grounded.*

Lemma 9.5 *In a respectful bilattice, if x is t-grounded, then $x \geq_t u$ and $x \leq_k t$. If x if f-grounded, then $x \leq_t u$ and $x \leq_k f$.*

Lemma 9.6 *If B is a respectful bilattice and x and y are t-grounded points of B, then so is $x \vee y$, and $x + y = x \vee y$. If x and y are f-grounded, then so is $x \wedge y$, and $x + y = x \wedge y$.*

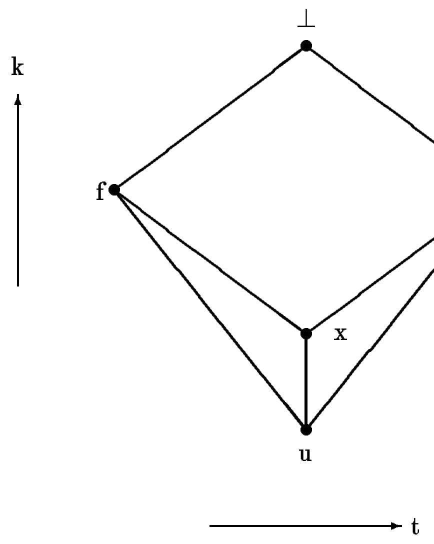

Figure 9: An ungrounded bilattice

Corollary 9.7 *The sum of t-grounded points is also t-grounded. The sum of f-grounded points is f-grounded.*

In a realistic situation, the grounded elements of the bilattice will correspond to the "primitive" bits of information available about the truth or falsity of any particular sentence. It therefore seems reasonable to require that *any* point in the bilattice is the sum of grounded elements:

Definition 9.8 *A respectful bilattice B will be called* grounded at $x \in B$ *if x can be written as a sum of grounded elements of B. A bilattice will be called* grounded *if it is grounded at every point.*

Theorem 9.9 *Let $x \in B$, where B is grounded. Then x can be written as $x_t + x_f$, where x_t is t-grounded and x_f is f-grounded.*

Not every distributive bilattice is grounded; an exception is shown in Figure 9.[9] This bilattice is ungrounded at x; it is hard to imagine what x means in an intuitive sense. Clearly it corresponds to *some* information, but what? It is neither more true nor more false than a lack of information, but cannot be decomposed as a sum of bits of information that have this property. It is because of our inability to interpret this bilattice intuitively that we will be focusing on grounded bilattices throughout this paper.

9.2 Canonical groundings

Theorem 9.9 tells us that in a grounded bilattice, it is possible to "split" any point x into "true" and "false" portions that can then be summed to recover the original point. In this section, we consider the possibility of describing these two portions of x more constructively. We begin by noting that, given x, there is a t-grounded point naturally associated with it:

Definition 9.10 *Let B be a respectful bilattice, and fix $x \in B$. Then the* t-grounding *of x, to be denoted $g_t(x)$, is given by:*

$$g_t(x) = \prod \{y | y \geq_t x\}.$$

[9] In fact, not every *respectful* bilattice is grounded. The default bilattice D, with the k partial order reversed, is an example.

The f-grounding *of* x *is given by:*

$$g_f(x) = \prod\{y | y \leq_t x\}.$$

Lemma 9.11 *For any* x, $g_f(x) = \neg g_t(\neg x)$.

Lemma 9.12 *For any* x, $g_t(x)$ *is* t*-grounded, and* $g_f(x)$ *is* f*-grounded.*

In the default bilattice, for example, we have $g_t(*) = dt$ and $g_f(*) = df$. The grounding functions extract the "primitive" bits of information that make up some compound truth value. In an analogous way, we have $g_t(\bot) = t$ and $g_f(\bot) = f$.

The notion of grounding is closely related to that of inference. To see this, suppose that we have some apparently closed truth assignment ϕ with $\phi(p) = x$ for a sentence p with $p \models q$. What can we say about $\phi(q)$?

Since $p \models q$, we know that $\mathrm{cl}(\phi)(q) \geq_t x$. It follows from this that $\mathrm{cl}(\phi)(q) \geq_k g_t(x)$. In other words, the "contribution" to the truth value of q that can be attributed to p is given by $g_t(x) = g_t[\phi(p)]$.

In general, of course, there will be many sentences of L that entail q. If p' is another one, we can similarly conclude that $\phi(q) \geq_p g_t[\phi(p')]$. We can combine this conclusion with that of the previous paragraph to conclude that

$$\phi(q) \geq_k \sum g_t[\phi(p_i)], \tag{14}$$

where the sum is taken over all p_i that entail q. We will return to this point in subsequent sections.

Before doing so, however, we take advantage of this relationship to define a bilattice to be *monotonic* if the grounding functions are. We are anticipating the k-monotonicity of the associated inference:

Definition 9.13 *A bilattice* B *will be called* monotonic *if, for any* $x, y \in B$ *with* $x \geq_k y$, $g_t(x) \geq_k g_t(y)$.

Lemma 9.14 *In a monotonic bilattice, if* $x \geq_k y$*, then* $g_f(x) \geq_k g_f(y)$.

On a related note, suppose that we have $p_1 \models q$ and $p_2 \models q$, so that $p_1 \vee p_2 \models q$. Suppose also that $\phi(p_1) = x_1$ and $\phi(p_2) = x_2$. Now, arguing as in the previous paragraph and considering the first two entailments, we conclude that $\phi(q) \geq_k g_t(x_1) + g_t(x_2)$.

Alternatively, if $\phi(p_1 \vee p_2) = \phi(p_1) \vee \phi(p_2)$, we can argue from the entailment $p_1 \vee p_2 \models q$ that $\phi(q) \geq_k g_t(x_1 \vee x_2)$. But these two conclusions should give us the same information. We have:

Theorem 9.15 *For any two points* x *and* y *in a respectful bilattice* B,

$$g_t(x) + g_t(y) = g_t(x \vee y),$$

and

$$g_f(x) + g_f(y) = g_f(x \wedge y).$$

Given that g_t and g_f associate with x t-grounded and f-grounded elements of the bilattice, it is natural to ask whether or not they provide the splitting of Theorem 9.9. One direction is trivial:

Lemma 9.16 *For any* x, $g_t(x) + g_f(x) \leq_k x$.

The reverse inequality need not always hold, however. Instead, we define:

Definition 9.17 *A respectful bilattice* B *will be called* canonically grounded at $x \in B$ *if* $x = g_t(x) + g_f(x)$. *A bilattice will be called* canonically grounded *if it is canonically grounded at every point.*

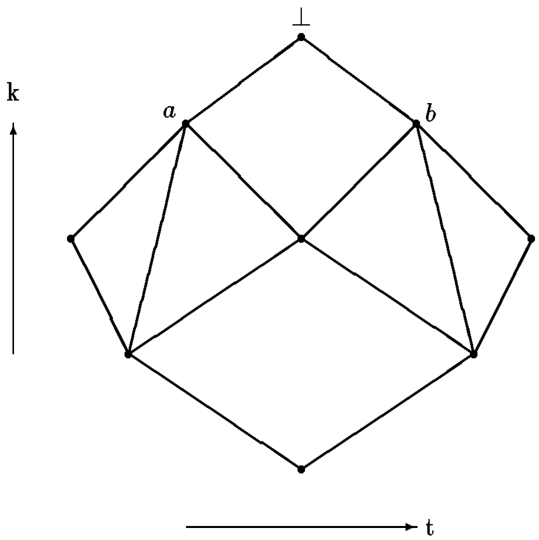

Figure 10: A bilattice that is not canonically grounded

Not every respectful, monotonic, grounded bilattice is canonically grounded; a counterexample is the bilattice shown in Figure 10, which is not canonically grounded at $\perp$, a and b.[10]

Lemma 9.18 *If B is a canonically grounded bilattice and x and y are t-grounded points of B, then so is $x \wedge y$, and $x \cdot y = x \wedge y$. If x and y are f-grounded, then so is $x \vee y$, and $x \cdot y = x \vee y$.*

Lemma 9.19 *Let B be a canonically grounded bilattice. Then for any $x \in B$, $g_t(x) = x \vee u$ and $g_f(x) = x \wedge u$.*

In the discussion leading to (14), we saw that the contribution to the truth value of q arising from a sentence p with $p \models q$ is given by $g_t[\phi(p)]$. The point of the lemma is that the assumption of coherent groundedness allows us to compute $g_t(x)$ in an effective way.

Theorem 9.20 *A canonically grounded bilattice is monotonic if and only if, for any t-grounded a and c and f-grounded b and d,*

$$a + b = c + d$$

implies $a = b$ and $c = d$.

In other words, a bilattice is monotonic if and only if the splitting of Theorem 9.9 is unique.

This result is to be expected. The reason that inference is nonmonotonic on the default bilattice, for example, is that the conclusion that $\phi(p) \geq_k dt$ can potentially be superseded by a subsequent conclusion that $\phi(p) \geq_k f$. The result of combining these two results is that $\phi(p) \geq_k f + dt$, but this is equivalent to simply $\phi(p) \geq_k f$. Although the original conclusion is telling us about a default proof of the truth of p, this information has been lost because $dt + f = f = u + f$. The fact that it is possible to write f in two distinct fashions as the sum of an f-grounded element of D (namely f itself) and a t-grounded element of D (either dt or u) is the source of the nonmonotonicity of inference on the bilattice D.

Theorem 9.20 allows us to completely characterize the set of canonically grounded and monotonic bilattices. Here is the result:

[10] In spite of this, we will generally assume our bilattices to be canonically grounded. All of the examples in this paper have this property.

Theorem 9.21 *Let B be a respectful bilattice. Then the following conditions are equivalent:*

1. *B is distributive.*
2. *B is canonically grounded and monotonic.*
3. *B is a subbilattice of a world-based bilattice.*

10 Monotonic inference

10.1 Theoretical results

In this section we present our first major result on closure, dealing with the case where the closure operation is k-monotonic. As a preliminary, we have the following:

Theorem 10.1 *Let B be a canonically grounded bilattice. Then the closure operation is k-monotonic on B if and only if B is distributive.*

Since we will be discussing the proofs of some sentence q from other sentences p_i, it will be useful to introduce some notation. For a subset U of our language L, we denote by $\phi(U)$ the conjunction of ϕ's values on the elements of U:

$$\phi(U) =_{df} \bigwedge_{p \in U} \phi(p). \tag{15}$$

We also denote by $\pi_+(p)$ the collection of all subsets of L from which it is possible to derive p:

$$\pi_+(p) =_{df} \{U | U \models p\}.$$

Similarly, we use $\pi_-(p)$ to represent the subsets that entail the *negation* of p:

$$\pi_-(p) =_{df} \{U | U \models \neg p\}.$$

Now suppose that we have some truth assignment ϕ whose closure we wish to determine. It is clear from Lemma 6.6 of Section 6 that calculating the "$\neg$-closure" of ϕ is straightforward. In order to simplify the notation in what follows, we will therefore assume that ϕ itself is $\neg$-closed.

Theorem 10.2 *Let B be a distributive bilattice, and suppose that ϕ is some $\neg$-closed truth assignment on B. Then the closure of ϕ is given by:*

$$\mathrm{cl}(\phi)(p) = \sum_{U \in \pi_+(p)} [\phi(U) \vee u] + \sum_{V \in \pi_-(p)} [\neg\phi(V) \wedge u]. \tag{16}$$

The expression (16) can be computed using the following result:

Corollary 10.3 *Suppose that the conditions of Theorem 10.2 are satisfied, and that, in calculating* $\mathrm{cl}(\phi)$, *we have determined that either*

$$\sum_{U \in \pi_+(p)} [\phi(U) \vee u] \geq_k x$$

or

$$\sum_{V \in \pi_-(p)} [\neg\phi(V) \wedge u] \geq_k \neg x.$$

Then any proof of p or $\neg p$ respectively from a set U with $\phi(U) \vee u \leq_t x$ will not contribute to $\mathrm{cl}(\phi)(p)$*. Specifically, no superset of a set underlying a previously discovered proof need be considered, and no proof using any sentence q with $\phi(q) \vee u \leq_t x$ need be considered. A sentence with $\phi(q) \leq_t u$ need never be considered.*

As we will see in the next section, Theorem 10.2 allows us to compute multivalued closure using any sound and complete proof procedure. The difficulty is that as the theorem was expressed, we needed to examine *all* proofs of p in order to determine p's eventual truth value. Corollary 10.3 enables us to make this procedure more efficient by restricting the class of proofs that need to be considered. In addition, in any particular application, we may be interested only in showing that $\mathrm{cl}(\phi)(p) \geq_k x$ for some fixed x, so that we need only attempt to show that $g_t[\mathrm{cl}(\phi)(p)] \geq_k g_t(x)$ and $g_f[\mathrm{cl}(\phi)(p)] \geq_k g_f(x)$. If $g_t(x) = u$ or $g_f(x) = u$, half of the expression (16) will be eliminated. More intuitively, if we are interested only in the truth of p (i.e, we are trying to show that $\mathrm{cl}(\phi)(p) \geq_k t$), then proofs of $\neg p$ need not be considered.

Finally, we can also use Theorem 10.2 to formalize the notion of forward chaining. A conventional forward chainer starts with a database that is assumed to be deductively closed and, when a new sentence p is inserted into that database, forms the deductive closure of the result by also adding to the database those sentences that can be derived from $S \cup \{p\}$ but could not be derived from S alone.

In the multivalued approach, we do not add sentences to the database, but instead modify the truth values assigned to them by our truth assignment ϕ. To formalize this, we denote by $\pi_+^q(p)$ the collection of sets including q that entail p, but that do not entail p if q is removed:

$$\pi_+^q(p) =_{df} \{U | U \models p \text{ and } U - \{q\} \not\models p\}.$$

We define π_-^q similarly, and now have:

Corollary 10.4 (Updates) *Let B be a balanced bilattice, and suppose that ϕ is a closed truth assignment on B. Now if ψ is a $\neg$-closed extension of ϕ that agrees with ϕ except for q, $\neg q$ and their $\neg$-equivalents, then*

$$\mathrm{cl}(\psi)(p) = \phi(p) + \sum_{U \in \pi_+^q(p) \cup \pi_+^{\neg q}(p)} [\phi(U) \vee u] + \sum_{V \in \pi_-^q(p) \cup \pi_-^{\neg q}(p)} [\neg\phi(V) \wedge u].$$

When the truth value assigned to q changes, the change in the truth values assigned to other sentences is that which can be traced to this modification.

10.2 Implementation

In Section 10.3, we will discuss the practical use of Theorem 10.2 and Corollary 10.3 in an inference system. Specifically, we will discuss the use of a general-purpose system to perform inference against both a first-order and an ATMS background. It is not interesting to demonstrate that our formal results can be used to produce special purpose inference engines, since such special purpose tools already exist. Instead, we will discuss the construction of a general purpose tool that automatically specializes to an efficient first-order or ATMS engine.

As described in the introduction, the tool will accept as inputs a description of the bilattice (in terms of the five bilattice operators), together with a database of logical sentences and their truth values. It will also accept a query sentence, the truth value of which is to be determined.

As in existing implementations of logic programming systems, we assume that the bilattice operators and the logical database are already known to the system, so that the tool is invoked by calling a function `query(q)`, where q is the sentence in question.

The following procedure can now be used to reply to a query statement q:

Procedure 10.5 (Monotonic backward inference) Given as input a query q:

1. Set $z = g_t[\phi(q)]$.
2. Find a first-order proof of q from a set U of sentences in the database with $\phi(U) \vee u \not\leq_t z$, using any conventional prover. If none exists, stop and return z.
3. Set $z = z + [\phi(U) \vee u]$ and return to step 2.

Theorem 10.6 (Monotonic backward inference) *Suppose that ϕ is a $\neg$-closed truth assignment on a distributive bilattice B and that q is a sentence of L. Then the result returned by Procedure 10.5 is $g_t[\mathrm{cl}(\phi)(q)]$.*

Theorem 10.7 *Suppose that $\phi(p) \neq u$ for a finite subset of our language L, and that the conventional prover used in Procedure 10.5 always terminates. Then the procedure itself terminates as well.*

Note that the result returned by the procedure is only the t-grounding of the actual value $\mathrm{cl}(\phi)(q)$, so that if $\mathrm{cl}(\phi)(q) = f$, for example, Procedure 10.5 will only tell us that $g_t[\mathrm{cl}(\phi)(q)] = u$. If we want to determine the complete truth value assigned to q, we invoke the procedure twice, once with the original query q and once with $\neg q$. The desired answer is then the sum of the two results obtained.

There are a few additional points we should make concerning the above inference procedure. The first regards the assumption that the truth assignment ϕ be $\neg$-closed.

If ϕ is not $\neg$-closed, it is not difficult to modify the above procedure to handle this. The prover used in step 2 must simply be made somewhat more sophisticated.

A more effective approach, however, is to modify the database tools used to search and update the database so that ϕ is effectively $\neg$-closed at all times. Thus, if $\neg p$ is inserted into the database with truth value t, for example, the actual effect should be to add the unnegated fact p to the database with truth value f. Later, when looking to see if $\neg p$ is in the database, we search for p instead, returning the negation of the stored truth value.

The "conventional prover" mentioned in step 2 should be augmented slightly if the procedure is to be efficient. Assuming that the prover proceeds by attempting to complete "partial" proofs of q using facts in the database, we should only retrieve a fact p from the database for use in this completion if

$$[\phi(p) \wedge \phi(U_p)] \vee u \not\leq_t z, \tag{17}$$

where U_p is the set of database facts already being used in the partial proof. As we successively reinvoke the prover in step 2, z in this expression will be increasing t-monotonically, so that portions of the prover's search space that were pruned because they could not lead to proofs satisfying (17) need not be reconsidered. The most effective way in which to do this is essentially to begin a separate proof process for q, suspending this process during steps 1 and 3 of the procedure and resuming it during step 2. We discuss this point at greater length in Section 14.3, where we discuss the implementation of our ideas.

Finally, I should discuss the invocation of the prover in step 2. As discussed at length in the introduction, a principal aim of the multivalued approach is to split proving efforts from those dealing simply with book-keeping; the call in Procedure 10.5 to an otherwise unspecified first-order inference engine is therefore to be expected. Of course, this causes Procedure 10.5 not to be an "algorithm" in the conventional sense, since the problem of finding a first-order proof of q is in general only semi-decidable. Given an algorithm that provides a satisfactory approximation of first-order inference, it would be accurate to describe Procedure 10.5 as algorithmic.

Also as discussed in the introduction, and as reflected in Reiter's comments quoted in Section 9, the use of a conventional prover gives us some flexibility in implementation. If an incomplete prover is used, the incompleteness will be reflected in the results returned by the procedure, as will the fact that an incomplete prover will often be more efficient than a complete one. In addition, if the first-order prover can be used to find specific solutions to existentially quantified queries (by returning a binding list for the variables that are instantiated), this property can be retained by the multivalued engine. In the looping step, we must maintain a list of variable bindings, with a calculated z for each one.

10.3 Examples

In this section, we examine the computational properties of two specific applications of the procedure of the last section, namely to the bilattices corresponding to first-order logic and to ATMS's.

10.3.1 First-order logic

To perform first-order inference, we provide the general-purpose monotonic inference engine of the last section with the bilattice F and a query q. Here is the procedure from the last section:

1. Set $z = g_t[\phi(q)]$.
2. Find a first-order proof of q from a set U of sentences in the database with $\phi(U) \vee u \not\leq_t z$, using any conventional prover. If none exists, stop and return z.
3. Set $z = z + [\phi(U) \vee u]$ and return to step 2.

It is not too hard to see that this simplifies to the following:

Procedure 10.8 (First-order backward inference) Given as input a query q:

1. If $\phi(q) = t$, then stop and return t.
2. Find a first-order proof of q from a set U of sentences in the database with $\phi(p) = t$ for each $p \in U$, using any conventional prover. If none exists, stop and return u.
3. Stop and return t.

We see that the procedure generated in the first-order case involves simply calling the prover and returning the result. Thus the overhead involved in using the general-purpose engine is small in this case.

10.3.2 ATMS's

ATMS bilattices The bilattices corresponding to assumption-based truth maintenance systems were introduced in Section 6.3.2. ATMS's themselves were developed by deKleer [5] in an attempt to maintain information about circumstances in which various sentences hold.

In an ATMS, the truth value assigned to some sentence contains information concerning potential reasons for its truth or falsity. We captured this in a multivalued setting by using a bilattice in which the truth values consist of pairs $[\ a\ .\ b\ ]$ where a and b are respectively justifications for the truth or falsity of the statement in question. The partial orders on the bilattice are inherited from the partial order on the lattice of justifications.

When using the monotonic procedure on an ATMS bilattice, we are actually performing *backward* inference in an ATMS setting. This is to be contrasted with deKleer's original description in [5], where the ATMS is described as a forward chaining tool.

One of the principal drawbacks of the ATMS approach is the tremendous amount of information generated by the forward chaining process. Indeed, one of deKleer's major contributions in [5] is to provide a computational setting using bit vectors in which this information can be handled tractably. But this information must still be handled within the forward chaining environment provided by the ATMS.

It is possible to use Corollary 10.4 to design a general tool for monotonic inference using forward chaining. Rather than discuss this, however, we investigate the use of our backward procedure in an ATMS setting.

Here is the simplification of the inference procedure for an ATMS bilattice:

Procedure 10.9 (ATMS backward inference) Given as input a query q:

1. Let z be the set of known justifications for q.
2. Find a new justification k in which q holds, where k is not a special case of any justification in z. If none exists, stop and return z.
3. Add k to z and return to step 2.

Once again, a minimal amount of inference is performed.

The above procedure will be inefficient if inconsistent justifications are considered. One possible way around this inefficiency is to use a conventional prover that will not tend to discover a proof of a sentence q from a database inconsistency; almost all implemented provers have this property. It does not appear, however, that there is a straightforward formal solution to this problem.[11]

11 Nonmonotonic inference

Not surprisingly, nonmonotonic inference is much more involved than its monotonic counterpart. Ideally, we would like a nonmonotonic version of (16); unfortunately, in light of the k-nonmonotonicity of inference on nonmonotonic bilattices, this will not be possible.

Instead, we will need to approach the problem more slowly. We view the derivation of (16) as essentially a three step process:

1. The first step involved the characterization of apparently closed truth assignments. In the monotonic case, we have as an immediate consequence of Theorem 10.2 that a truth assignment ψ will be apparently closed if and only if

$$\psi(p) = \sum_{U \in \pi_+(p)} [\psi(U) \vee u] + \sum_{V \in \pi_-(p)} [\neg\psi(V) \wedge u]. \tag{18}$$

2. The second step involved proving that any truth assignment defined on a distributive bilattice had a unique k-minimal apparently closed extension.

3. The third step involved using the result (18) to construct this apparently closed extension explicitly. If we denote the expression on the right hand side of (18) by $F(\psi)(p)$, then it is clear that the F operator is monotonic for distributive bilattices, and we get

$$\mathrm{cl}(\phi)(p) = \sum_{U \in \pi_+(p)} [\phi(U) \vee u] + \sum_{V \in \pi_-(p)} [\neg\phi(V) \wedge u]$$

   as before.

11.1 Fixed-point description

We cannot expect the fixed-point expression in (18) to remain valid in the nonmonotonic case. The reason is that we cannot necessarily conclude from $p_1 \wedge p_2 \models q$ that $\psi(q) \geq_k g_t[\psi(p_1) \wedge \psi(p_2)]$. (As a counterexample, let $\psi(p) = *$, take $p_1 = p$ and $p_2 = \neg p$, and let q be a sentence logically unrelated to p.) Instead, all that we can legitimately conclude is that $\psi(q) \geq_t \psi(p_1 \wedge p_2)$ if ψ is apparently closed. But redefining $\psi(U)$ to be

$$\psi(U) =_{df} \psi(\wedge_i p_i)$$

is unsatisfactory, since this does not reflect the condition on apparent closure that $\psi(\wedge p_i) \geq_k \wedge_i \psi(p_i)$.

A more useful modification is the following: If ψ is a truth assignment, and $U = \{p_i\}$ is a subset of our language L, we denote by $\psi(U)$ the sum of the conjunction of ψ's values on the elements of U *and ψ's value on the conjunction itself*:

$$\psi(U) =_{df} \wedge_i \psi(p_i) + \psi(\wedge_i p_i). \tag{19}$$

We can now prove an analog to (18):

[11] This issue is closely related to deKleer's investigation of nogoods. The description of nogoods within the multivalued framework is discussed in Section 6.3.2.

Theorem 11.1 *Let B be a canonically grounded bilattice, and suppose that ψ is some truth assignment on B. Then ψ is apparently closed if and only if it satisfies:*

$$\psi(p) = \sum_{U \in \pi_+(p)} [\psi(U) \vee u] + \sum_{V \in \pi_-(p)} [\neg\psi(V) \wedge u]. \tag{20}$$

Unlike the analogous monotonic expression (16), the nonmonotonic version (20) does *not* describe the truth assignment $\mathrm{cl}(\phi)$ in terms of the original truth assignment ϕ. As a result, (20) cannot be used directly to construct the closure of a truth assignment ϕ. We will address this point in subsequent sections; here, we only note the similarity between our result and the usual fixed-point descriptions of nonmonotonic inference appearing in default logic [41] or modal descriptions of it [36, 38, 39]. The principal difference between our approach and earlier ones is that (20) in fact describes a substantially more general result, and that the appearance of $\perp$ as a well-defined point of any bilattice guarantees that the fixed-point equation (20) will have at least one solution (namely, the constant function $\perp$) that is an extension of any particular ϕ.

11.2 Stratification

The intuition behind the definitions of closure and of boundedness in Section 7 depended on the idea that the truth values of $*$, *dt* and *df* all correspond to some sort of "default" information, as opposed to the "certain" information corresponding to t, f or $\perp$. Another way to look at this is by realizing that if we identify the four points $*$, *dt*, *df* and u (i.e., we "ignore" all default information), the resulting set inherits a bilattice structure from the original bilattice D (the result is in fact isomorphic to the first-order bilattice F).

After making this identification, we can make a preliminary evaluation of $\mathrm{cl}(\phi)$ by computing the closure on the smaller bilattice consisting of the points t, f, $\perp$ and u (the last being the identification of $*$, *dt*, *df* and u from the original bilattice D). We can then extend the conclusions drawn on the restricted bilattice to obtain $\mathrm{cl}(\phi)$ on the original bilattice.

This approach will be applicable in any situation where we have a bilattice B and a point $x \in B$, and the following set is closed under the bilattice operations:

$$B^x =_{df} \{y \in B | y \not<_k x\}.$$

Definition 11.2 *Let B be bilattice, and fix $x \in B$. We will say that B* splits *at x if B^x is a subbilattice of B.*

Consider the default bilattice, which splits at $*$. This splitting has the following properties:

1. $* = \neg *$.
2. Every point in D^* is $\geq_k *$. Everything left after the bilattice splits corresponds to "certain" information.[12]
3. If $z \neq *$ is a point of D^*, then all of the points in $D_* = \{*, dt, df, u\}$ have the same t-relationship to z. Specifically, t is $>_t$ all of the points in D_*, $f <_t D_*$, and $\perp$ and D_* are t-incomparable.

These properties will actually hold for any splitting of a canonically grounded bilattice. We have:

Lemma 11.3 *If B splits at x, then $x = \neg x$.*

Lemma 11.4 *If B splits at x, then $y \geq_k x$ for every $y \in B^x$.*

Lemma 11.5 *Suppose that B is canonically grounded and splits at x. Now fix $z \in B^x$ with $z \neq x$. If $z \geq_t y$ for some $y \leq_k x$, then $z \geq_t y$ for* every *$y \leq_k x$.*

[12] Except for the splitting point $*$, which corresponds to a *lack* of certain information. This observation relates the current definition of B^x to that appearing in Theorem 6.16.

A similar result holds if $z \leq_t y$ for some $y \leq_k x$. We also have:

Theorem 11.6 *Suppose that B is canonically grounded and splits at x. Then the natural mapping $\pi : B \to B^x$ given by*

$$\pi(y) = \begin{cases} x, & \text{if } y \leq_k x; \\ y, & \text{otherwise.} \end{cases}$$

is a bilattice homomorphism.

The mapping appearing in the theorem is the generalization of the remarks with which we opened this section. It makes precise the idea of treating as unknown the portion of the bilattice below the splitting point x.

The reason splitting is of interest to us is that given a split bilattice, it is possible to compute the closure by working first in the subbilattice B^x, and then modifying the resulting truth assignment to incorporate information about points below x. (This is essentially the content of Lemma A.5 in the appendix.)

Now consider the default bilattice. This bilattice is nonmonotonic because, even though $dt <_k f$, we have $g_t(dt) >_k g_t(f)$. But in some sense, we can expect to be able to deal with the nonmonotonicity because the bilattice splits at $*$. This means that we can calculate the closure first for the points $\perp$, t, f (drawing "certain" conclusions), and then incorporate information about the points $*$, dt, df and u (the default conclusions).

This sort of approach is possible precisely because of the splitting at $*$, and leads us to:

Definition 11.7 *We will call a bilattice* stratified *if, for any $x, y \in B$ with $x \leq_k y$ but $g_t(x) >_k g_t(y)$, there exists a z with $x \leq_k z \leq_k y$ such that B splits at z.*

Apt, Blair and Walker [1] have already used this term in logic programming applications; our two uses are related in that they both order default rules to make it possible to determine under what circumstances one default should be applied preferentially over another.

Lemma 11.8 *Let B be a canonically grounded, stratified bilattice. Then if $x \leq_k y$ but $g_t(x) \not\leq_k g_t(y)$, there exists a z with $x \leq_k z \leq_k y$ such that B splits at z.*

Note that the condition $g_t(x) >_k g_t(y)$ in the definition of stratification has been weakened somewhat in the statement of the lemma.

Not every canonically grounded bilattice is stratified; a counterexample is shown in Figure 11. This bilattice does not split at $*$.

Although the bilattice S shown in Figure 11 is not stratified, it seems that most bilattices of interest will be stratified. The default bilattice D is stratified, as are default bilattices incorporating justification information (as discussed in Section 18 or [24]). If there is a clear prioritization among the truth values, as in prioritized or pointwise circumscription [31, 35] or chronological ignorance [49], the associated bilattice will also be stratified.

Finally, we point out that the stratification requirement is needed for computational reasons only; the definition of closure given by Definition 7.3 remains valid in the nonstratified case. Thus our use of the term is more general than the related usage introduced by Apt et al.[13]

We also have the following result, which enables us to determine easily whether or not a stratified bilattice is in fact monotonic:

Theorem 11.9 *A canonically grounded, stratified bilattice B is monotonic if and only if $t \cdot f = u$.*

Since monotonic inference is far more efficient than its nonmonotonic counterpart, it is useful to be able to make this distinction quickly.

The notion of stratification is useful because of the following result:

[13] The logic programming description of stratification also appears in [15, 14].

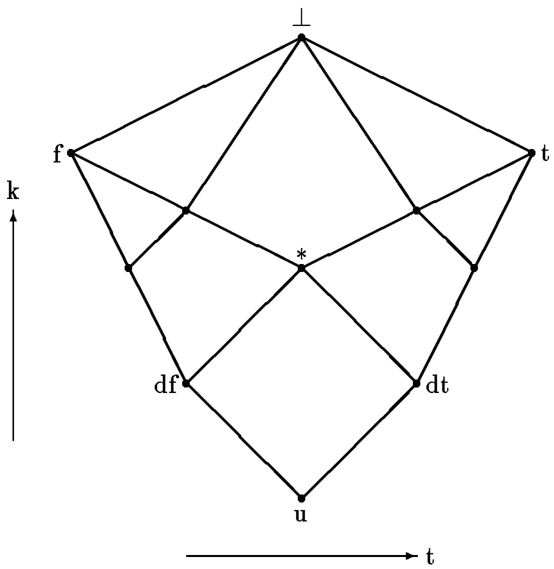

Figure 11: S, a nonstratified bilattice

Theorem 11.10 *Let B be a stratified bilattice. Then any truth assignment ϕ defined on B has a unique $\lhd$-minimal apparently closed extension.*

Because of this, we can replace the problem of discovering all solutions to the fixed-point equation (20) with the simpler problem of finding the unique $\lhd$-minimal apparently closed extension of the truth assignment ϕ under consideration.

11.3 Implementation

In this section, we use the results of Sections 11.1 and 11.2 to describe a procedure for computing closure on nonmonotonic bilattices.

The essence of the procedure is contained in the proof of Theorem 11.10, where we noted that it was possible to calculate closure in a stratified bilattice "from the top down," working one stratification level at a time. Conclusions later derived on lower stratification levels will not affect results computed in this fashion. (Indeed, this observation is essentially the content of Lemma A.5, appearing in the appendix.)

In order to formalize this, we introduce a new relation $\leq_s$ onto our bilattice. The intention is to use $\leq_s$ to capture the notion of "stratification level:"

Definition 11.11 *Let x and y be points of an arbitrary bilattice B. We will say that x and y are* stratification equivalent*, writing $x \approx_s y$, if, for any $z \in B$ such that B splits at z, $x \leq_k z$ if and only if $y \leq_k z$.*

Lemma 11.12 *The relation $\approx_s$ is an equivalence relation on B.*

Definition 11.13 *Given a point x in a bilattice B, the* stratification level *of x, to be denoted $[x]_s$, is defined to be the equivalence class of x under $\approx_s$.*

Lemma 11.14 *The collection of stratification levels in a bilattice B inherits the k partial order of B.*

This allows us to make the following definition:

Definition 11.15 *For an arbitrary bilattice B and points $x, y \in B$, we will write $x \leq_s y$, saying that x's stratification level is* below *that of y, whenever $[x]_s \leq_k [y]_s$.*

Note that $\leq_s$ is a preorder, as opposed to a partial order, since it is possible to have $x \leq_s y$ and $y \leq_s x$ without $x = y$. This will happen whenever x and y are stratification equivalent.

We are now in a position to present a (somewhat) constructive description of nonmonotonic closure. We begin by repeating (19), defining $\psi(U)$, for some set of sentences U and truth assignment ψ, to be:

$$\psi(U) =_{df} \wedge_i \psi(p_i) + \psi(\wedge_i p_i).$$

Here is the monotonic procedure from Section 10.2:

1. Set $z = g_t[\phi(q)]$.
2. Find a first-order proof of q from a set U of sentences in the database with $\phi(U) \vee u \not\leq_t z$, using any conventional prover. If none exists, stop and return z.
3. Set $z = z + [\phi(U) \vee u]$ and return to step 2.

In the nonmonotonic case, the problem is that we also need to investigate the facts in the various sets U, since a change in any of these truth values may affect the validity of a subsequent proof. In the following procedure, we are using $\mathrm{cl}_\neg$ to denote $\neg$-closure, which is easily calculated using Lemma 6.6. Alternatively, our database tool can work with $\neg$-closed truth assignments automatically, as discussed at the end of Section 10.2.

Procedure 11.16 (Nonmonotonic inference I) Given as input a query q:

1. Set $\psi = \phi$. Set $F = \{(q, u)\}$. Each element of F is a pair (p, x), where p is a sentence to be investigated, and x is a point in the bilattice such that we are only interested in proofs of p that result in $\psi(p) >_s x$.
2. For some (p, x) in F, find a first-order proof of p from a set $U = \{p_i\}$ of sentences in the database with $\psi(U) \vee u \not\leq_t g_t[\psi(p)]$ and $\psi(U) >_s x$, using any conventional prover. If no such proof exists, stop and return $\psi(q)$.
3. Set $z = \psi(p)$, and $\psi(p) = \psi(p) + [\psi(U) \vee u]$.
4. If z is on a different stratification level from $\psi(p)$, then set $\psi(r) = \phi(r)$ for any r with $\psi(r) <_s \psi(p)$, and then set $\psi = \mathrm{cl}_\neg(\psi)$. Remove from F any pair (r, x) with $x <_s z$ and $r \neq q$.
5. If $\psi(U)$ is not on the top stratification level of the bilattice, add $(\neg \wedge p_i, \psi(U))$ to F. Return to step 2.

Informally, the procedure proceeds by propagating forward *all* justifications for q, even though some of them may subsequently be overturned when their premises are examined more closely. To ensure that the premises are in fact considered, we maintain a list of sentences still to be investigated; when something new is learned about one of these, we retract all conclusions drawn at lower levels of the bilattice.

The quantification here is important: we retract *all* conclusions drawn at lower levels of the bilattice. When we deduce a new fact with certainty, for example, *all* of our default conclusions become suspect and must be retracted. The reason this is necessary is that we have no way of knowing whether or not some default conclusion depended crucially on a default fact that was overturned; we develop in Section 12 a general method for using justification information to avoid retracting conclusions unnecessarily.

Theorem 11.17 *Let B be a canonically grounded, stratified bilattice. Then for any $\neg$-closed truth assignment ϕ and query q, the value returned by Procedure 11.16 will be $g_t[\mathrm{cl}(\phi)(q)]$.*

Theorem 11.18 *Suppose that $\phi(p) \neq u$ for a finite subset of our language L, that the bilattice B has a finite number of splitting points, and that the conventional prover used in Procedure 11.16 always terminates. Then the procedure itself terminates as well.*

In step 4 of the procedure, we need to determine whether or not $z \approx_s \psi(p)$. This is straightforward, because of the following result:

Theorem 11.19 *Fix z and y in a canonically grounded, stratified bilattice, with $z \geq_k y$ and $z \neq u$. Then z and y are on different stratification levels if and only if $g_t(z) \cdot \neg g_t(z) \not<_k y$ or $g_f(z) \cdot \neg g_f(z) \not<_k y$.*

Determining whether or not $\psi(U)$ is on the top stratification level of the bilattice in step 5 of the procedure is similarly straightforward; we need merely determine whether or not $\psi(U) \approx_s t$.

Note that the nonmonotonic procedure 11.16 retains many of the attractive features of its monotonic predecessor 10.5. Most importantly, it is "incremental," in that the eventual truth value at q is calculated piecemeal, with F used to retain a list of inferences still pending. Tentative conclusions that are justified by default information can be drawn quickly; as long as the inference procedure is eventually run to completion, an accurate final picture will be obtained.

Finally, we note that Procedure 11.16 can in fact also be applied to the product of stratified bilattices;[14] in the resetting step 4, we must simply do the resetting one coordinate at a time. Note that testing to see whether z and $\psi(p)$ are on different stratification levels in any coordinate remains efficient, in light of the following generalization of Theorem 11.19:

Corollary 11.20 *Fix z and y in a canonically grounded product of stratified bilattices, with $z \geq_k y$ and $z \neq u$. Then z and y are on different stratification levels in some coordinate if and only if $g_t(z) \cdot \neg g_t(z) \not<_k y$ or $g_f(z) \cdot \neg g_f(z) \not<_k y$.*

11.4 Example: default reasoning

In this section, we apply Procedure 11.16 to a simple form of nonmonotonic inference, using the default bilattice D.

Before proceeding, however, we note once again that Procedure 11.16 reproduces the monotonic procedure if the bilattice is monotonic. In fact, an examination of the details of the proof of Lemma A.6 in the appendix shows that for monotonic bilattices, the nonmonotonic procedure is slightly *more* efficient than its monotonic predecessor, owing to the inclusion of the additional term $\psi(\wedge_i p_i)$ in the expression (19) for $\psi(U)$.

In the default case, we note that there are two stratification levels in the default bilattice, corresponding to certain conclusions and to default conclusions respectively. The general nonmonotonic procedure becomes:

Procedure 11.21 (Default inference) Given as input a query q:

1. Set $F = \{(q, u)\}$. Set $\psi = \phi$.
2. For some (p, x) in F, find a first-order proof of p from a set $U = \{p_i\}$ of sentences in the database with $\psi(U) \vee u \not\leq_t g_t[\psi(p)]$ and $\psi(U) >_s x$, using any conventional prover. If no such proof exists, stop and return $\psi(q)$.
3. Set $z = \psi(p)$, and $\psi(p) = \psi(p) + [\psi(U) \vee u]$.
4. If $\psi(p) \in \{\perp, t, f\}$ while $z \in \{*, dt, df\}$, then set $\psi(r) = \phi(r)$ for any r with $\psi(r) \in \{*, dt, df\}$, and then set $\psi = \mathrm{cl}_\neg \psi$. Remove from F any pair (r, dt).
5. If $\psi(U) = dt$, add $(\neg \wedge p_i, dt)$ to F. Return to step 2.

We perform backward inference normally, but need to check the consistency of our nonmonotonic assumptions. The procedure above allows us to continue to perform nonmonotonic inference before doing the consistency check; this has the advantage that we can draw default conclusions quickly, but the disadvantage that all of this work may be wasted if we subsequently discover any of our nonmonotonic inferences to have been invalid. Should we know in advance that we either want to draw default inferences as quickly as possible, or that we want to compulsively check the consistency of our nonmonotonic premises, we can modify step 2 to focus the inference in the desired way.

[14] The product of stratified bilattices will not itself be stratified, unless the individual bilattices are distributive.

12 Justification

The difficulty with Procedure 11.16 and its derivatives is in the amount of work it forces us to repeat. Consider the default bilattice, with its two stratification levels. Whenever *any* new nondefault result is obtained, *all* of the default conclusions must be abandoned. Here is the reason for the inefficiency, step 4 from the nonmonotonic procedure:

4. If z is on a different stratification level from $\psi(p)$, then set $\psi(r) = \phi(r)$ for any r with $\psi(r) <_s \psi(p)$ ...

Suppose we consider in somewhat greater detail the reason for the need to reset ψ to its original value. The problem, of course, is that an inference that depended on p may no longer be valid.

It follows that the nonmonotonic procedure can be improved in efficiency if we only retract those results that depended crucially on the original truth value of p in the first case.[15]

12.1 Justified interpretations

In order to pursue this approach, we consider once again the ATMS bilattice discussed in Section 6.3.2. The points in the ATMS bilattice consisted of pairs of justifications; the content of Theorem 6.15 was essentially that these justifications corresponded to legitimate proofs of the sentence being justified. This leads us to the following definition:

Definition 12.1 *Let A be a bilattice, and suppose that we have a mapping π that is a lattice homomorphism from the k-lattice of A into the lattice of justifications $\mathcal{P}[\mathcal{P}(L)]$. We will call π an* interpretation *of A.*[16]

Informally, an interpretation captures justification information that might otherwise be "hidden" inside the bilattice A. If A actually is an ATMS bilattice, for example, so that elements of A are pairs of justifications $[\ a\ .\ b\ ]$, we can take $\pi[\ a\ .\ b\ ] = a$, corresponding to the "true" part of the overall justification.

In order for this idea to be useful, we need to know that π really does reflect justification information. In other words, if p is some sentence with truth value x, and $\pi(x) = j$ for some justification j, then j should justify p. More precisely, if $U \in \pi(x)$ is one of the conjunctive clauses making up the overall justification, we should have $U \models p$.

Definition 12.2 *Let A be a bilattice, and suppose that π is an interpretation for A. We will say that a truth assignment ϕ defined on A is* consistent *with the interpretation if, for any $p \in L$, $U \in \pi[\phi(p)]$ implies that $U \models p$.*

Finally, we need to know that the consistency property is preserved when we perform inference on A. In other words, if ϕ is consistent with the interpretation, then $\mathrm{cl}(\phi)$ should be as well. This is in fact a property not of ϕ, but of the interpretation itself:

Definition 12.3 *We will say that π is a* justified interpretation *for a bilattice A if the closure of any consistent truth assignment is consistent.*

We now have:

Theorem 12.4 *Let A be the ATMS bilattice. Then the interpretation given by $\pi[\ a\ .\ b\ ] = a$ for an element $[\ a\ .\ b\ ]$ of A is a justified interpretation of A.*

[15] And furthermore, we can expect no additional improvement to be possible. If all of the default assumptions underlying a proof become invalid then we would *expect* to need to rederive the proof's conclusion.

[16] This definition is unrelated to the notion of an interpretation that appears in first-order logic. It is the common usage definition of "interpretation" that is being appealed to.

Another justified interpretation of A might define

$$\pi(x) = \emptyset$$

for any $x \in A$. This last is in fact a justified interpretation for any bilattice whatsoever:

Lemma 12.5 *Let B be an arbitrary bilattice, and ϕ a truth assignment on B. Then the interpretation given by $\pi(x) = \emptyset$ for all $x \in B$ is a justified interpretation, and ϕ is consistent with this interpretation.*

We will refer to the interpretation appearing in Lemma 12.5 as the *empty* interpretation.

12.2 Inference using justified interpretations

We now return to Procedure 11.16. In many cases, B may not admit justification information, and we will have no choice but to accept the rederivations forced upon us by the nonmonotonic procedure 11.16. But it is possible that B actually contains justification information, and the following procedure exploits it:

Procedure 12.6 (Nonmonotonic inference II) Given as input a query q, a bilattice B, and a justified interpretation π for B. We assume that ϕ is a $\neg$-closed, consistent truth assignment on B.

1. Set $\psi = \phi$, and $F = \{(q, u)\}$.
2. For some (p, x) in F, find a first-order proof of p from a set $U = \{p_i\}$ of sentences in the database with $\psi(U) \vee u \not\leq_t g_t[\psi(p)]$ and $\psi(U) >_s x$, using any conventional prover. If no such proof exists, stop and return $\psi(q)$.
3. Set $z = \psi(p)$, and $\psi(p) = \psi(p) + [\psi(U) \vee u]$.
4. If z is on a different stratification level from $\psi(p)$, then set
$$\psi(r) = \phi(r) + \sum\{\psi(U) \vee u | U \in \pi[\psi(r)]\} \tag{21}$$
for any r with $\psi(r) \leq_s z$. Set $\psi = \mathrm{cl}_\neg \psi$.
5. If $\psi(U)$ is not on the top stratification level of the bilattice, add $(\neg \wedge p_i, \psi(U))$ to F. Return to step 2.

Theorem 12.7 (Nonmonotonic backward inference) *Let B be a canonically grounded product of stratified bilattices, and π a justified interpretation for B. Then for any $\neg$-closed truth assignment ϕ consistent with this justification and query q, the value returned by Procedure 12.6 will be $g_t[\mathrm{cl}(\phi)(q)]$.*

Theorem 12.8 *Suppose that $\phi(p) \neq u$ for a finite subset of our language L, that the bilattice B has a finite number of splitting points, and that the conventional prover used in Procedure 12.6 always terminates. Then the procedure itself terminates as well.*

There are two points to note about this revised nonmonotonic procedure. First, it is actually a generalization of the previous procedure 11.16, since Lemma 12.5 ensures that we can use the revised procedure with any stratified bilattice and the empty interpretation.[17]

Secondly, note that given a bilattice B without justification information, the new procedure suggests a modification to it that will lead to more effective inference. Specifically, suppose that A is an ATMS bilattice for our language, and is therefore equipped with a justified interpretation as in Theorem 12.4. It is not hard to see the composition of this justified interpretation with the natural projection from $B \times A$ to A is a justified interpretation on $B \times A$. Working with the product bilattice $B \times A$ therefore ensures that justification information is used effectively in the computation.

[17] In fact, this is not quite the case, since the original procedure pruned some pairs from F in step 4, while the current one does not. This can be recovered by maintaining information concerning the reason that various pairs were added to F in the first place.

12.3 Example: default reasoning

We conclude by discussing an application of the full nonmonotonic algorithm, contrasting it with the procedure discussed in Section 11.4.

As in the previous section, Procedure 12.6 reproduces the monotonic procedure if the bilattice being used is monotonic. This is to be expected: The modification that led to Procedure 12.6 from its predecessor 11.16 only involved the resetting step 4, and we have already seen that this step plays no role if B is monotonic.

Here is the default inference procedure from Section 11.4:

Given as input a query q:

1. Set $F = \{(q, u)\}$. Set $\psi = \phi$.
2. For some (p, x) in F, find a first-order proof of p from a set $U = \{p_i\}$ of sentences in the database with $\psi(U) \vee u \not\leq_t g_t[\psi(p)]$ and $\psi(U) >_s x$, using any conventional prover. If no such proof exists, stop and return $\psi(q)$.
3. Set $z = \psi(p)$, and $\psi(p) = \psi(p) + [\psi(U) \vee u]$.
4. If $\psi(p) \in \{\perp, t, f\}$ while $z \in \{*, dt, df\}$, then set $\psi(r) = \phi(r)$ for any r with $\psi(r) \in \{*, dt, df\}$, and then set $\psi = \mathrm{cl}_\neg \psi$.
5. If $\psi(U) = dt$, add $(\neg \wedge p_i, dt)$ to F. Return to step 2.

The new version of this procedure is identical except for step 4, which becomes:

4. If a new nondefault conclusion has been drawn, then reexamine any sentence r with truth value in $\{*, dt, df\}$. For each such sentence, set
$$\psi(r) = \phi(r) + \sum\{\psi(U) \vee u | U \in \pi[\psi(r)]\}.$$
Then set $\psi = \mathrm{cl}_\neg \psi$.

Assuming that we have recorded justification information with any sentence with a default truth value, the above expression allows us to reconstruct that truth value provided that the truth values of premises underlying it have not changed. This example is discussed in greater detail in Section 16.

13 Applications: Introduction

The work we have discussed thus far has reduced the definition of multivalued closure appearing in Definition 7.3 to the execution of the inference procedure 12.6. This procedure realizes our intention of splitting the bookkeeping work from the inference effort; multivalued inference is performed through calls to a conventional first-order prover. As we have already remarked, first-order logic is only semi-decidable; in the inevitable absence of an algorithmic first-order prover, there can be no multivalued algorithm, either.

Nevertheless, as discussed in Section 9, the development of a single procedure that calls bookkeeping functions and a prover as subroutines is precisely the intended result of this research. Use of this general procedure provides enormous flexibility to the developer of an inference system. Should he need to change from conventional first-order logic to an ATMS, for example, the modification is straightforward – he must merely change the bookkeeping functions provided to the system. Should his first-order prover be unsatisfactory for some reason (perhaps the inference tree is recursive, and depth-first search is unacceptable), it, too, can be replaced in a modular way.

The final major portion of this paper discusses a functioning implementation of this approach based on the construction of a multivalued inference engine.[18] As discussed in the next section, such a tool will need to incorporate facilities to:

[18] The system is in the public domain, and runs on a Symbolics 3600 running Genera 7. It is available for a nominal charge from Stanford University; interested parties should contact the author.

1. Accept and modify a database that includes truth values for the sentences being provided,
2. Interface smoothly with a first-order theorem prover and with functions describing the bilattice corresponding to the form of inference being used, and
3. Query the database. Tools should be provided both to simply determine whether or not some particular query q appears in the database, and to determine whether or not q can be derived from the database.

Sections 15 through 18 describe the application of this system to four specific inference systems. For each system, we discuss three issues:

1. The description of the bilattice in question using a variety of LISP functions (the code for which appears in Appendix B),
2. A description of a sample problem to be solved using the given bilattice, and
3. A description of the solution to the problem using the general-purpose inference engine.

The specific inference systems being discussed are first-order logic (Section 15), an ATMS (Section 16), a default logic (Section 17) and a default logic incorporating justification information (Section 18).

14 General considerations

14.1 The database

14.1.1 Structure

The implementation that is the topic of this paper is an enhancement to the MRS logic programming system developed at Stanford [16, 45]. Facts are stored in MRS in prefix normal form, so that $\mathrm{foo}(a) \wedge \mathrm{goo}(b)$ is expressed as `(AND (FOO A) (GOO B))`. An implication such as $f(a) \rightarrow g(a)$ will be represented `(<= (G A) (F A))`.

Variables are prefixed with a $, and are assumed to be universally quantified, as in PROLOG. Existential quantification is handled via Skolemization. Variables in queries are assumed to be existentially quantified, also as is typical in the logic programming community.

In the multivalued approach, we must supply the system with a truth assignment, or function $\phi : L \rightarrow B$, where L is our logical language and B the set of available truth values. Although this function ϕ is assumed to be defined on all of L, in practice we will have $\phi(p) = u$ for most $p \in L$, and we need only specify the values taken by ϕ on sentences with $\phi(p) \neq u$. The actual implementation assigns a proposition number to each sentence pattern, so that `(AND (FOO A) (GOO B))` might be `P324`, for example; the truth value is stored on the property list of this proposition number.

We have argued in a variety of places that it is useful to work only with truth assignments that are $\neg$-closed. (See, for example, Procedure 12.6 and Theorem 12.7.) In order to achieve this, the implementation only stores information about unnegated sentences. Thus, instead of assigning the truth value t to `(NOT (FOO A))`, we would assign the truth value f to `(FOO A)`. This ensures that all truth assignments considered by the system are $\neg$-closed; as a result, negation is treated in a uniform way. Many other logic programming systems (including MRS itself) are unaware of the connection between p and $\neg p$.

14.1.2 Database modification tools

There are two basic tools used to modify the truth assignment being considered by the system; their purposes are to add and remove sentences from the database. In theory, one could "remove" a sentence by making its truth value u; in practice, a function for this purpose is quite useful.

The forms of the two functions are as follows (we are using Common Lisp syntax here; details can be found in [29], for example):

```
mvl-stash   (prop &optional (value (stash-value prop))   increment-p)
mvl-unstash (prop &optional (value (unstash-value prop)) decrement-p)
```

`Stash`, which adds a sentence to the database, accepts as arguments a sentence, a point in the bilattice (the truth value to be assigned to the sentence), and an incrementation flag. If `increment-p` is not nil, the supplied truth value is added to any existing one, using the bilattice $+$ operation. If `increment-p` is nil, the supplied truth value simply replaces any current one. The default value for `increment-p` is `nil`.

If no truth value is supplied, a default value is used that is the result of applying the function `stash-value` to the sentence being stashed. In many cases, `stash-value` will simply return t, ignoring its argument; in others, it may be desirable to use information about the sentence when choosing a truth value.[19]

`Unstash` is similar. If `decrement-p` is `nil`, the sentence is simply removed from the database. If it is not nil, the new truth value assigned to the sentence p is given by:

$$\phi'(p) = \phi(p) \cdot \neg x,$$

where x is the bilattice point supplied by the user. This has the effect of "removing" from ϕ's original truth value the contribution corresponding to x.

14.1.3 Database query tools

As in the original MRS system, there are basically two tools for querying the database. The first, `lookup`, accepts a query sentence q and returns an instantiation of q that can be found in the database.[20] A second procedure, `truep`, returns any instantiations of q that can be *proven* from the information in the database. "Plural" versions exist for both procedures; these return *all* available answers, instead of only one.

Somewhat more delicate questions can be asked in a multivalued setting. In general, for example, we will be interested in finding in the database (or its deductive closure) sentences p with $\phi(p) \geq_k t$. In other words, we look for p that are known to be true. But we may be interested in sentences that are known to be false, requiring $\phi(p) \geq_k f$, or perhaps sentences about which anything is known at all. This would correspond to $\phi(p) >_k u$.

In order to cater to this, the database query tools accept as an optional argument a list of conditions that the truth value of the sentence being retrieved must satisfy. A typical value might be

```
((true .  t-ge));                                         (22)
```

any particular element of the list is a dotted pair, the `car` of which is a point x in the bilattice, and the `cdr` of which is a relation R so that we must have $\phi(p)Rx$ if p is to be returned in response to the query. Thus the list above would return only those p with $\phi(p) \geq_t t$. The allowable relations are `t-ge`, `t-gt`, `t-le`, `t-lt`, their negated versions `t-not-ge` and so on, and similarly for the k and s partial orders. Recall that the s partial order refers to the stratification level of the points in question; the negated versions of the functions are needed because the orders are only partial, and $x \leq_t y$ need not be equivalent to $x \not>_t y$.

We will refer to a list such as that appearing in (22) as a *cutoff*. The arguments to the lookup routine are therefore:

```
mvl-lookup  (prop &optional (cutoff std-cutoff))
mvl-lookups (prop &optional (cutoff std-cutoff))
```

where `std-cutoff` is the default value for the cutoff. The expression appearing in (22) is commonly used for `std-cutoff`.

These functions return three values. The first, as is typical in logic programming languages, is a binding list for the variables in `prop` that need to be instantiated to match some particular database fact. The second

[19] The ATMS bilattice is typical. Here, one often wants to label a new sentence as "self-justified." See sections 6.3.2 and 16.

[20] As already remarked, queries with free variables are assumed to be existentially quantified. If no variables in q need to be instantiated, the binding list `((T . T))` is returned.

is the truth value of the sentence in question; this has no analog in conventional systems. The third value is the sentence being retrieved. This is useful in forming the conjunctions needed in steps 3 and 4 of Procedure 12.6.

Inference is more subtle still. I have remarked elsewhere [20] that one of the attractive features of a system such as the one being discussed here is that it is often possible to curtail inference early, accepting an answer as valid even though the investigation is not complete. We may, for example, wish to accept that Tweety can fly *without* checking to see if Tweety is a penguin or an ostrich, has his feet set in concrete, and so on. In order to cater to this, the query tool that performs inference accepts another cutoff value; as soon as evidence is accumulated indicating that some instantiation of `prop` apparently satisfies this new test, the instantiation is returned.

As an example, suppose that we supplied for this "success" cutoff the value

```
((dt .  t-ge)).
```

If Tweety is a bird, a call to the prover with query `(flies $x)` would return an answer with `$x` bound to `Tweety`, since there *is* a default proof that Tweety can fly. A more careful investigation might show that Tweety cannot fly after all, but this more careful investigation would not be undertaken in our example. This is very much in keeping with the spirit of nonmonotonic reasoning – a default answer can be returned quickly if conclusions must be drawn rapidly.

Here, then, are the proof-type queries:

```
mvl-truep  (prop &optional (cutoff std-cutoff) (success std-success))
mvl-trueps (prop &optional (cutoff std-cutoff) (success std-success))
```

These functions return two values. The first is a binding list and the second a truth value, as for `lookup` and `lookups`. A third value is not returned because a variety of sentences may have been accessed in a variety of fashions during the inference effort.

14.2 Bilattice description

In order to supply the general-purpose inference engine with a bilattice with which to work, the five functions corresponding to the bilattice operations $\wedge$, $\vee$, $+$, $\cdot$ and $\neg$ must be supplied. The implementation expects the functions:

```
mvl-and  (x y)
mvl-or   (x y)
mvl-plus (x y)
mvl-dot  (x y)
mvl-not  (x)
```

Each of these returns a point of the bilattice. Note that the user need *not* specify the elements of the bilattice explicitly; they are simply generated as needed using the bilattice operators above. This has substantial practical import, since the bilattice may not be finite in many applications.[21]

The system also expects a predicate `mvl-eq(x y)` that tests two bilattice points for equality. In some applications (such as an ATMS), there may be alternate representations for the same element of the bilattice, and such a function is needed in order to avoid treating points as different when they are in fact the same.[22]

The system also requires that the user define the four constants `true`, `false`, `unknown` and `bottom` (recall that the bilattice is only defined implicitly in terms of the known elements and the bilattice functions, so

[21] Recall, however, that Procedure 12.6 is only guaranteed to terminate if the number of splitting points *is* finite.

[22] In applications such as these, it is important to keep the function `mvl-eq` as efficient as possible. Since the equality of two justifications is determined by the intersections of the contexts they include, it is possible to ensure that the equality test is at worst $o(n^2)$ in the size of the justifications involved.

these distinguished elements cannot necessarily be determined in any other fashion), and that he supply the default functions `stash-value`, `unstash-value`, `std-cutoff` and `std-success`.

Finally, the user must supply the justification function `pi` that was discussed in Section 12.1. This function takes an element of the bilattice and returns a justification; the justification is an element of $\mathcal{P}(\mathcal{P}(L))$, and is represented as a list of lists.

14.3 The first-order prover

The first-order prover is invoked by step 2 of the inference procedure, which reads as follows:

2. For some (p, x) in F, find a first-order proof of p from a set $U = \{p_i\}$ of sentences in the database with $\psi(U) \vee u \not\leq_t g_t[\psi(p)]$ and $\psi(U) >_s x$, using any conventional prover. If no such proof exists, stop and return $\psi(q)$.

Note first that there are a variety of conditions that need to be satisfied by the set U used by the first-order prover to prove some sentence p. In addition, it is not hard to see that if U is some set that fails to satisfy these conditions, then any superset of U will *also* fail to satisfy them. In other words, we can prune any portion of the prover's search space that lies below some node corresponding to a set U that fails to satisfy the above conditions. Additionally, the prover itself, when looking facts up in the database, need only retrieve facts that have truth values satisfying these conditions. Thus the "lookup" activity of the prover can further prune the search space by passing the above requirements to `mvl-lookups` instead of calling a less general query procedure.

We also note that unless the truth value of p is reset by step 4 of the procedure, the above conditions will become more and more stringent as time passes, since $g_t[\psi(p)]$ will be increasing t-monotonically. It follows from this that any portion of the prover's search space that is pruned at one point in the overall inference process need not be reconsidered the next time the prover is invoked.[23]

The most natural way to implement this is by establishing a separate proof process for each pair (p, x) in F. Rather than restart the prover each time a new proof of p is needed, the proof process for p need only be resumed from whatever state it was in when the last proof was discovered. This is effected using stack groups in the current implementation.[24] The LISP functions used are:

```
mvl-initialize-prover (prop cutoff),
```

which takes a sentence and some initial cutoff information and returns a stack group that has been established to prove `prop`, and

```
mvl-continue-prover (sg cutoff),
```

which accepts a stack group and (possibly updated) cutoff information, and gets the next acceptable answer from the first-order inference engine.[25] The first-order prover needs to be modified to make use of the cutoff information (this is for efficiency reasons only), and to interface properly with `mvl-initialize-prover` and `mvl-continue-prover`. Both of these modifications are straightforward.

15 First-order logic

15.1 The bilattice

The simplest nontrivial bilattice is that corresponding to first-order logic, and is repeated in Figure 12.

[23] In some cases, the prover need not be reset even if the truth value of p is. Briefly, the prover needs to be reinitialized only if some instantiation of p other than the most recently discovered has a truth value that changes stratification level.

[24] Stack groups are not available in Common Lisp, which appears to lack a standard feature for process suspension and resumption. This will affect the portability of the code described here.

[25] `mvl-continue-prover` returns both a binding list and a list of the sentences used in the proof. This information is needed to evaluate $\psi(U)$ in step 3 of the inference procedure.

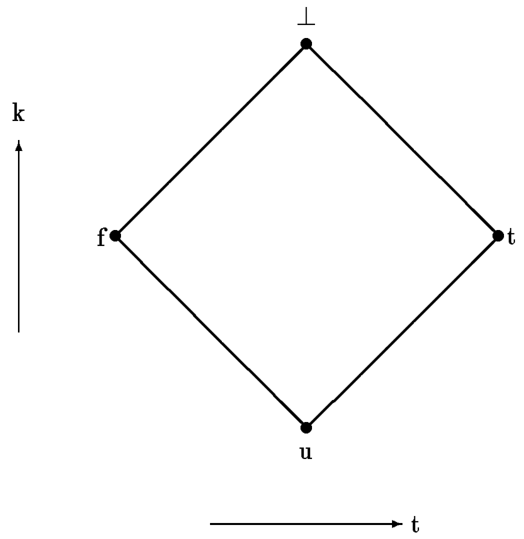

Figure 12: F, the bilattice for first-order logic

The LISP description of this bilattice appears in Appendix B.1. The four points of the bilattice are labelled **true**, **false**, **unknown** and **bottom**, and the various combination functions are all straightforward. The standard value with which a sentence should be stashed is **true**; the standard value for unstashing is **unknown**. An inference or lookup operation should succeed if the truth value found is $\geq_t t$, and there is no justification information available in this bilattice.

15.2 Problem description

The first-order bilattice will be used to solve a simple problem involving the simulation of a digital circuit, as discussed in [17]. This is a useful problem on which to test a theorem prover, and also is the problem most frequently used to benchmark new versions of MRS.

The digital circuit we will investigate is a full adder, introduced by Genesereth in [17] and shown in Figure 13. We quote from [17]:

> A full-adder is essentially a one-bit adder with carry-in and carry-out, and it is usually used as one of n elements in an n-bit adder ... It has three inputs and two outputs and consists of two XOR-gates (`XOR-1` and `XOR-2`), two AND-gates (`AND-1` and `AND-2`), and an OR-gate (`OR-1`) ... In normal operation, the first output (the "sum" line) is "on" if and only if an odd number of inputs is "on"; the second output (the "carry line") is "on" if and only if at least two inputs are "on." [17, p. 414]

An MRS description of the full adder is given in Appendix B.2; the comments give the names associated by the database tool to the various sentences being asserted. We are told what the connections are between the various components, how the components function, and so on, and are also told that the inputs to the full adder are on, off and on, respectively.

It is possible to deduce from this information that the first output of the full adder (the "sum" line) should be off – this is a logical consequence of the information supplied in Appendix B.2. The simulation problem is simply a matter of proving some instantiation of the sentence `(val (out 1 f1) $z)`.

15.3 Problem solution

This problem was presented to the general multivalued inference engine; here is the result:

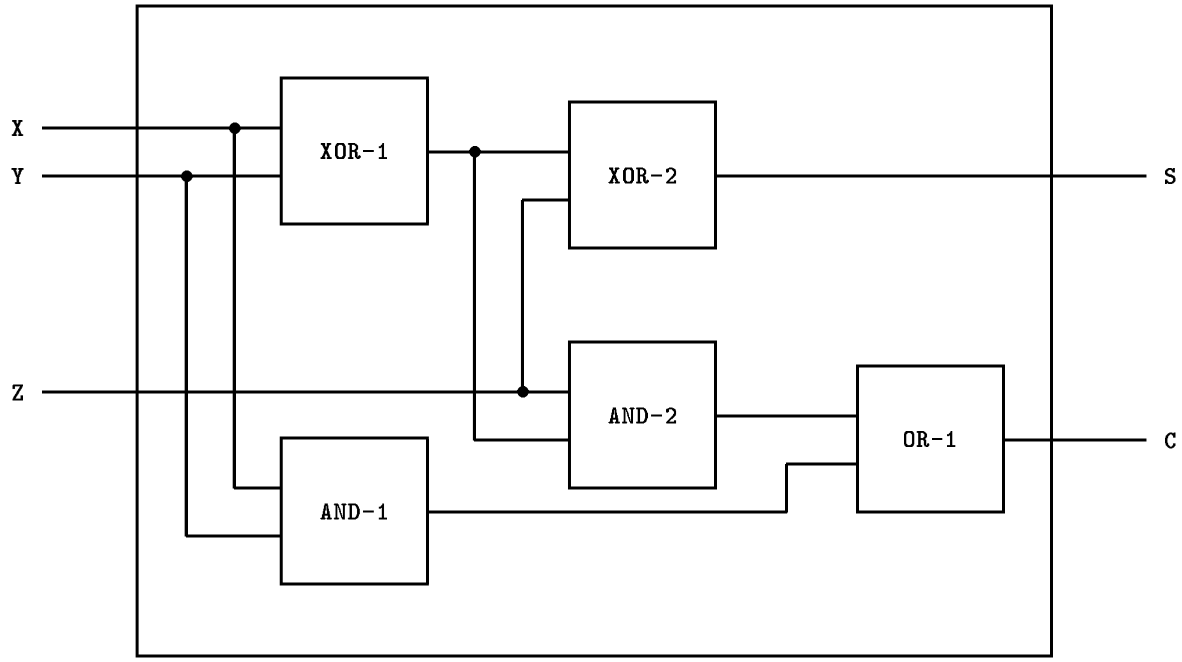

Figure 13: A full adder

```
(mvl-truep '(val (out 1 f1) $z))
     Proof process for (VAL (OUT 1 F1) $Z) being initialized.
     Prover for (VAL (OUT 1 F1) $Z) invoked.
     Truth value of (VAL (OUT 1 F1) OFF) being initialized to TRUE.
     Aborting attempt to prove (VAL (OUT 1 F1) $Z).
(($Z . OFF))
TRUE
```

The indented lines are diagnostics output by the program.

The first thing to note is that the general-purpose engine did indeed solve the simulation problem, with `$z` bound to `OFF`; the truth value obtained was `TRUE`.

Next, note that the proof process was aborted as soon as this answer was found. This is because the truth value `TRUE` passes the default success test for this bilattice. Had we supplied the success test `((TRUE . T-GT))` (which can never succeed), the result would have been as follows:

```
(mvl-truep '(val (out 1 f1) OFF) std-cutoff `((,true . t-gt)))
     Proof process for (VAL (OUT 1 F1) OFF) being initialized.
     Prover for (VAL (OUT 1 F1) OFF) invoked.
     Truth value of (VAL (OUT 1 F1) OFF) being initialized to TRUE.
     Prover for (VAL (OUT 1 F1) OFF) invoked.
     Prover for (VAL (OUT 1 F1) OFF) terminated.
((T . T))
TRUE
```

(The binding list `((T . T))` indicates that no variables needed to be bound in the result.)

In this case, the prover is reinvoked in case there is additional evidence that will affect the final truth value of the statement `(VAL (OUT 1 F1) OFF)`. But note that here, since the criteria in step 2 of the inference procedure 12.6 can never be met,[26] the entire remaining search space can be pruned, and the prover returns immediately.

The time taken to run this example using the multivalued engine was 60% greater than the time normally taken by MRS to solve the problem. This figure reflects the overhead required by the multivalued algorithms, as compared to their first-order counterparts.

16 Assumption-based truth maintenance systems

16.1 The bilattice

As discussed in sections 6.3.2 and 10.3, the ATMS bilattices are constructed from the lattice J of justifications.

Justifications are collections of subsets of L and are partially ordered by taking

$$(a_1 \ldots a_n) \leq (b_1 \ldots b_m)$$

if for each a_i, there is some b_j with $b_j \subseteq a_i$. It is not hard to see that the empty justification () is minimal under this partial order, while ({}) (a justification needing no premises) is maximal. A LISP construction of this lattice is given in Appendix B.3.

The construction of the full ATMS bilattice from the lattice of justifications is also discussed in sections 6.3.2 and 10.3. Briefly, a point of the ATMS bilattice is a pair $[\ j_1\ .\ j_2\ ]$ of justifications, where j_1 is the justification for some sentence p, and j_2 is the justification for its negation. The partial orders are given by:

$$\begin{array}{lll} [\ j_1\ .\ k_1\ ] \leq_t [\ j_2\ .\ k_2\ ] & \text{iff} & j_1 \leq j_2 \text{ and } k_1 \geq k_2, \\ [\ j_1\ .\ k_1\ ] \leq_k [\ j_2\ .\ k_2\ ] & \text{iff} & j_1 \leq j_2 \text{ and } k_1 \leq k_2. \end{array}$$

The LISP code for the ATMS bilattice is shown in Appendix B.4. The values of `std-success` and `std-cutoff` are different here; we accept as true any sentence that can be proved without any assumptions at all, but the database query tools eventually return anything for which a justification can be found. When adding a sentence to the database, its truth value is initially "self-justified." By this we mean that if we are adding some sentence p whose name is `P37`, the justification for p should initially be {{`P37`}}, indicating that p will hold in any context where `P37` holds. The negation of p is initially unjustified.

16.2 Problem description

The idea of using a theorem prover for the purposes of automated diagnosis from first principles is due to Genesereth [17]. In this original paper, however, Genesereth proceeds by explicitly indicating that one (and only one) of the components of the device may be faulty. It is then possible to draw sharp conclusions from an observation that one of the outputs is not as expected.

In our full adder example, suppose that the sum output, instead of being off,[27] is on. Now our database is in fact contradictory; Genesereth suggests that we retract the original device assumptions such as `(XORG X1)`, and replace them with a "single fault assumption:"

```
(OR (XORG X1) (XORG X2))
(OR (XORG X1) (ORG  O1))
(OR (XORG X1) (ANDG A1))
(OR (XORG X1) (ANDG A2))
(OR (XORG X2) (ORG  O1))
(OR (XORG X2) (ANDG A1))
(OR (XORG X2) (ANDG A2))
(OR (ORG  O1) (ANDG A1))
```

[26] The criteria require that $\psi(U) \not\leq_t t$, an impossibility.

[27] Recall that two of the inputs are on, and the third is off.

```
(OR (ORG  O2) (ANDG A2))
(OR (ANDG A1) (ANDG A2))
```

This says that *either* X1 is functioning as an XOR gate, *or* X2 is functioning, and so on. From the fact that (VAL (OUT 1 F1) ON), we can now prove

```
(OR (NOT (XORG X1)) (NOT (XORG X2))),
```

indicating that one of the two XOR gates is malfunctioning, and

```
(AND (ANDG A1) (ANDG A2) (ORG O1)),
```

indicating that the rest of the components are working (this is a consequence of the single fault assumption).

An alternative approach is introduced in [22]. There, it is suggested that the diagnostic task is essentially that of finding a minimal set of device assumptions from which it is possible to prove that some output signal should have a value *other than* its observed one. From the facts that X1 and X2 are XOR gates, it follows that the sum line should be off. If the sum line is on, one of these gates must be faulty. The other components are assumed to be functioning because there is no reason to believe otherwise; this is related to our search for a *minimal* set of device assumptions that are in conflict with the observed failure.[28]

DeKleer and Williams point out in [6] that an ATMS is a natural tool with which to do this sort of analysis. If we can determine the assumptions underlying the conclusion that the sum line should be off, these assumptions will correspond naturally to the diagnosis as discussed in the previous paragraph.

16.3 Problem solution

Given a description of a device used for simulation purposes as in Section 15, modifying it to do diagnosis is extremely straightforward when working with a general purpose inference engine; we need only change from the first-order bilattice used earlier to the ATMS bilattice of Section 16.1. We also need to add axioms to the effect that a line can be either on or off, but not both:

```
(<= (NOT (VAL $L OFF)) (VAL $L ON))
(<= (NOT (VAL $L ON))  (VAL $L OFF))
```

Having done this, we reinvoke the multivalued theorem prover:

```
(mvl-truep '(not (val (out 1 f1) on)))
     Proof process for (NOT (VAL (OUT 1 F1) ON)) being initialized.
     Prover for (NOT (VAL (OUT 1 F1) ON)) invoked.
     Truth value of (NOT (VAL (OUT 1 F1) ON)) being initialized to
          (((P29 P21 P19 P26 P5 P15 P27 P4 P9 P1 P11 P2 P13 P3))).
     Prover for (NOT (VAL (OUT 1 F1) ON)) invoked.
     Prover for (NOT (VAL (OUT 1 F1) ON)) terminated.
((T . T))
(((P29 P21 P19 P26 P5 P15 P27 P4 P9 P1 P11 P2 P13 P3)))
```

The problem is that the justification list returned contains information about *every* assumption made in order to prove the goal (NOT (VAL (OUT 1 F1) ON)). P1, for example, is the database fact (VAL (IN 1 F1) ON); we only want this to be returned as an assumption if the sum line's being low is possibly due to a faulty input. Other elements of the justification are even less likely to be suspect. P29, for example, is the rule indicating that any line that is on is not off.

In order to handle this, we need simply to label some of our database facts as true *without* any assumptions. Thus the fact stating that the first input is on should be true in *all* contexts, corresponding to its truth value being TRUE (i.e., ((NIL) . NIL)), as opposed to (((P1)) . NIL).

[28] There is a clear connection with nonmonotonic inference here. This is also discussed in [22], and in somewhat greater generality by Reiter in [42].

We therefore modify the truth values of the facts describing the full adder as shown in Appendix B.5. (In modification appearing in the Appendix, only the facts describing the functionality of the various components are assumptions. The other facts describing the connections between these components, the inputs to the device, and so on, need not be justified.) When we reinvoke the prover, we obtain:

```
(mvl-truep '(not (val (out 1 f1) on)))
     Proof process for (NOT (VAL (OUT 1 F1) ON)) being initialized.
     Prover for (NOT (VAL (OUT 1 F1) ON)) invoked.
     Truth value of (NOT (VAL (OUT 1 F1) ON)) being initialized to
          (((P5 P4))).
     Prover for (NOT (VAL (OUT 1 F1) ON)) invoked.
     Prover for (NOT (VAL (OUT 1 F1) ON)) terminated.
((T . T))
(((P5 P4)))
```

One of the two XOR gates is faulty.

There are a variety of points to be made here. Note first that, as an implementation of a diagnostician using an ATMS, the multivalued approach inherits the advantages of being able to handle multiple faults [6]. In addition, the code is efficient in comparison with the mechanisms available in MRS to handle justification information. On simple problems, the multivalued approach is 3–4 times faster than the approach taken by the original system. (The original approach involved stashing sentences that gave justification information explicitly, as opposed to incorporating this information in a truth value.)

Our approach is inefficient by comparison with deKleer's system, however. The reason is that deKleer has implemented the lattice operations appearing in Appendix B.3 extremely efficiently, using bit vectors for the logical operations. It would be a simple matter to replace our description of the justification lattice with this more efficient one.

It is also a simple matter to replace the justification lattice with one that represents the justifications in conjunctive normal form, as opposed to the implementation presented above (which represents them in disjunctive normal form). Information dealing with multiple faults is presented somewhat more usefully in this fashion; the difficulty is that the justification function `pi` becomes more complicated.

If this modification is made, the above example becomes:

```
(mvl-truep '(not (val (out 1 f1) on)))
     Proof process for (NOT (VAL (OUT 1 F1) ON)) being
          initialized.
     Prover for (NOT (VAL (OUT 1 F1) ON)) invoked.
     Truth value of (NOT (VAL (OUT 1 F1) ON)) being
          initialized to (((P5) (P4)) . (NIL)).
     Prover for (NOT (VAL (OUT 1 F1) ON)) invoked.
     Prover for (NOT (VAL (OUT 1 F1) ON)) terminated.
((T . T))
(((P5) (P4)) . (NIL))
```

This explicitly lists `P5` and `P4` as the two possible diagnoses of the failure.

The situation is more interesting if we have a more complex fault; suppose that in addition to the sum line being on, the carry line is off. Now we need merely invoke the prover on the sentence

```
(NOT (AND (VAL (OUT 1 F1) ON) (VAL (OUT 2 F1) OFF))).
```

Here is the result. We have replaced the above expression with `(NOT FAULT)` in order to make the output more readable:

```
(mvl-truep '(not fault))
```

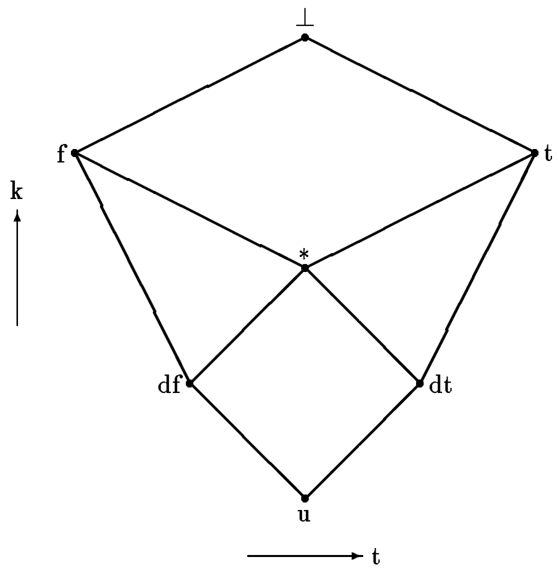

Figure 14: D, the bilattice for default reasoning

```
        Proof process for (NOT FAULT) being initialized.
        Prover for (NOT FAULT) invoked.
        Truth value of (NOT FAULT) being initialized to
             (((P4) (P5)) . (NIL)).
        Prover for (NOT FAULT) invoked.
        Truth value of (NOT FAULT) being changed to incorporate
             (((P4) (P7) (P8)) . (NIL)).
        Prover for (NOT FAULT) invoked.
        Prover for (NOT FAULT) terminated.
   ((T . T))
   (((P4) (P7 P5) (P8 P5)) NIL)
```

We see that there are three possible diagnoses. One involves the failure of the XOR gate `X1`, while the other two involve the combination failures of `X2` and either `A2` or `O1` respectively.

17 Default reasoning

17.1 The bilattice

The bilattice for default reasoning is repeated once again in Figure 14; the LISP code for this bilattice is in appendices B.6 and B.7. The five bilattice functions are described explicitly by giving tables of their results for any possible inputs.

17.2 Problem description

The problem we will investigate using the default bilattice is AI's old chestnut involving birds flying.

By default, birds fly. Ostriches are birds that don't fly. Fred is an ostrich, and Tweety is a bird. To this we add the fact that anything that can fly is not afraid of heights:

```
(mvl-trueps '(flies $x))
     Proof process for (FLIES $X) being initialized.
     Prover for (FLIES $X) invoked.
     Proof process for (AND (NOT (FLIES TWEETY)) (BIRD TWEETY)) being initialized.
     Truth value of (FLIES TWEETY) being initialized to DT.
     Prover for (AND (NOT (FLIES TWEETY)) (BIRD TWEETY)) invoked.
     Prover for (AND (NOT (FLIES TWEETY)) (BIRD TWEETY)) terminated.
     Prover for (FLIES $X) invoked.
     Proof process for (AND (NOT (FLIES FRED)) (BIRD FRED)) being initialized.
     Truth value of (FLIES FRED) being initialized to DT.
     Prover for (AND (NOT (FLIES FRED)) (BIRD FRED)) invoked.
     Truth value of (AND (NOT (FLIES FRED)) (BIRD FRED)) being changed to TRUE.
     Truth value of (FLIES TWEETY) being changed to unknown.
     Truth value of (FLIES FRED) being changed to unknown.
     Aborting attempt to prove (FLIES $X).
     Proof process for (FLIES $X) being initialized.
     Prover for (FLIES $X) invoked.
     Proof process for (AND (NOT (FLIES TWEETY)) (BIRD TWEETY)) being initialized.
     Truth value of (FLIES TWEETY) being initialized to DT.
     Prover for (AND (NOT (FLIES TWEETY)) (BIRD TWEETY)) invoked.
     Prover for (AND (NOT (FLIES TWEETY)) (BIRD TWEETY)) terminated.
     Prover for (FLIES $X) invoked.
     Binding (($X . FRED)) not acceptable because of other evidence.
     Prover for (FLIES $X) invoked.
     Prover for (FLIES $X) terminated.
     Prover for (AND (NOT (FLIES FRED)) (BIRD FRED)) invoked.
     Prover for (AND (NOT (FLIES FRED)) (BIRD FRED)) terminated.

((($X . TWEETY)))
(DT)
```

Figure 15: Who flies?

```
(mvl-stash '(<= (flies $x) (bird $x)) dt)
(mvl-stash '(<= (not (flies $x)) (ostrich $x)))
(mvl-stash '(<= (bird $x) (ostrich $x)))
(mvl-stash '(<= (not (acrophobic $x)) (flies $x)))

(mvl-stash '(ostrich fred))
(mvl-stash '(bird tweety))
```

Note that it is only the default rule that is assigned the truth value **dt**; all of the other facts are inserted into the database with the truth value **true**, since this is given as the value of **stash-value** in Appendix B.6.

We are interested in knowing whether Fred or Tweety flies, and whether or not each is afraid of heights. This second question is rather a subtle one: There is a default proof that Fred is not acrophobic (using the fact that he is a bird and can therefore, by default, fly); even though Fred's not being acrophobic is *consistent* with the given database, it is not a valid conclusion because one of the steps in the proof is unfounded.

17.3 Problem solution

The results of querying the inference engine with **(FLIES $X)** and with **(NOT (ACROPHOBIC $X))** are shown in Figures 15 and 16 respectively. The two examples are in fact quite similar; let us discuss only the second in detail.

```
(mvl-trueps '(not (acrophobic $x)))
```

```
(mvl-trueps '(not (acrophobic $x)))
     Proof process for (NOT (ACROPHOBIC $X)) being initialized.
     Prover for (NOT (ACROPHOBIC $X)) invoked.
     Proof process for (AND (NOT (FLIES TWEETY)) (BIRD TWEETY)) begin initialized.
     Truth value of (NOT (ACROPHOBIC TWEETY)) being initialized to DT.
     Prover for (AND (NOT (FLIES TWEETY)) (BIRD TWEETY)) invoked.
     Prover for (AND (NOT (FLIES TWEETY)) (BIRD TWEETY)) terminated.
     Prover for (NOT (ACROPHOBIC $X)) invoked.
     Proof process for (AND (NOT (FLIES FRED)) (BIRD FRED)) being initialized.
     Truth value of (NOT (ACROPHOBIC FRED)) being initialized to DT.
     Prover for (AND (NOT (FLIES FRED)) (BIRD FRED)) invoked.
     Truth value of (AND (NOT (FLIES FRED)) (BIRD FRED)) being changed to TRUE.
     Truth value of (NOT (ACROPHOBIC TWEETY)) being changed to unknown.
     Truth value of (NOT (ACROPHOBIC FRED)) being changed to unknown.
     Aborting attempt to prove (NOT (ACROPHOBIC $X)).
     Proof process for (NOT (ACROPHOBIC $X)) being initialized.
     Prover for (NOT (ACROPHOBIC $X)) invoked.
     Proof process for (AND (NOT (FLIES TWEETY)) (BIRD TWEETY)) being initialized.
     Truth value of (NOT (ACROPHOBIC TWEETY)) being initialized to DT.
     Prover for (AND (NOT (FLIES TWEETY)) (BIRD TWEETY)) invoked.
     Prover for (AND (NOT (FLIES TWEETY)) (BIRD TWEETY)) terminated.
     Prover for (NOT (ACROPHOBIC $X)) invoked.
     Binding (($X . FRED)) not acceptable because of other evidence.
     Prover for (NOT (ACROPHOBIC $X)) invoked.
     Prover for (NOT (ACROPHOBIC $X)) terminated.
     Prover for (AND (NOT (FLIES FRED)) (BIRD FRED)) invoked.
     Prover for (AND (NOT (FLIES FRED)) (BIRD FRED)) terminated.

((($X . TWEETY)))
(DT)
```

Figure 16: Who is not afraid of heights?

```
          .
          .
(((\$X . TWEETY)))
(DT)
```

The first thing to note is that we ask the prover to find *all* nonacrophobic things, as opposed to simply the first one. Thus the answers returned are lists – the first being a list of binding lists (only one is found, since Tweety is the only object believed not to fly), and the second a list of truth values (DT in this case).

```
     (mvl-trueps '(not (acrophobic $x)))
1        Proof process for (NOT (ACROPHOBIC $X)) being initialized.
2        Prover for (NOT (ACROPHOBIC $X)) invoked.
                 .
                 .
                 .
```

The prover begins by initializing and invoking the first-order prover in an attempt to find a proof that something is not acrophobic. (The numbers appearing in the margin above indicate the step in Procedure 12.6 that corresponds to the given output.) The first-order prover succeeds, and returns information to the effect that Tweety is known by default not to be acrophobic, and that the default sentence (OR (FLIES TWEETY) (NOT (BIRD TWEETY))) was used in the proof. (This is simply a reexpression of one of the rules of inference supplied to the database.)

```
                 .
                 .
                 .
5        Proof process for (AND (NOT (FLIES TWEETY)) (BIRD TWEETY)) begin initialized.
3        Truth value of (NOT (ACROPHOBIC TWEETY)) being initialized to DT.
                 .
                 .
                 .
```

As required by step 2 of the inference procedure, a new proof process is created to consider the negation of this statement, and the truth value of (NOT (ACROPHOBIC TWEETY)) is set to DT.

```
                 .
                 .
                 .
2        Prover for (AND (NOT (FLIES TWEETY)) (BIRD TWEETY)) invoked.
2        Prover for (AND (NOT (FLIES TWEETY)) (BIRD TWEETY)) terminated.
                 .
                 .
                 .
```

The prover next chooses to invoke the proof process that is considering the validity of the proof that Tweety can fly. This proof process terminates without success, since the proof that Tweety is unafraid of heights is in fact valid.

.
.
.

```
2        Prover for (NOT (ACROPHOBIC $X)) invoked.
5        Proof process for (AND (NOT (FLIES FRED)) (BIRD FRED)) being initialized.
3        Truth value of (NOT (ACROPHOBIC FRED)) being initialized to DT.
```

.
.
.

The only remaining proof process is the original one looking for things that are not acrophobic; this proof process is reinvoked, and returns the answer `DT` for Fred as well. The default sentence used in the proof is `(OR (FLIES FRED) (NOT (BIRD FRED)))`. This is used as above to generate the check that the proof that Fred can fly is not invalid. (We will refer to this as the "invalidity test for Fred.")

.
.
.

```
2        Prover for (AND (NOT (FLIES FRED)) (BIRD FRED)) invoked.
3        Truth value of (AND (NOT (FLIES FRED)) (BIRD FRED)) being changed to TRUE.
4        Truth value of (NOT (ACROPHOBIC TWEETY)) being changed to unknown.
4        Truth value of (NOT (ACROPHOBIC FRED)) being changed to unknown.
```

.
.
.

The attempt to prove the negation of the above statement succeeds. At this point, since a new nondefault conclusion has been drawn, the system must retract *all* of its default conclusions. The reason for this is that it has no way of knowing which of the default conclusions may have depended upon a premise that is now known to be invalid. This is done, with the truth values of both `(NOT (ACROPHOBIC TWEETY))` and `(NOT (ACROPHOBIC FRED))` being reset to unknown.[29]

.
.
.

```
2        Aborting attempt to prove (NOT (ACROPHOBIC $X)).
2        Proof process for (NOT (ACROPHOBIC $X)) being initialized.
```

.
.
.

At this point, the attempt to prove `(NOT (ACROPHOBIC $X))` must be restarted. As the system has no way of knowing which nonmonotonic conclusions are no longer valid, it has no way of knowing which *are* valid, and therefore needs to generate them once again.

[29] It is not valid in general simply to reset to unknown the instantiation that generated the consistency check. Between the time the consistency check is generated and the time it is actually examined, another default proof that Fred is unafraid of heights may have been found.

```
            .
            .
            .

2       Prover for (NOT (ACROPHOBIC $X)) invoked.
5       Proof process for (AND (NOT (FLIES TWEETY)) (BIRD TWEETY)) being initialized.
3       Truth value of (NOT (ACROPHOBIC TWEETY)) being initialized to DT.
2       Prover for (AND (NOT (FLIES TWEETY)) (BIRD TWEETY)) invoked.
2       Prover for (AND (NOT (FLIES TWEETY)) (BIRD TWEETY)) terminated.
2       Prover for (NOT (ACROPHOBIC $X)) invoked.
2       Binding (($X . FRED)) not acceptable because of other evidence.
2       Prover for (NOT (ACROPHOBIC $X)) invoked.
2       Prover for (NOT (ACROPHOBIC $X)) terminated.

            .
            .
            .
```

This is done; the second pass through the proof effort is much like the first, until Fred is generated as a candidate nonacrophobe. Now the invalidity test for Fred is *known* to be valid, so that $\psi(U) = f$ and the requirement that $\psi(U) \vee u \not\leq_t g_t[\psi(p)]$ fails. It follows that the truth value for Fred should not be updated, and the system reports that binding **$X** to Fred is "not acceptable because of other evidence." At this point, the proof process for **(NOT (ACROPHOBIC $X))** is reinvoked, but no new solutions are found.

```
            .
            .
            .

2       Prover for (AND (NOT (FLIES FRED)) (BIRD FRED)) invoked.
2       Prover for (AND (NOT (FLIES FRED)) (BIRD FRED)) terminated.

   ((($X . TWEETY)))
   (DT)
```

Finally, the proof process for the invalidity test for Fred is reinvoked. After all, it is possible that we are working with a multi-level default system, and we only know *by default* that the invalidity test holds (perhaps because we only know by default that Fred is an ostrich). If additional investigation shows that Fred is in fact *not* an ostrich, we will need to reconsider our uncertainty about his acrophobia. In this particular example, since the conclusion about the invalidity test is at the top level of the bilattice, the prover terminated with failure immediately.

Finally, we remark that the inference procedure used to calculate these answers is provably correct, in that it is demonstrably computing the bilattice closure operation using the default bilattice.

18 Default reasoning with justification information

18.1 The bilattice

As is clear from the description in Section 17.3, the lack of justification information forced the system to duplicate a substantial amount of work when the default proof that Fred was unafraid of heights was found to be invalid.

In order to avoid this, we use a bilattice that includes both default and justification information. To do this, we simply define a new bilattice $B = D \times A$, where D is the default bilattice, and A is the ATMS bilattice. In LISP, points of B are represented as dotted pairs $[\ d\ .\ a\ ]$, where $d \in D$ and $a \in A$. All of the

```
(mvl-trueps '(not (acrophobic $x)))
     Proof process for (NOT (ACROPHOBIC $X)) being initialized.
     Prover for (NOT (ACROPHOBIC $X)) invoked.
     Proof process for invalidity test for Tweety being initialized.
     Truth value of (NOT (ACROPHOBIC TWEETY)) being initialized to
          (DT ((P60 P57 P62))).
     Prover for invalidity test for Tweety invoked.
     Prover for invalidity test for Tweety terminated.
     Prover for (NOT (ACROPHOBIC $X)) invoked.
     Proof process for invalidity test for Fred being initialized.
     Truth value of (NOT (ACROPHOBIC FRED)) being initialized to
          (DT ((P60 P57 P59 P61))).
     Prover for invalidity test for Fred invoked.
     Truth value of invalidity test for Fred being changed to
          (TRUE ((P58 P61 P59))).
     Truth value of (NOT (ACROPHOBIC FRED)) being changed to
          (UNKNOWN ((P61 P59 P57 P60))).
     Prover for (NOT (ACROPHOBIC $X)) invoked.
     Prover for (NOT (ACROPHOBIC $X)) terminated.
     Prover for invalidity test for Fred invoked.
     Prover for invalidity test for Fred terminated.

((($X . TWEETY)) (($X . FRED)))
((DT ((P62 P57 P60))) (UNKNOWN ((P61 P59 P57 P60))))
```

Figure 17: The acrophobia example with justification information

bilattice operations are computed component by component, except for the justification function **pi**, which is given by:

```
(defun pi (x) (cadr x)),
```

which returns the **car** of the **cdr** of x. In other words, it returns the positive part of the justification information included in the ATMS component of the truth value.

18.2 The problem

Describing our default scenario using this product bilattice is straightforward:

```
(setq p '(<= (flies $x) (bird $x)))
(mvl-stash p (cons dt `(((,(prop-name p))) . nil)))
(mvl-stash '(<= (not (flies $x)) (ostrich $x)))
(mvl-stash '(<= (bird $x) (ostrich $x)))
(mvl-stash '(<= (not (acrophobic $x)) (flies $x)))

(mvl-stash '(ostrich fred))
(mvl-stash '(bird tweety))
```

It takes some effort to get the correct truth value for the statement that birds fly by default (namely, that it is true by default and self-justified); other than this, the above assertions are identical to those of the Section 17.

18.3 Problem solution

The results of invoking the prover with the query **(NOT (ACROPHOBIC $X))** are shown in Figure 17.

Now, when the proof about Fred is found to be invalid, the truth value of **(NOT (ACROPHOBIC FRED))** is changed to **(UNKNOWN ((P61 P59 P57 P60)))**. The justification term here records the fact that **P61** etc. make up a context in which Fred *would* be nonacrophobic, although the context is currently believed to be inconsistent. The truth value for the instantiation involving Tweety need not be changed, and the prover does not need to be reset. Eventually, two answers are returned, since something *is* known about Fred's mental attitudes – namely, that if only P61, P59, P57 and P60 all held, he would be unafraid of heights.

19 Summary and future work

19.1 Summary

In this paper, we have developed a general framework for performing multivalued inference. The principal theoretical contributions of the paper have been the following:

1. The development and analysis of a new mathematical structure known as a *bilattice*, including the characterization theorem 9.21 that completely identifies a wide class of bilattices.
2. The development of sharp mathematical results describing the inference itself, including the following:
   (a) The demonstration that inference is monotonic on a canonically grounded bilattice if and only if the bilattice itself is monotonic.
   (b) For nonmonotonic bilattices, the presentation of a fixed-point description of closure that is a generalization of similar descriptions in modal or default approaches. We also described a general class of bilattices, called *stratified* bilattices, for which any truth assignment has a unique minimal extension satisfying this fixed-point equation.
3. The demonstration that with specific choices of bilattice, the inference procedure developed generalizes the following existing ideas:
   (a) First-order logic, as discussed in sections 6.3.1, 10.3.1 and 15.
   (b) ATMS's, as discussed in sections 6.3.2, 10.3.2 and 16.
   (c) Nonmonotonic reasoning, as discussed in sections 7.3.2, 11.4, 12.3, 17 and 18. The application of our ideas to the construction of a circumscriptive theorem prover is discussed in [24].

The practical contributions that rest on the theoretical results are more substantial. We presented a procedure for monotonic inference that generalizes both first-order inference and earlier work on assumption-based truth maintenance systems, and showed both that the generalization was achieved without substantial computational overhead and that the procedure could be implemented as a straightforward enhancement to any existing first-order prover.

In the nonmonotonic case, we first presented a procedure that was shown to be a generalization of existing procedures for default inference. A further generalization of this procedure to incorporate justification information leads to computational procedures capable both of drawing default inferences quickly and of withdrawing them gracefully if subsequent analysis shows their premises to be unfounded.

We went on to demonstrate the applicability of an implementation or our ideas to a wide variety of deductive problems. Tailoring the general-purpose tool to any of the specific inference tasks considered involved simply replacing the set of bookkeeping functions used by the general-purpose inference engine.

19.2 Future work

Future work will involve the construction of bilattices corresponding to other sorts of inference. Some obvious candidates are:

- Bilattices incorporating probabilistic information.

- Bilattices corresponding to context hierarchies. Although it is possible to use an ATMS for this purpose, it is often useful to be able to work with a system capable of dealing with conflicting theories without resorting to the powerful (and computationally expensive) machinery involved in an ATMS.

- A bilattice corresponding to circumscription. This idea is developed in [24].

Additionally, forward inference in the multivalued framework needs to be explored. Although Corollary 10.4 presents a forward-inferencing procedure for cases where the bilattice being considered is monotonic, a general procedure has yet to be developed. Developing and implementing a multivalued forward chainer is another area where work is needed.

Most importantly, however, the true applicability of the multivalued approach has yet to be demonstrated. What we have done is to show that a variety of existing inference systems can be implemented quickly using the multivalued approach. But the real applications are not new inference *tools*, but new inference programs that use these tools. It is here that the true value of the ideas we have presented will become apparent – and here that the real work remains to be done.

A Proofs

Lemma 4.2. *In any bilattice, $\neg t = f$ and $\neg f = t$.*

Proof. Since t is the maximal element of the bilattice under the partial order $\geq_t$, it follows that $\neg t$ is the minimal element under this partial order, so that $\neg t = f$. That $\neg f = t$ is similar. □

Lemma 4.3. *In any nontrivial bilattice, t and f are k-incomparable.*

Proof. If we had, for example, $t \geq_k f$, then we would have $\neg t \geq_k \neg f$, or $f \geq_k t$. It would follow that $t = f$, and B would be trivial. □

Lemma 4.4. *For any set W, $(B_W, \wedge, \vee, \cdot, +, \neg)$ is a bilattice.*

Proof. The only thing to check is that if $(U,V) \geq_t (R,S)$, then $\neg(U,V) \leq_t (R,S)$, etc. But

$$\begin{aligned} (U,V) \geq_t (R,S) &\rightarrow U \supseteq R \wedge V \subseteq S \\ &\rightarrow V \subseteq S \wedge U \supseteq R \\ &\rightarrow (V,U) \leq_t (S,R). \quad \square \end{aligned}$$

Lemma 5.4. *Let ϕ and ψ be two W-closed truth assignments on a bilattice B_W. Then $\phi \cdot \psi$ is also W-closed. In general, if $\{\phi_i\}$ are all W-closed, then so is*

$$\prod_i \phi_i = \phi_1 \cdot \ldots \cdot \phi_n.$$

Proof. For the first condition of Definition 5.3 of W-closure, we know that $\tilde{\phi}_{i+}$ is closed for each i, and it is not hard to see that

$$\widetilde{\left(\prod_i \phi_i\right)}_+ = \cap_i \tilde{\phi}_{i+}.$$

Since the intersection of logically closed sets is logically closed, it follows that $\widetilde{(\prod_i \phi_i)}_+$ is logically closed as well.

For the second condition, we have

$$\left[\prod_i \phi_i\right](\neg p) = \prod_i \phi_i(\neg p) = \prod_i [\neg \phi_i(p)] = \neg \prod_i \phi_i(p). \quad \square$$

Corollary 5.6. *If ϕ is a truth assignment defined on a world-based bilattice, then $\mathrm{cl}_W(\phi)$ is W-closed.*

Proof. Immediate from the lemma. $\square$

Lemma 6.2. *A truth assignment is apparently closed if and only if it is W-closed.*

Proof. We first show that any W-closed truth assignment ϕ is apparently closed. If $p \models q$, q will be true in at least as many worlds as p is, and can be false in no more. Thus $\phi(q) \geq_t \phi(p)$.

Next, we note that if p_i is true in U_i and false in V_i for each of a collection of p_i, then the conjunction $\wedge_i p_i$ will certainly be true in the intersection of the U_i, and will certainly be false in the union of the V_i. Since

$$\wedge_i (U_i, V_i) = (\cap_i U_i, \cup_i V_i),$$

we have that $\phi(\wedge_i p_i) \geq_k \wedge_i \phi(p_i)$.

Satisfaction of the third condition is immediate, since it is shared by the two definitions of closure.

For the converse, suppose that ϕ is apparently closed, but not W-closed. Then since $\phi(\neg p) = \neg\phi(p)$ for all p, there must be some $w \in W$ such that $\tilde{\phi}_+(w)$ is not W-closed. There must therefore be some collection $p_1, \ldots, p_n$ with $p_i \in \tilde{\phi}_+(w)$ and $p_1, \ldots, p_n \models p$ but $p \notin \tilde{\phi}_+(w)$.

But since ϕ is apparently closed,

$$\wedge_i \phi(p_i) \leq_k \phi(\wedge_i p_i) \leq_t \phi(p).$$

It follows that

$$\cap_i \phi_+(p_i) \subseteq \phi_+(\wedge_i p_i) \subseteq \phi_+(p),$$

or that $w \in \phi_+(p)$, so that $p \in \tilde{\phi}_+(w)$ after all. This contradiction completes the proof. $\square$

Theorem 6.3. $\mathrm{cl}_W(\phi) = \prod\{\psi | \psi \geq_k \phi$ *and* ψ *is apparently closed*$\}$.

Proof. Immediate from the lemma. $\square$

Lemma 6.6. *A truth assignment ϕ is $\neg$-closed if and only if*

$$\phi(p) = \sum\{\phi(q) | q \equiv_\neg p\} + \sum\{\neg\phi(q) | q \equiv_\neg \neg p\}$$

for all p.

Proof. It is clear that any ϕ satisfying the above expression will be $\neg$-closed; the converse is also clear (by induction). $\square$

Lemma 6.8. *Any balanced bilattice is distributive.*

Proof. It clearly suffices to show that any world-based bilattice is distributive. For this, we have:

$$\begin{aligned}
[(P,Q) + (R,S)] \wedge (U,V) &= (P \cup R, Q \cup S) \wedge (U,V) \\
&= [(P \cup R) \cap U, (Q \cup S) \cup V] \\
&= [(P \cap U) \cup (R \cap U), (Q \cup V) \cup (S \cup V)] \\
&= (P \cap U, Q \cup V) + (R \cap U, S \cup V) \\
&= [(P,Q) \wedge (U,V)] + [(R,S) \wedge (U,V)],
\end{aligned}$$

and so on. $\square$

Theorem 6.9. *Let (h_1, W_1) and (h_2, W_2) both balance a bilattice B. Then if ϕ is a truth assignment on B, and χ_i are truth assignments on B_{W_i} for $i = 1, 2$, with $h_i(\chi_i) = \phi$,*

$$h_1[\mathrm{cl}(\chi_1)] = h_2[\mathrm{cl}(\chi_2)]$$

(and both are equal to $\mathrm{cl}_B(\phi)$). In fact,

$$\mathrm{cl}_B(\phi) = \prod\{\psi | \psi \geq_k \phi \text{ and } \psi \text{ is apparently closed}\}. \tag{23}$$

Proof. The entire theorem in fact follows from (23). And this will follow from the following two results:

1. If χ is an apparently closed truth assignment on B_W, then $h(\chi)$ is an apparently closed truth assignment on B, and

2. If ψ is an apparently closed truth assignment on B, there exists an apparently closed χ on B_W with $h(\chi) = \psi$.

The first of these is straightforward, since h, as a bilattice homomorphism, will preserve the conditions defining apparent closure.

The second is substantially more involved. We first fix an inverse to h, choosing h^{-1} to be any bilattice homomorphism from B to B_W such that $h(h^{-1}(x)) = x$. (We might, for example, choose h^{-1} to be

$$h^{-1}(x) = \prod\{y \in B_W | h(y) = x\}$$

for any $x \in B$.)

In order to prove the theorem, we need only show that $h^{-1}(\phi)$ is both $\neg$-closed and $+$-closed. We first show that it is $\neg$-closed.

To see this, we have that

$$\begin{aligned}&\sum\{h^{-1}(\phi)(q) | q \equiv_\neg p\} + \sum\{\neg h^{-1}(\phi)(q) | q \equiv_\neg \neg p\} \\ &\quad = h^{-1}\left(\sum\{\phi(q) | q \equiv_\neg p\} + \sum\{\neg\phi(q) | q \equiv_\neg \neg p\}\right) \\ &\quad = h^{-1}(\phi)(p).\end{aligned}$$

It remains to show that $h^{-1}(\phi)$ is $+$-closed. If this were not the case, there would need to be some sentence p such that the "$+$-closure" of $h^{-1}(\phi)$, evaluated at p, was distinct from $h^{-1}(\phi)(p)$. In order for this to happen, there must be some proof of p that forces its truth value away from $h^{-1}(\phi)(p)$ in the $+$-closure. Suppose that this proof depended upon the sentences $q_1, \ldots, q_n$; we denote $(h^{-1}(\phi))_+(q_i)$ by Q_i, so that Q_i is the set of worlds where q_i is known to hold. Now in order for $h^{-1}(\phi)$ not to be $+$-closed, we would need to have

$$\cap_i Q_i \not\subseteq (h^{-1}(\phi))_+(p). \tag{24}$$

Now set $q = \wedge_i q_i$. Since ϕ is apparently closed, $\phi(q) \geq_k \wedge_i \phi(q_i)$, and it follows that if x and $\{y_i\}$ are elements of B_W with $h(x) = \phi(q)$ and $h(y_i) = \phi(q_i)$ for each i, then $x \geq_k \wedge_i y_i$, so that

$$x_+ \supseteq \cap_i y_{i+}.$$

Again since ϕ is apparently closed and $q \models p$, if z is an element of B_W with $h(z) = \phi(p)$, we must have

$$z_+ \supseteq x_+ \supseteq \cap_i y_{i+}.$$

Finally, we have that $h(h^{-1}(\phi)(q_i)) = \phi(q_i)$, so that we can take y_{i+} to be Q_i, and it follows that

$$(h^{-1}(\phi))_+(p) \supseteq \cap_i Q_i.$$

This is in conflict with (24) and completes the proof. □

Corollary 6.10. *For any truth assignment ϕ, $\mathrm{cl}_B(\phi) \geq_k \phi$ and $\mathrm{cl}_B(\mathrm{cl}_B(\phi)) = \mathrm{cl}_B(\phi)$.*

Proof. The first statement is clear: Any apparently closed extension of ϕ will be an extension of ϕ. For the second, note that any apparently closed extension of ϕ will be an apparently closed extension of $\mathrm{cl}_B(\phi)$, and vice versa. Thus $\mathrm{cl}_B(\mathrm{cl}_B(\phi)) = \mathrm{cl}_B(\phi)$. □

Corollary 6.11. *If ϕ is a truth assignment defined on a balanced bilattice, $\mathrm{cl}_B(\phi)$ is apparently closed.*

Proof. In the notation of Theorem 6.9, $\mathrm{cl}(\chi_1)$ is W-closed, and therefore apparently closed. Thus $h[\mathrm{cl}(\chi_1)]$ is apparently closed as well. □

Corollary 6.12. cl_B *is k-monotonic. In other words, if $\phi \leq_k \psi$, then $\mathrm{cl}_B(\phi) \leq_k \mathrm{cl}_B(\psi)$.*

Proof. This follows immediately from the definition of balanced closure (4): The set over which the product is taken varies k-monotonically with the truth assignment in question. □

Theorem 6.13. *Let S be a set of sentences, and q a sentence. Then:*

$$\mathrm{cl}_B\phi_S(q) = \begin{cases} \bot, & \text{if } S \text{ is inconsistent;} \\ t, & \text{if } S \text{ is consistent and } S \models q; \\ f, & \text{if } S \text{ is consistent and } S \models \neg q; \\ u, & \text{otherwise.} \end{cases} \tag{25}$$

Proof. We first deal with the case where S is inconsistent, so that $S \models p$ for all p. There will then be some conjunction $p_1 \wedge \cdots \wedge p_n$ of sentences in S that is inconsistent, so that

$$p_1 \wedge \cdots \wedge p_n \models q$$

for all q. But for any apparently closed extension ψ of ϕ_S, since $\phi_S(p_i) = t$ for all i, we must have

$$\psi(\wedge_i p_i) \geq_k \wedge_i \phi_S(p_i) = t,$$

so that $\psi(\wedge_i p_i) \in \{t, \bot\}$. Since $\wedge_i p_i \models q$ for all q, we have

$$\psi(q) \geq_t \psi(\wedge_i p_i),$$

so that $\psi(q) \in \{t, \bot\}$ as well. But we also have that $\psi(\neg q) \in \{t, \bot\}$ and $\psi(q) = \neg\psi(\neg q)$; these together imply that $\psi(q) = \psi(\neg q) = \bot$ for all q. Thus

$$\mathrm{cl}_B(\phi_S) = \bot$$

(the constant truth assignment $\bot$) if S is inconsistent.

If S is consistent, then the result is immediate, since the truth assignment appearing in (25) is apparently closed, and it is clear that any apparently closed extension of ϕ_S must be an extension of (25). □

Corollary 6.14. *Let S and T be two sets of sentences. Then the following two conditions are equivalent:*

1. $T \subseteq \mathrm{cl}(S)$
2. $\phi_T \leq_k \mathrm{cl}_B(\phi_S)$.

Proof. Obvious. □

Theorem 6.15. *Let S be the set of assumptions in our knowledge base, and suppose that ϕ is given by*

$$\phi(q) = \begin{cases} [\ (\{q\})\ .\ \mathrm{nil}\ ], & \text{if } q \in S; \\ u, & \text{otherwise.} \end{cases}$$

Then if $\{q_1, \ldots, q_m\} \subseteq S$ and p is an arbitrary sentence,

$$[\ (\{q_1, \ldots, q_m\})\ .\ \mathrm{nil}\ ] \leq_k \mathrm{cl}_B\phi(p)$$

if and only if the q_i's entail p. In addition,

$$[\ \mathrm{nil}\ .\ (\{q_1, \ldots, q_m\})\ ] \leq_k \mathrm{cl}_B\phi(p)$$

if and only if the q_i's entail $\neg p$.

Proof. We prove the theorem by directly exhibiting the closure of ϕ to be

$$\mathrm{cl}_B\phi(p) = \sum_{U \subseteq S} \{[\ (U)\ .\ \mathrm{nil}\ ] | U \models p\} + \sum_{V \subseteq S} \{[\ \mathrm{nil}\ .\ (V)\ ] | V \models \neg p\}. \tag{26}$$

To see this, note first that any apparently closed extension of ϕ will necessarily be an extension of

$$\psi(p) = \bigvee_{U \subseteq S} \{[\ (U)\ .\ \mathrm{nil}\ ] | U \models p\} = \sum_{U \subseteq S} \{[\ (U)\ .\ \mathrm{nil}\ ] | U \models p\},$$

and that the expression in (26) is simply the $\neg$-closure of this expression.

Since the expression in (26) is $\neg$-closed, we must only show that it satisfies the first two clauses of Definition 6.1 of apparent closure.

To see this, suppose first that we have some collection $p_1, \ldots, p_n$ of sentences, and that $U_{i1}, \ldots, U_{im_i}$ are the subsets of S that entail p_i. If we additionally denote by $\{V_j\}$ the set of subsets of S that entail $\wedge_i p_i$, then we must show that

$$\sum_j [\ (V_j)\ .\ \mathrm{nil}\ ] \geq_k \bigwedge_i \sum_j [\ (U_{ij})\ .\ \mathrm{nil}\ ].$$

(There is a similar inequality that must be proven if the *negations* of the p_i are entailed by S, but the proof is similar.)

If we denote by $+$ and $\cdot$ the join and meet operations on the lattice of justifications, this will follow if we can show that

$$\sum_i (V_i) \geq \prod_i \sum_j (U_{ij}).$$

Distributing the product and sum in the right hand side of this expression, we see that the result will follow if we can show that

$$\sum_i (V_i) \geq \prod_k \{(U_k)\} \tag{27}$$

for any set $\{U_k\}$ that contains at least one of the U_{1i}'s, at least one of the U_{2i}'s, and so on. But

$$\prod_k (U_k) = (\cup_k U_k) = V_i$$

for some i, since the union of the U_k justify the conjunction $\wedge_i p_i$. We can conclude (27) from this.

The remaining clause in the definition of apparent closure is similar, essentially requiring us simply to observe that if $p \models q$, then any justification for p will also be a justification for q. □

Theorem 6.16. *Let B be a distributive bilattice. Then if x is any element of B with $x = \neg x$, the subset of B given by*

$$B^x = \{y \in B | y \geq_k x\}$$

is a bilattice. In addition, any extension of a truth assignment with values in B^x will also take values in B^x.

Proof. It is clear that B^x inherits partial orders from B; we must show that it is closed under the lattice operations and under $\neg$.

That B^x is closed under the various lattice operations is immediate: It is closed under $+$ and $\cdot$ because it is a lattice in the k direction, and is closed under $\vee$ and $\wedge$ because B is distributive. Finally, it is closed under $\neg$ because if $z \geq_k x$, then

$$\neg z \geq_k \neg x = x.$$

The final statement in the theorem is obvious. □

Theorem 6.17. *If B is a distributive bilattice such that the k-lattice of B is complemented, there is a natural bilattice homomorphism $i^x : B^x \to B$ such that $i^x(x) = u$ and i^x maps B^x isomorphically onto its image.*

Proof. An element of a bounded distributive lattice can have only one complement [26]; let the k-complement of x be x^*, and set

$$i^x(y) = y \cdot x^*.$$

Since B is distributive, i^x is a bilattice homomorphism; we must show that $i^x(x) = u$ and that i^x is one-to-one.

The first of these is clear, since $i^x(x) = x \cdot x^* = u$ because x^* is a k-complement for x.

For the second, suppose that $y \cdot x^* = z \cdot x^*$. Now if y and z are both in B^x, then $y \geq_k x$ and $z \geq_k x$. It follows that $y \cdot x = x = z \cdot x$, and

$$z \cdot x + z \cdot x^* = y \cdot x + y \cdot x^* = y \cdot (x + x^*) = y \cdot \perp = y = z. \qquad \square$$

Lemma 7.2. *The relation $\lhd$ induces a partial order on the set of truth assignments from L to B.*

Proof. We must show that $\lhd$ is transitive and anti-symmetric. For the first, suppose that $\psi_1 \lhd \psi_2$ and $\psi_2 \lhd \psi_3$; we must show that $\psi_1 \lhd \psi_3$.

If either $\psi_1 <_k \psi_2$ or $\psi_2 <_k \psi_3$, this is immediate. If not, there exist an x with $\psi_1(x) \neq \psi_2(x)$ and

$$\sum\{\psi_1(z) | \psi_1(z) \neq \psi_2(z)\} <_k \psi_2(x).$$

and a y with $\psi_2(y) \neq \psi_3(y)$ and

$$\sum\{\psi_2(z) | \psi_2(z) \neq \psi_3(z)\} <_k \psi_3(y).$$

Now if $\psi_3(x) = \psi_2(x)$, then it is clear that $\psi_1 \lhd \psi_3$, so suppose otherwise. Now we claim that $\psi_3(y) \neq \psi_1(y)$, and that

$$\sum\{\psi_1(z) | \psi_1(z) \neq \psi_3(z)\} <_k \psi_3(y).$$

For the first claim, suppose that $\psi_3(y) = \psi_1(y)$. Then $\psi_1(y) \neq \psi_2(y)$, so that

$$\psi_3(y) = \psi_1(y) \leq_k \sum\{\psi_1(z) | \psi_1(z) \neq \psi_2(z)\} <_k \psi_2(x).$$

But we also have that

$$\psi_2(x) \leq_k \sum\{\psi_2(z) | \psi_2(z) \neq \psi_3(z)\} <_k \psi_3(y); \tag{28}$$

this contradiction proves that $\psi_3(y) \neq \psi_1(y)$.

For the second claim, we have

$$\begin{aligned}
\sum\{\psi_1(z) | \psi_1(z) \neq \psi_3(z)\} &= \sum\{\psi_1(z) | \psi_2(z) = \psi_1(z) \neq \psi_3(z)\} \\
&\quad + \sum\{\psi_1(z) | \psi_2(z) \neq \psi_1(z) \neq \psi_3(z)\} \\
&\leq_k \sum\{\psi_2(z) | \psi_2(z) \neq \psi_3(z)\} + \sum\{\psi_1(z) | \psi_2(z) \neq \psi_1(z)\} \\
&<_k \psi_3(y) + \psi_2(x) \\
&\leq_k \psi_3(y) + \psi_3(y) = \psi_3(y),
\end{aligned}$$

where we have used (28) again. Thus both claims are valid, and $\psi_1 \lhd \psi_3$.

To see that $\lhd$ is anti-symmetric, we must show that we cannot have both $\psi \lhd \phi$ and $\phi \lhd \psi$. It is clear that if this is to happen, one of the truth assignments, say ϕ, must be a proper extension of the other. But now if $\phi \lhd \psi$, then there would be an x with $\phi(x) \neq \psi(x)$ and

$$\psi(x) >_k \sum\{\phi(z) | \phi(z) \neq \psi(z)\},$$

and specifically $\psi(x) >_k \phi(x)$. But this contradicts $\psi <_k \phi$. $\square$

Lemma 7.4. *For any ϕ, $\mathrm{cl}(\phi) \geq_k \mathrm{cl}_B(\phi)$.*

Proof. The product in (11) appearing in the definition of closure is taken over a subset of the set over which the product is taken in (4). $\square$

Theorem 7.5. *For any truth assignment ϕ,* $\mathrm{cl}(\phi) \geq_k \phi$ *and* $\mathrm{cl}(\mathrm{cl}(\phi)) = \mathrm{cl}(\phi)$.

Proof. As for the corresponding result involving cl_B, the first of the two statements is clear.

For the second, we must show that if ψ is a $\triangleleft$-minimal apparently closed extension of $\mathrm{cl}(\phi)$, then it is also a $\triangleleft$-minimal apparently closed extension of ϕ. The set over which the product is taken to calculate $\mathrm{cl}(\mathrm{cl}(\phi))$ would then be a superset of that over which it is taken in $\mathrm{cl}(\phi)$, so that $\mathrm{cl}(\mathrm{cl}(\phi)) \leq_k \mathrm{cl}(\phi)$. But $\mathrm{cl}(\mathrm{cl}(\phi)) \geq_k \mathrm{cl}(\phi)$, so the proof would be complete.

Suppose then that this were not so, so that there were some apparently closed ψ that was $\triangleleft$-minimal for $\mathrm{cl}(\phi)$ but not for ϕ. It follows that there would be some apparently closed extension χ of ϕ with $\chi \triangleleft \psi$; since $\triangleleft$ is transitive, we can assume without loss of generality that χ is $\triangleleft$-minimal. But now $\chi \geq_k \mathrm{cl}(\phi)$, so ψ would not have been a $\triangleleft$-minimal apparently closed extension of $\mathrm{cl}(\phi)$. □

Theorem 7.6. *Closure and balanced closure are identical for balanced bilattices.*

Proof. Recall that $\mathrm{cl}_B(\phi)$ is apparently closed:

$$\mathrm{cl}_B(\phi) = \prod\{\psi|\psi \in A(\phi)\} \in A(\phi).$$

We also have that $\mathrm{cl}_B(\phi) \leq_k \psi$ for every $\psi \in A(\phi)$ and therefore that $\psi \not\triangleleft \mathrm{cl}_B(\phi)$ for every $\psi \in A(\phi)$. It follows that $\mathrm{cl}_B(\phi)$ is an element of the set over which the product is taken to obtain $\mathrm{cl}(\phi)$, so that $\mathrm{cl}(\phi) \leq_k \mathrm{cl}_B(\phi)$. Combining this with Lemma 7.4 gives us the desired result. □

Corollary 7.7. *The closure operator is k-monotonic on balanced bilattices.*

Proof. This follows immediately from Corollary 6.12. □

Theorem 7.8. *Let (R,T) be a supernormal default theory where T is consistent. Associate with (R,T) a truth assignment ϕ given by*

$$\phi(p) = \begin{cases} t, & \text{if } p \in T; \\ dt, & \text{if } p \text{ is } r_i \text{ for some } \frac{:r_i}{r_i} \in R \text{ but } p \notin T; \\ u, & \text{otherwise.} \end{cases}$$

Then for any i,

$$\mathrm{cl}(\phi)(r_i) = \begin{cases} t, & \text{iff } T \models r_i; \\ f, & \text{iff } T \models \neg r_i; \\ *, & \text{iff } r_i \text{ is true in some extensions of } (R,T) \text{ and false in others;} \\ dt, & \text{iff } T \not\models r_i \text{ but } r_i \text{ is true in all extensions of } (R,T). \end{cases}$$

Proof. We take closure on the default bilattice D in two stages. First, we take $\psi(p) = t$ if $T \models p$, and $\psi(p) = f$ if $T \models \neg p$; it is clear that any apparently closed extension of ϕ will be an extension of ψ.

We now take $\psi(r_i) = dt$ for any default fact r_i that is logically independent of the facts in T. We can now restrict ψ to the subbilattice consisting of the four points $\{dt, df, u, *\}$, and take the closure on this subbilattice, restricting our language to the set of sentences that are independent of T. The result, when combined with the truth values assigned to consequences of T, will clearly be a k-minimal apparently closed extension of ϕ. It will be the unique $\triangleleft$-minimal apparently closed extension because any other apparently closed extension will involve assigning a truth value outside of $\{dt, df, u, *\}$ to some sentence unnecessarily.[30]

It remains to see that the ψ we have constructed satisfies the conclusion of the theorem. It is clear that $\psi(r_i) \geq_k dt$ for all i, and also that we will have $\psi(r_i) = *$ if and only if $\neg r_i$ is implied by some other fact whose truth value is dt or $*$. But this means that there must be some subset of the r_j's that imply $\neg r_i$. The theorem will therefore follow from the following lemma:

Lemma A.1 *Let (R,T) be a supernormal default theory, and let S be the set of default facts r_i. Then there is a natural correspondence between extensions of (R,T) and maximal consistent subsets of S (by consistent here, we mean consistent with T).*

[30] The essential idea here is that we work our way "down" from the top of the bilattice, considering first the "certain" values t, f and $\perp$, and only then the default values dt, df and $*$. A proof in greater detail can be found in the proof of Theorem 11.10.

Proof. For any maximal consistent $U \subseteq S$, let U' denote the deductive closure of $U \cup T$. Since the U's are maximal, it is clear that the closures of these sets will be distinct. We will show that every such U' is an extension of (R, T), and that all such extensions arise in this fashion.

It is clear that $T \subseteq U'$, so the first of Reiter's three conditions is satisfied. The second is immediate, since U' is the deductive closure of some possibly smaller set.

For the third condition, if some $q \in S$ is not in U, then its negation must be in U', since U was maximal. Contraposing this observation gives us the desired conclusion.

This shows that U' is an extension of (R, T); it remains to show that all such extensions are of this form. To see this, take any such extension E and consider $E \cap S$. Condition (3) of Reiter's definition guarantees that $E \cap S$ will be a maximal consistent subset of S. $\square$

Theorem 7.9. *Suppose we fix some set of sentences T, together with two default sentences p_1 and p_2, and that ϕ is given by:*

$$\phi(p) = \begin{cases} t, & \text{if } p \in T; \\ dt_1, & \text{if } p = p_1; \\ dt_2, & \text{if } p = p_2; \\ u, & \text{otherwise.} \end{cases}$$

Then for any sentence p,

$$\mathrm{cl}(\phi)(p) = \begin{cases} t, & \text{iff } T \models p; \\ f, & \text{iff } T \models \neg p; \\ dt_1 & \text{iff } p \text{ is independent of } T \text{ but } T \cup \{p_1\} \models p; \\ df_1 & \text{iff } p \text{ is independent of } T \text{ but } T \cup \{p_1\} \models \neg p; \\ dt_2 & \text{iff } p \text{ is independent of } T \cup \{p_1\} \text{ but } T \cup \{p_1, p_2\} \models p; \\ df_2 & \text{iff } p \text{ is independent of } T \cup \{p_1\} \text{ but } T \cup \{p_1, p_2\} \models \neg p; \\ u, & \text{otherwise.} \end{cases}$$

Proof. Similar to the proof of Theorem 7.8. $\square$

Lemma 7.10. *If p and q are logically equivalent, $\phi(p) = \phi(q)$ for any closed truth assignment ϕ.*

Proof. In any apparently closed extension of ϕ, we have

$$\phi(p) \geq_t \phi(q) \geq_t \phi(p). \qquad \square$$

Theorem 7.11. *Let ϕ be a closed truth assignment, and suppose that $\phi(q) = t$ and $\phi(r) = f$. Then*

$$\begin{aligned} \phi(p \wedge q) &= \phi(p) \\ \phi(p \vee r) &= \phi(p) \\ \phi(p \to r) &= \neg\phi(p) \\ \phi(q \to p) &= \phi(p) \end{aligned}$$

for all p.

Proof. We prove the first claim for any apparently closed truth assignment; the theorem then follows easily.

Suppose first that $\phi[\neg(p \wedge q) \wedge q] = \phi[\neg(p \wedge q)]$. Now since $p \wedge q \models p$, we certainly have $\phi(p \wedge q) \leq_t \phi(p)$, and need therefore only show that $\phi(p \wedge q) <_t \phi(p)$ is impossible. But if $\phi(p \wedge q) <_t \phi(p)$, we would have $\phi[\neg(p \wedge q)] >_t \neg\phi(p)$, so that (in light of our assumption), $\phi[\neg(p \wedge q) \wedge q] >_t \phi(\neg p)$. But $\neg(p \wedge q) \wedge q \models \neg p$, so this is impossible.

It follows that if the theorem is to fail, we must have $\phi[\neg(p \wedge q) \wedge q] \neq \phi[\neg(p \wedge q)]$. But $\phi[\neg(p \wedge q) \wedge q] \geq_k \phi[\neg(p \wedge q)] \wedge \phi(q) = \phi[\neg(p \wedge q)]$, so we must have

$$\phi[\neg(p \wedge q) \wedge q] >_k \phi[\neg(p \wedge q)]. \tag{29}$$

In addition, since we have $\phi[\neg(p \wedge q) \wedge q] \neq \phi[\neg(p \wedge q)]$, the theorem fails for $\neg(p \wedge q)$. It follows that we must have

$$\phi(\neg[\neg(p \wedge q) \wedge q] \wedge q) >_k \phi(\neg[\neg(p \wedge q) \wedge q]).$$

But $\neg[\neg(p \wedge q) \wedge q] \wedge q \equiv [(p \wedge q) \vee \neg q] \wedge q \equiv p \wedge q$, so this becomes

$$\phi(p \wedge q) >_k \phi(\neg[\neg(p \wedge q) \wedge q]),$$

or

$$\phi[\neg(p \wedge q)] >_k \phi[\neg(p \wedge q) \wedge q].$$

This is in contradiction with (29), and the proof is complete. $\square$

Corollary 7.12 (Modus Ponens). *Let ϕ be a closed truth assignment. Then for any p and q,*

$$\phi(q) \geq_t \phi[p \wedge (p \rightarrow q)].$$

In addition, if $\phi(p \rightarrow q) = t$, then $\phi(q) \geq_t \phi(p)$.

Proof. The first claim follows from the fact that $p \wedge (p \rightarrow q) \models q$; the second from the fact that $\phi[p \wedge (p \rightarrow q)] = \phi(p)$ if $\phi(p \rightarrow q) = t$. $\square$

Lemma 7.13. *If ϕ is a closed truth assignment, $\phi(p \wedge q) \geq_k \phi(p) \wedge \phi(q)$. In addition, $\phi(p \vee q) \geq_k \phi(p) \vee \phi(q)$.*

Proof. The first inequality is immediate. For the second, we have:

$$\neg\phi(p \vee q) = \phi[\neg(p \vee q)] = \phi(\neg p \wedge \neg q) \geq_k \phi(\neg p) \wedge \phi(\neg q) = \neg\phi(p) \wedge \neg\phi(q) = \neg[\phi(p) \vee \phi(q)],$$

so that

$$\phi(p \vee q) \geq_k \phi(p) \vee \phi(q). \qquad \square$$

Theorem 7.14. *Let ϕ be a closed truth assignment, and for any sentence p containing a free variable x, denote by p_x^t the result of substituting the term t for x. Then for any p and q:*

$$\begin{array}{rcl}
\phi(p \wedge q) & \leq_t & \phi(p) \wedge \phi(q) \\
\phi(p \vee q) & \geq_t & \phi(p) \vee \phi(q) \\
\phi(p \rightarrow q) & \geq_t & \neg\phi(p) \vee \phi(q) \\
\phi(\forall x.p) & \leq_t & \mathrm{glb}_t\{\phi(p_x^t) | t \text{ is substitutable for } x \text{ in } p\} \\
\phi(\exists x.p) & \geq_t & \mathrm{lub}_t\{\phi(p_x^t) | t \text{ is substitutable for } x \text{ in } p\}
\end{array}$$

Proof. Since $p \wedge q \models p$, we have $\phi(p \wedge q) \leq_t \phi(p)$, etc. $\square$

Theorem 7.16 (Resolution). *Let ϕ be a closed truth assignment. Then*

$$\begin{array}{rcl}
\phi[(a \wedge d) \rightarrow (b \vee e)] & \geq_t & \phi\{[a \rightarrow (b \vee c)] \wedge [(c \wedge d) \rightarrow e]\} \\
& \geq_k & \phi[a \rightarrow (b \vee c)] \wedge \phi[(c \wedge d) \rightarrow e].
\end{array}$$

Proof. $[a \rightarrow (b \vee c)] \wedge [(c \wedge d) \rightarrow e] \models [(a \wedge d) \rightarrow (b \vee e)]$. $\square$

Theorem 9.1. *Let B be a nontrivial bilattice, and ϕ a truth assignment defined on B. Then if $p \in L$ is a sentence in our language, determining the value of $\mathrm{cl}(\phi)(p)$ is in general at best semi-decidable.*

Proof. The theorem holds for the bilattice corresponding to first-order logic. For any collection $S \subseteq L$ of sentences in our language, we define ϕ_S to be

$$\phi_S(q) = \begin{cases} t, & \text{if } q \in S; \\ u, & \text{otherwise.} \end{cases}$$

We know from Theorem 6.13 that the closure of ϕ_S is given by:

$$\mathrm{cl}(\phi)_S(p) = \begin{cases} \bot, & \text{if } S \text{ is inconsistent;} \\ t, & \text{if } S \text{ is consistent and } S \models p; \\ f, & \text{if } S \text{ is consistent and } S \models \neg p; \\ u, & \text{otherwise.} \end{cases}$$

Since determining whether or not $S \models p$ is semi-decidable, it follows that determining $\mathrm{cl}(\phi)(p)$ is at best semi-decidable. □

Lemma 9.4. *A point x of a bilattice is t-grounded if and only if $\neg x$ is f-grounded.*

Proof. To see that $\neg x$ is f-grounded, suppose that $y \leq_t \neg x$, so that $\neg y \geq_t x$. Now it follows that $\neg y \geq_k x$, or $y \geq_k \neg x$. The reverse implication is similar. □

Lemma 9.5. *In a respectful bilattice, if x is t-grounded, then $x \geq_t u$ and $x \leq_k t$. If x if f-grounded, then $x \leq_t u$ and $x \leq_k f$.*

Proof. Since $x \leq_k x$ and $u \leq_k x$, it follows that $x \vee u \leq_k x$. But since x is t-grounded and $x \vee u \geq_t x$, $x \vee u \geq_k x$. Therefore $x \vee u = x$ and $u \leq_t x$. In addition, since $t \geq_t x$, $t \geq_k x$. The result for f-groundedness is similar. □

Lemma 9.6. *If B is a respectful bilattice and x and y are t-grounded points of B, then so is $x \vee y$, and $x + y = x \vee y$. If x and y are f-grounded, then so is $x \wedge y$, and $x + y = x \wedge y$.*

The proof uses the following useful result:

Lemma A.2 *A bilattice B is k-respectful if and only if $x + y \geq_k x \wedge y \geq_k x \cdot y$ for all x and y in B. It is t-respectful if and only if $x \cdot y \geq_t x \wedge y$ and $x + y \geq_t x \wedge y$ for all $x, y \in B$.*

Proof. Suppose B is k-respectful. Then since $x \geq_k x \cdot y$ and $y \geq_k x \cdot y$, $x \wedge y \geq_k x \cdot y$.

Alternatively, suppose that the condition of the lemma is satisfied. Now if $x \geq_k z$ and $y \geq_k z$, then $x \cdot y \geq_k z$, so $x \wedge y \geq_k x \cdot y \geq_k z$. In addition, since $\neg x \wedge \neg y \geq_k \neg x \cdot \neg y$, we have that $x \vee y = \neg(\neg x \wedge \neg y) \geq_k \neg(\neg x \cdot \neg y) = x \cdot y \geq_k z$. The other conditions of respectfulness are similar. □

We now return to the proof of Lemma 9.6:

Proof. To see that $x \vee y$ is t-grounded if both x and y are, suppose that $z \geq_t x \vee y$. Now $z \geq_t x$ and $z \geq_t y$, so that $z \geq_k x$ and $z \geq_k y$. Thus $z \geq_k x \vee y$, since the bilattice in question is respectful.

To see that $x + y = x \vee y$, note that since $x \vee y \geq_t x$, $x \vee y \geq_k x$; $x \vee y \geq_k y$ similarly. It follows that $x \vee y \geq_k x + y$. But $x \vee y \leq_k x + y$ by Lemma A.2. Thus $x + y = x \vee y$. □

Corollary 9.7. *The sum of t-grounded points is also t-grounded. The sum of f-grounded points is f-grounded.*

Proof. This follows immediately from Lemma 9.6. □

Theorem 9.9. *Let $x \in B$, where B is grounded. Then x can be written as $x_t + x_f$, where x_t is t-grounded and x_f is f-grounded.*

Proof. This follows immediately from Corollary 9.7. □

Lemma 9.11. *For any x, $g_f(x) = \neg g_t(\neg x)$.*

Proof. Obvious. □

Lemma 9.12. *For any x, $g_t(x)$ is t-grounded, and $g_f(x)$ is f-grounded.*

Proof. Note first that $g_t(x) \geq_t x$ in a respectful bilattice. Now if $y \geq_t g_t(x)$, then $y \geq_t x$, so y is an element of the set over which the product is taken in computing $g_t(x)$. Thus $y \geq_k g_t(x)$. □

Lemma 9.14. *In a monotonic bilattice, if $x \geq_k y$, then $g_f(x) \geq_k g_f(y)$.*

Proof. Since $x \geq_k y$, $\neg x \geq_k \neg y$ and $g_t(\neg x) \geq_k g_t(\neg y)$. □

Theorem 9.15. *For any two points x and y in a respectful bilattice B,*

$$g_t(x) + g_t(y) = g_t(x \vee y),$$

and

$$g_f(x) + g_f(y) = g_f(x \wedge y).$$

Proof. Since $g_t(x) \geq_t x$ and $g_t(y) \geq_t y$, it follows that $g_t(x) \vee g_t(y) \geq_t x \vee y$. But $g_t(x)$ and $g_t(y)$ are both t-grounded, so $g_t(x) \vee g_t(y) = g_t(x) + g_t(y) \geq_t x \vee y$. Therefore $g_t(x) + g_t(y)$ is one of the points over which the product is taken in evaluating $g_t(x \vee y)$, and we have

$$g_t(x) + g_t(y) \geq_k g_t(x \vee y). \tag{30}$$

Next, note that if $a \geq_t b$, then $g_t(a) \geq_k g_t(b)$, since the product involved in $g_t(a)$ is taken over a subset of that in $g_t(b)$. Now since $x \vee y \geq_t x$, we conclude that $g_t(x \vee y) \geq_k g_t(x)$. Similarly, $g_t(x \vee y) \geq_k g_t(y)$, so that

$$g_t(x \vee y) \geq_k g_t(x) + g_t(y).$$

It follows from this and (30) that $g_t(x \vee y) = g_t(x) + g_t(y)$. □

Lemma 9.16. *For any x, $g_t(x) + g_f(x) \leq_k x$.*

Proof. $g_t(x) \leq_k x$ and $g_f(x) \leq_k x$. □

Lemma 9.18. *If B is a canonically grounded bilattice and x and y are t-grounded points of B, then so is $x \wedge y$, and $x \cdot y = x \wedge y$. If x and y are f-grounded, then so is $x \vee y$, and $x \cdot y = x \vee y$.*

Proof. Since x and y are t-grounded, we have $u \leq_t x$ and $u \leq_t y$, so that $u \leq_t (x \wedge y)$ and $g_f(x \wedge y) = u$. It follows that

$$x \wedge y = g_t(x \wedge y) + g_f(x \wedge y) = g_t(x \wedge y),$$

so that $x \wedge y$ is t-grounded.

Now, since $x \geq_t x \wedge y$, we have that $x \geq_k x \wedge y$, so $x \cdot y \geq_k x \wedge y$. But $x \wedge y \geq_k x \cdot y$ since B is respectful, so $x \cdot y = x \wedge y$. □

Lemma 9.19. *Let B be a canonically grounded bilattice. Then for any $x \in B$, $g_t(x) = x \vee u$ and $g_f(x) = x \wedge u$.*

Proof. We know that $g_f(x \vee u) = u$, since $u \leq_t x \vee u$. Thus $g_t(x \vee u) = x \vee u$, so that $x \vee u$ is t-grounded.

Now $g_t(x) \leq_k x \vee u$, since $x \vee u \geq_t x$. But also $g_t(x) \geq_t x$ and $g_t(x) \geq_t u$, so that $g_t(x) \geq_t x \vee u$. But now since $x \vee u$ is t-grounded, we have $g_t(x) \geq_k x \vee u$. Therefore $g_t(x) = x \vee u$. □

Theorem 9.20. *A canonically grounded bilattice is monotonic if and only if, for any t-grounded a and c and f-grounded b and d,*

$$a + b = c + d$$

implies $a = b$ and $c = d$.

Proof. Suppose first that B is monotonic. It suffices to show that, if a is t-grounded and b is f-grounded, then

$$a = g_t(a + b)$$

and

$$b = g_f(a + b),$$

since we would then have that

$$a = g_t(a + b) = g_t(c + d) = c,$$

and similarly for b and d.

Since $a \geq_t u \geq_t b$, $a \geq_t a + b$, and it follows that $a \geq_k g_t(a + b)$. On the other hand, $a + b \geq_k a$, so $g_t(a + b) \geq_k g_t(a) = a$. Thus $a = g_t(a + b)$; the identity for b is similar.

For the converse, suppose that $y \geq_k x$. Now $y \geq_k x \geq_k g_t(x)$, so $g_t(x) \leq_k y = g_t(y) + g_f(y)$. Thus

$$g_t(x) + g_t(y) + g_f(y) = g_t(y) + g_f(y) = y.$$

Since $g_t(x) + g_t(y)$ is t-grounded, it follows that $g_t(x) + g_t(y) = g_t(y)$, or $g_t(y) \geq_k g_t(x)$. □

Theorem 9.21. *Let B be a respectful bilattice. Then the following conditions are equivalent:*

1. *B is distributive.*
2. *B is canonically grounded and monotonic.*
3. *B is a subbilattice of a world-based bilattice.*

Proof. It is clear that (3) implies (1). We show that (1) and (2) are equivalent, and that they together imply (3).

To see that distributive bilattices are canonically grounded and monotonic, let B be a distributive bilattice. Then for $x \in B$,

$$g_t(x) = \prod_{y \in B} x \vee y = x \vee \prod_{y \in B} y = x \vee u.$$

Thus

$$\begin{aligned} g_t(x) + g_f(x) &= (x \vee u) + (x \wedge u) \\ &= [x + (x \wedge u)] \vee [u + (x \wedge u)] \\ &= x \vee (x \wedge u) \\ &= x. \end{aligned}$$

To see that B is monotonic, note that if $y \geq_k x$, then

$$g_t(y) = y \vee u \geq_k x \vee u = g_t(x).$$

Showing that a canonically grounded and monotonic bilattice is distributive is rather more involved. We need the following lemmas:

Lemma A.3 *In a canonically grounded monotonic bilattice, if x is t-grounded and y is f-grounded, then $x \cdot y = u$.*

Proof. Since $y \geq_k x \cdot y$, $u = g_t(y) \geq_k g_t(x \cdot y)$. Thus $g_t(x \cdot y) = u = g_f(x \cdot y)$, and $x \cdot y = u$. □

Lemma A.4 *Let B be canonically grounded and monotonic. Then for any $x, y \in B$:*

$$\begin{aligned} g_t(x \wedge y) &= g_t(x) \wedge g_t(y) \\ g_t(x \vee y) &= g_t(x) \vee g_t(y) \\ g_t(x + y) &= g_t(x) + g_t(y) \\ g_t(x \cdot y) &= g_t(x) \cdot g_t(y), \end{aligned}$$

and

$$\begin{aligned} g_f(x \wedge y) &= g_f(x) \wedge g_f(y) \\ g_f(x \vee y) &= g_f(x) \vee g_f(y) \\ g_f(x + y) &= g_f(x) + g_f(y) \\ g_f(x \cdot y) &= g_f(x) \cdot g_f(y). \end{aligned}$$

Proof. We prove the results for g_t only. We have already shown the first in Theorem 9.15. For the second, we note first that since $g_t(x) \wedge g_t(y) = g_t(x) \cdot g_t(y)$, and $x \geq_k g_t(x) \geq_k g_t(x) \cdot g_t(y)$, we can conclude that:

$$x \wedge y \geq_k g_t(x) \cdot g_t(y) = g_t(x) \wedge g_t(y).$$

Next, we have that $x \wedge y \geq_k g_f(x \wedge y)$, so that

$$x \wedge y \geq_k [g_t(x) \wedge g_t(y)] + g_f(x \wedge y).$$

But $x \geq_t x \wedge y$, so $g_t(x) \geq_k g_t(x \wedge y)$ and therefore $g_t(x) \wedge g_t(y) \geq_k g_t(x \wedge y)$. Since $x \wedge y = g_t(x \wedge y) + g_f(x \wedge y)$, this allows us to conclude that

$$x \wedge y \leq_k [g_t(x) \wedge g_t(y)] + g_f(x \wedge y).$$

It follows from this and the previous equation that

$$\begin{aligned} x \wedge y &= g_t(x \wedge y) + g_f(x \wedge y) \\ &= [g_t(x) \wedge g_t(y)] + g_f(x \wedge y). \end{aligned}$$

It now follows from Theorem 9.20 that $g_t(x \wedge y) = g_t(x) \wedge g_t(y)$.

For the third assertion, note that

$$x + y = g_t(x + y) + g_f(x + y)$$

and

$$\begin{aligned} x + y &= g_t(x) + g_f(x) + g_t(y) + g_t(y) \\ &= [g_t(x) + g_t(y)] + [g_f(x) + g_f(y)]. \end{aligned}$$

We can now apply Theorem 9.20 again to conclude that $g_t(x + y) = g_t(x) + g_t(y)$. Similarly, we conclude that $g_t(x \cdot y) = g_t(x) \cdot g_t(y)$ from

$$x \cdot y = g_t(x \cdot y) + g_f(x \cdot y)$$

and

$$\begin{aligned} x \cdot y &= [g_t(x) + g_f(x)] \cdot [g_t(y) + g_t(y)] \\ &= [g_t(x) \cdot g_t(y)] + [g_f(x) \cdot g_f(y)], \end{aligned}$$

where we have used the k-distributivity of the bilattice and the previous lemma. □

Proof of Theorem 9.21 (continued). We must show that in a canonically grounded and monotonic bilattice, $(x + y) \wedge z = (x \wedge z) + (y \wedge z)$, etc.

It clearly suffices to show that $g_t[(x + y) \wedge z] = g_t[(x \wedge z) + (y \wedge z)]$, and similarly for g_f. But

$$\begin{aligned} g_t[(x + y) \wedge z] &= g_t(x + y) \wedge g_t(z) \\ &= [g_t(x) + g_t(y)] \cdot g_t(z) \\ &= [g_t(x) \cdot g_t(z)] + [g_t(y) \cdot g_t(z)] \\ &= [g_t(x) \wedge g_t(z)] + [g_t(y) \wedge g_t(z)] \\ &= g_t(x \wedge z) + g_t(y \wedge z), \end{aligned}$$

and so on. This completes the proof that canonically grounded and monotonic bilattices are distributive.

To see that a distributive bilattice B is necessarily a subbilattice of a world-based bilattice, note that since B is canonically grounded and monotonic, it follows from earlier results that B is generated by its t-grounded elements, together with their negations (the f-grounded elements).

The collection of t-grounded elements itself forms a complete distributive lattice, and is therefore a sublattice of the lattice of subsets of some set W. Now it is not hard to see that B is a subbilattice of the world-based bilattice B_W. □

Theorem 10.1. *Let B be a canonically grounded bilattice. Then the closure operation is k-monotonic on B if and only if B is distributive.*

Proof. We know from Corollary 6.12 that closure is k-monotonic on balanced bilattices. For the converse, it suffices to show that if closure is k-monotonic, then B is monotonic.

To do this, fix two logically independent sentences p and q in L, and suppose we denote by ϕ_x the truth assignment that takes truth value x at p and u elsewhere. Then it is not hard to see that $\mathrm{cl}(\phi_x)(p \vee q) = g_t(x)$. Since $\phi_y \geq_k \phi_x$ if $y \geq_k x$, it follows that $g_t(y) \geq_k g_t(x)$ if closure is k-monotonic on B. □

Theorem 10.2. *Let B be a distributive bilattice, and suppose that ϕ is some $\neg$-closed truth assignment on B. Then the closure of ϕ is given by:*

$$\mathrm{cl}(\phi)(p) = \sum_{U \in \pi_+(p)} [\phi(U) \vee u] + \sum_{V \in \pi_-(p)} [\neg\phi(V) \wedge u]. \tag{31}$$

Proof. Suppose that we denote the truth assignment appearing in (31) by ψ; we immediately have that:

$$\psi(p) = \sum_{U \in \pi_+(p)} g_t[\phi(U)] + \sum_{V \in \pi_-(p)} \neg g_t[\phi(V)]. \tag{32}$$

Now in order to show that $\psi = \mathrm{cl}(\phi)$, it will suffice to show:

1. That any apparently closed extension of ϕ is an extension of ψ, and
2. That ψ is apparently closed.

For the first, suppose that χ is an apparently closed extension of ϕ. We show that $\chi \geq_k \psi$ by showing that $\chi(p) \geq_k g_t[\phi(U)]$ for any $U \in \pi_+(p)$. For $V \in \pi_-(p)$, the technique is similar.

Fix $U \in \pi_+(p)$, and suppose that $U = \{q_i\}$. Now $\wedge q_i \models p$, so $\chi(p) \geq_t \chi(\wedge q_i)$ and therefore $\chi(p) \geq_k g_t[\chi(\wedge q_i)]$. But $\chi(\wedge q_i) \geq_k \wedge\chi(q_i)$ and, since B is monotonic, it follows that $g_t[\chi(\wedge q_i)] \geq_k g_t[\wedge\chi(q_i)]$. Applying Lemma A.4, we conclude that

$$g_t[\chi(\wedge q_i)] \geq_k \wedge g_t[\chi(q_i)],$$

or that

$$\chi(p) \geq_k \wedge g_t[\chi(q_i)]. \tag{33}$$

Now recall that χ is an extension of ϕ, so $\chi(q_i) \geq_k \phi(q_i)$. Since B is monotonic, it follows that $g_t[\chi(q_i)] \geq_k g_t[\phi(q_i)]$, and therefore that

$$\wedge g_t[\chi(q_i)] \geq_k \wedge g_t[\phi(q_i)] = g_t[\wedge\phi(q_i)] = g_t[\phi(U)],$$

where we have applied Lemma A.4 again. Combining this with (33), we obtain

$$\chi(p) \geq_k g_t[\phi(U)].$$

It remains to show that ψ in (32) is apparently closed. We check the three conditions in the definition of apparent closure separately.

First, to see that ψ is $\neg$-closed, we have

$$\begin{aligned}
\psi(\neg q) &= \sum_{V \in \pi_+(\neg q)} g_t[\phi(V)] + \sum_{U \in \pi_-(\neg q)} \neg g_t[\phi(U)] \\
&= \neg \sum_{U \in \pi_+(q)} g_t[\phi(U)] + \sum_{V \in \pi_-(q)} g_t[\phi(V)] \\
&= \neg\psi(q).
\end{aligned}$$

Next, suppose that $p \models q$. We must show that $\psi(q) \geq_t \psi(p)$.

Since $p \models q$, it follows that $\pi_+(p) \subseteq \pi_+(q)$ and $\pi_-(p) \supseteq \pi_-(q)$. Thus in the expressions (32) for $\psi(p)$ and $\psi(q)$, there will be extra terms in the first sum for $\psi(q)$, and extra terms in the second sum for $\psi(p)$. It follows that

$$\psi(q) = a + b$$

and

$$\psi(p) = c + d,$$

where $a \geq_t c$ and $b \geq_t d$. Since B is distributive, it follows that $a + b \geq_t c + d$.

Finally, we must show that $\psi(p_1 \wedge p_2) \geq_k \psi(p_1) \wedge \psi(p_2)$. For notational convenience, we set

$$a = \sum_{U \in \pi_+(p_1 \wedge p_2)} g_t[\phi(U)] \qquad b = \sum_{V \in \pi_-(p_1 \wedge p_2)} \neg g_t[\phi(V)]$$

$$c_i = \sum_{U \in \pi_+(p_i)} g_t[\phi(U)] \qquad d_i = \sum_{V \in \pi_-(p_i)} \neg g_t[\phi(V)]$$

We are trying to show that

$$\begin{aligned} a + b &\geq_k (c_1 + d_1) \wedge (c_2 + d_2) \\ &= (c_1 \wedge c_2) + (d_1 \wedge c_2) + (d_2 \wedge c_1) + (d_1 \wedge d_2) \\ &= (c_1 \wedge c_2) + d_1 + d_2 + (d_1 \wedge d_2) \\ &= (c_1 \wedge c_2) + (d_1 \wedge d_2). \end{aligned}$$

It suffices to show that $a \geq_k c_1 \wedge c_2$ and $b \geq_k d_1 \wedge d_2$. We can show that $b \geq_k d_i$ by noting that $\pi_-(p_1 \wedge p_2) \supseteq \pi_-(p_i)$. To see that $a \geq_k c_1 \wedge c_2$, we have:

$$\begin{aligned} c_1 \wedge c_2 &= \sum_{U \in \pi_+(p_1)} g_t[\phi(U)] \wedge \sum_{V \in \pi_+(p_2)} g_t[\phi(V)] \\ &= \sum_{\substack{U \in \pi_+(p_1) \\ V \in \pi_+(p_2)}} g_t[\phi(U)] \wedge g_t[\phi(V)] \\ &= \sum_{\substack{U \in \pi_+(p_1) \\ V \in \pi_+(p_2)}} g_t[\phi(U) \wedge \phi(V)] \\ &= \sum_{\substack{U \in \pi_+(p_1) \\ V \in \pi_+(p_2)}} g_t[\phi(U \cup V)] \\ &\leq_k a, \end{aligned}$$

since $U \cup V \models p_1 \wedge p_2$ if $U \models p_1$ and $V \models p_2$.

This completes the proof that ψ in (31) is apparently closed. □

Corollary 10.3. *Suppose that the conditions of Theorem 10.2 are satisfied, and that, in calculating* $\mathrm{cl}(\phi)$, *we have determined that either*

$$\sum_{U \in \pi_+(p)} [\phi(U) \vee u] \geq_k x$$

or

$$\sum_{V \in \pi_-(p)} [\neg\phi(V) \wedge u] \geq_k \neg x.$$

Then any proof of p or $\neg p$ respectively from a set U with $\phi(U) \vee u \leq_t x$ will not contribute to $\mathrm{cl}(\phi)(p)$. *Specifically, no superset of a set underlying a previously discovered proof need be considered, and no proof using any sentence q with $\phi(q) \vee u \leq_t x$ need be considered. A sentence with $\phi(q) \leq_t u$ need never be considered.*

Proof. If $\phi(U) \vee u \leq_t x$, then $\phi(U) \vee u \leq_k x$, since $\phi(U) \vee u$ is t-grounded. That $\neg\phi(V) \wedge u \leq_k \neg x$ is similar.

Next, suppose that $V \supseteq U$, where $x \geq_k \phi(U) \vee u$. Now since $\phi(V) \leq_t \phi(U)$,

$$\phi(V) \vee u \leq_t \phi(U) \vee u,$$

so that

$$\phi(V) \vee u \leq_k \phi(U) \vee u \leq_k x.$$

A similar argument holds if $\phi(q) \vee u \leq_t x$.

Finally, if $\phi(q) \leq_t u$, then $\phi(q) \vee u = u$, and we can apply the corollary before doing any deduction at all. $\square$

Corollary 10.4 (Updates). *Let B be a balanced bilattice, and suppose that ϕ is a closed truth assignment on B. Now if ψ is a $\neg$-closed extension of ϕ that agrees with ϕ except for q, $\neg q$ and their $\neg$-equivalents, then*

$$\mathrm{cl}(\psi)(p) = \phi(p) + \sum_{U \in \pi_+^q(p) \cup \pi_+^{\neg q}(p)} [\phi(U) \vee u] + \sum_{V \in \pi_-^q(p) \cup \pi_-^{\neg q}(p)} [\neg\phi(V) \wedge u].$$

Proof. Evaluating $\mathrm{cl}(\psi)$ and $\mathrm{cl}(\phi)$ using (16), we see that the only differences are in the terms in the corollary. $\square$

Procedure 10.5 (Monotonic backward inference). Given as input a query q:

1. Set $z = g_t[\phi(q)]$.
2. Find a first-order proof of q from a set U of sentences in the database with $\phi(U) \vee u \not\leq_t z$, using any conventional prover. If none exists, stop and return z.
3. Set $z = z + [\phi(U) \vee u]$ and return to step 2.

Theorem 10.6 (Monotonic backward inference). *Suppose that ϕ is a $\neg$-closed truth assignment on a distributive bilattice B and that q is a sentence of L. Then the result returned by Procedure 10.5 is $g_t[\mathrm{cl}(\phi)(q)]$.*

Proof. Note first that in the first step of the procedure, we could in fact have taken $z = u$, if the first "proof" considered for q was simply that depending on q itself. Thus the procedure directly reproduces the sum appearing in Theorem 10.2. Not all proofs of q are considered, but we know from Corollary 10.3 that the simplification made is acceptable. $\square$

Theorem 10.7. *Suppose that $\phi(p) \neq u$ for a finite subset of our language L, and that the conventional prover used in Procedure 10.5 always terminates. Then the procedure itself terminates as well.*

Proof. As a consequence of the requirement in the proof that $\phi(U) \vee u \not\leq_t z$ in the procedure, it follows that:

1. No subset of L is ever considered twice, and
2. Every subset of L considered is a subset of $\{p | \phi(p) \neq u\}$.

We see from the second condition that there are only a finite number of possible subsets that might be considered, and from the first that each of these subsets will only be considered once. It follows that the procedure will invoke the theorem prover only a finite number of times, and will therefore terminate. $\square$

Theorem 11.1. *Let B be a canonically grounded bilattice, and suppose that ψ is some truth assignment on B. Then ψ is apparently closed if and only if it satisfies:*

$$\psi(p) = \sum_{U \in \pi_+(p)} [\psi(U) \vee u] + \sum_{V \in \pi_-(p)} [\neg\psi(V) \wedge u]. \tag{34}$$

Proof. As for Theorem 10.2, we can immediately rewrite (34) as:

$$\psi(p) = \sum_{U \in \pi_+(p)} g_t[\psi(U)] + \sum_{V \in \pi_-(p)} \neg g_t[\psi(V)]. \tag{35}$$

We will denote the expression on the right-hand side of this equation by $F(\psi)$, so that we are trying to show that ψ is apparently closed if and only if $\psi = F(\psi)$.

We first show that if ψ is apparently closed, it is indeed a fixed point of the F operator. Note first that since $p \models p$, $p \in \pi_+(p)$ for any p; $\neg p \in \pi_-(p)$ similarly. Now since $\psi(\{p\}) = \psi(p)$, it follows that

$$\begin{aligned} F(\psi)(p) &\geq_k g_t[\psi(p)] + \neg g_t[\psi(\neg p)] \\ &= g_t[\psi(p)] + g_f[\psi(p)] \\ &= \psi(p). \end{aligned}$$

We therefore need only show that $\psi(p) \geq_k F(\psi)(p)$ if ψ is apparently closed.

It suffices to show that for any $U \in \pi_+(p)$, $\psi(p) \geq_k g_t[\psi(U)]$, and similarly for $V \in \pi_-(p)$. We show the result for U only.

In order to show that $\psi(p) \geq_k g_t[\psi(U)]$, it suffices to show that $\psi(p) \geq_t \psi(U)$. But now fix some $U = \{q_i\}$ in $\pi_+(p)$; we have

$$\begin{aligned} \psi(U) &= \wedge_i \psi(q_i) + \psi(\wedge_i q_i) \\ &= \psi(\wedge_i q_i) \\ &\leq_t \psi(p). \end{aligned}$$

The second equality follows from the fact that ψ is apparently closed, and the inequality from the fact that the q_i collectively entail p.

The converse, that any ψ satisfying (34) is apparently closed, is harder. That ψ is $\neg$-closed is easy and is unchanged from the proof of Theorem 10.2. We next show that if $p \models q$, then $\psi(p) \leq_t \psi(q)$.

Since $p \models q$, it follows that $\pi_+(p) \subseteq \pi_+(q)$ and $\pi_-(p) \supseteq \pi_-(q)$. As in Theorem 10.2, in the expressions (34) for $\psi(p)$ and $\psi(q)$, there will be extra terms in the first sum for $\psi(q)$, and extra terms in the second sum for $\psi(p)$. It follows that

$$\psi(q) = a + b$$

and

$$\psi(p) = c + d,$$

where $a \geq_t c$ and $b \geq_t d$, and also a and c are t-grounded and b and d are f-grounded. We show from this that $a + b \geq_t c + d$.[31]

In fact, we show only that $a + b \geq_t c + b$. There is a similar proof that $c + b \geq_t c + d$, and the conclusion will then follow.

To see that $a + b \geq_t c + b$, note that since $a \geq_k c$, $a + b \geq_k b + c$, and we have

$$a + b \geq_k (a + b) \vee (b + c) \geq_k b + c \geq_k b \geq_k g_f(a + b), \tag{36}$$

where the last inequality follows from the fact that $b \leq_t a + b$, since $b \leq_t u \leq_t a$.

We also have $(a + b) \vee (b + c) \geq_t a + b$, so that $(a + b) \vee (b + c) \geq_k g_t(a + b)$. Combining this with (36), we get

$$(a + b) \vee (b + c) \geq_k g_t(a + b) + g_f(a + b) = a + b.$$

In light of the first inequality of (36), we therefore have that

$$(a + b) \vee (b + c) = a + b,$$

so that $a + b \geq_t b + c$. It follows that if $p \models q$, then $\psi(q) \geq_t \psi(p)$.

Finally, we must show that if ψ satisfies (34), then for any collection of p_i, $\psi(\wedge p_i) \geq_k \wedge \psi(p_i)$. But since $\{p_i\} \models \wedge p_i$, we know that

$$\psi(\wedge p_i) \geq_k g_t[\wedge \psi(p_i) + \psi(\wedge p_i)].$$

[31] This is more difficult than in the distributive case, since we can no longer pass the t-inequalities through the summation in the expressions for $\psi(p)$ and $\psi(q)$.

If we denote $\psi(\wedge p_i)$ by x and $\wedge\psi(p_i)$ by y, then we know that

$$x \geq_k g_t(x+y), \tag{37}$$

and are trying to show that $x \geq_k y$.

But now note that since $\wedge p_i \models p_i$ for each i, we know from the previous part of the proof that $\psi(\wedge p_i) \leq_t \psi(p_i)$ for each i, so that $\psi(\wedge p_i) \leq_t \wedge\psi(p_i)$. In other words, $x \leq_t y$ in (37).

Since $x \leq_t y$, it follows that $x \leq_t x+y$, and therefore that $g_f(x+y) \leq_k x$. But we already know that $g_t(x+y) \leq_k x$ from (37), so

$$x+y = g_t(x+y) + g_f(x+y) \leq_k x.$$

Thus $x = x+y$, and $x \geq_k y$. This completes the proof. □

Lemma 11.3. *If B splits at x, then $x = \neg x$.*

Proof. Since $x \in B^x$, so is $\neg x$, and $x \cdot \neg x \in B^x$. Thus $x \cdot \neg x \not< x$, and $x \leq_k \neg x$. But this implies $\neg x \leq_k \neg\neg x = x$, so $x = \neg x$. □

Lemma 11.4. *If B splits at x, then $y \geq_k x$ for every $y \in B^x$.*

Proof. Fix $y \in B^x$. Now $y \cdot x \leq_k x$ is also an element of B^x, so $y \cdot x = x$, and $y \geq_k x$. □

Lemma 11.5. *Suppose that B is canonically grounded and splits at x. Now fix $z \in B^x$ with $z \neq x$. If $z \geq_t y$ for some $y \leq_k x$, then $z \geq_t y$ for* every *$y \leq_k x$.*

Proof. Suppose we have that $z \geq_t y$, and that $y' \leq_k x$. Now since $z \geq_t y$, $g_f(z) \leq_k y \leq_k x$. Now we cannot have $g_t(z) \leq_k x$, since this would imply $z = g_t(z) + g_f(z) \leq_k x$. Thus $g_t(z) \in B^x$, and $g_t(z) \geq_k x$ by the preceding lemma. Thus $g_t(z) \geq_k g_f(z)$ and $z = g_t(z) + g_f(z) = g_t(z)$, so that z is t-grounded.

But now note that $z \geq_k y'$, so that $z \geq_k z \vee y'$. We also have that $z \vee y' \geq_t z$, so that $z \vee y' \geq_k z$, since z is t-grounded. It follows that $z = z \vee y'$, so that $z \geq_t y'$. □

Theorem 11.6. *Suppose that B is canonically grounded and that B splits at x. Then the natural mapping $\pi : B \to B^x$ given by*

$$\pi(y) = \begin{cases} x, & \text{if } y \leq_k x; \\ y, & \text{otherwise.} \end{cases}$$

is a bilattice homomorphism.

Proof. We must show that for any $y, z \in B$, $\pi(y \vee z) = \pi(y) \vee \pi(z)$, and similarly for $\wedge$, $\cdot$ and $+$. The proofs are all similar, with the proofs for $\vee$ and $\wedge$ being somewhat more involved. We present the proof for $\vee$ only.

If $y, z \in B^x$, then $y \vee z \in B^x$ because B^x is closed under the bilattice operations. In this case $\pi(y) = y$, $\pi(z) = z$ and $\pi(y \vee z) = y \vee z$.

If, on the other hand, $y \leq_k x$ and $z \leq_k x$, then $y \vee z \leq_k x$ and $x = \pi(y \vee z) = x \vee x = \pi(y) \vee \pi(z)$. In the remaining case, we can assume without loss of generality that $y \leq_k x$ and $z >_k x$, so that $\pi(y) = x$ and $\pi(z) = z$. There are still two cases:

First, suppose that $y \vee z \leq_k x$, so that $\pi(y \vee z) = x$. We must show that $x \vee z = x$, or that $x \geq_t z$. Now we know that $z \leq_t y \vee z$ and $y \vee z \leq_k x$, so $z \leq_t y'$ for any $y' \leq_k x$, and specifically $z \leq_t x$.

The final case is where $y \vee z >_k x$. Note that in this case, we cannot have $x \vee z \leq_k x$, since we could then conclude from $z \leq_t x \vee z$ that $z \leq_t y$, and we would have $y \vee z = y \leq_k x$. Thus if $y \vee z >_k x$, $x \vee z >_k x$ as well. Now since $y \vee z \geq_t y$, we also have $y \vee z \geq_t x$; since $x \vee z \geq_t x$, we have $x \vee z \geq_t y$. It follows that $y \vee z = x \vee z$, and $\pi(y \vee z) = y \vee z = x \vee z = \pi(y) \vee \pi(z)$. □

Lemma 11.8. *Let B be a canonically grounded, stratified bilattice. Then if $x \leq_k y$ but $g_t(x) \not\leq_k g_t(y)$, there exists a z with $x \leq_k z \leq_k y$ such that B splits at z.*

Proof. Consider $x \wedge y$. Clearly $x \leq_k x \wedge y$; in addition,

$$g_t(x \wedge y) = g_t(x) \wedge g_t(y) = g_t(x) \cdot g_t(y) \leq_k g_t(x).$$

However, $g_t(x) \cdot g_t(y) \neq g_t(x)$, since $g_t(x) \not\leq_k g_t(y)$. Thus $g_t(x \wedge y) <_k g_t(x)$, while $x \wedge y \geq_k x$. Thus there exists a z with $y \geq_k x \wedge y \geq_k z \geq_k x$ such that B splits at z. □

Theorem 11.9. *A canonically grounded, stratified bilattice B is monotonic if and only if $t \cdot f = u$.*

Proof. If B is monotonic, we know from Lemma A.3 that $t \cdot f = u$.

For the converse, suppose that $t \cdot f = u$, and that B splits at x. Clearly $t \geq_k x$ and $f \geq_k x$, so $t \cdot f \geq_k x$, and $x = u$. Thus B is monotonic. □

Theorem 11.10. *Let B be a stratified bilattice. Then any truth assignment ϕ defined on B has a unique $\lhd$-minimal apparently closed extension.*

The proof requires the following lemma:

Lemma A.5 *Suppose B splits at x, with π the usual mapping from B to B^x. Now fix a truth assignment ϕ on B. Then if ψ is a $\lhd$-minimal apparently closed extension of ϕ, $\pi(\psi)$ is a $\lhd$-minimal apparently closed extension of $\pi(\phi)$.*

Proof. Definition 7.1 defined $\chi \lhd \xi$ to mean that either $\chi <_k \xi$ or there was some z with $\chi(z) \neq \xi(z)$ and

$$\sum\{\chi(p)|\chi(p) \neq \xi(p)\} <_k \xi(z).$$

An apparently closed extension ψ of ϕ will be $\lhd$-minimal if and only if for any other apparently closed extension ξ of ϕ, we do not have $\xi \lhd \psi$.

To prove the lemma, suppose that $\pi(\psi)$ is not $\lhd$-minimal on B^x, so that there exists some apparently closed extension χ of $\pi(\phi)$ with $\chi \lhd \pi(\psi)$ and, specifically, either $\chi <_k \pi(\psi)$ or there exists a z such that $\chi(z) \neq \pi[\psi(z)]$ and

$$\sum\{\chi(p)|\chi(p) \neq \pi[\psi(p)]\} <_k \pi[\psi(z)].$$

We will show that if $\chi <_k \pi(\psi)$, then $\chi \cdot \psi$ is an apparently closed extension of ϕ with $\chi \cdot \psi <_k \psi$, and if $\chi \not<_k \pi(\psi)$, so that the other clause in the definition of $\lhd$ holds, then χ is an apparently closed extension of ϕ with $\chi \lhd \psi$.

Now note that since B^x is a subbilattice of B, χ is also an apparently closed extension of ϕ on B. For the first claim made above, we must show that $\chi \cdot \psi$ is also an apparently closed truth assignment on B; that $\chi \cdot \psi <_k \psi$ is clear.

That $\chi \cdot \psi$ is $\neg$-closed is immediate, since each of χ and ψ is. Next, we need to show that since $\chi(p \wedge q) \geq_k \chi(p) \wedge \chi(q)$ and $\psi(p \wedge q) \geq_k \psi(p) \wedge \psi(q)$,

$$(\chi \cdot \psi)(p \wedge q) \geq_k (\chi \cdot \psi)(p) \wedge (\chi \cdot \psi)(q). \tag{38}$$

To see this, note first that χ and ψ are always k-comparable. The reason for this is that if $\chi(p) \not\leq_k \psi(p)$, since $\chi(p) \leq_k \pi[\psi(p)]$, we must have $\psi(p) <_k x$ and $\chi(p) = x$. (Recall that χ is a truth assignment on B^x.)

Now suppose that χ is $\leq_k \psi$ at both p and q. Then since $\psi(p) \geq_k x$ and $\psi(q) \geq_k x$, we will have $\psi(p) \wedge \psi(q) \geq_k x$, and $\psi(p \wedge q) \geq_k x$, so that $\chi(p \wedge q) \leq_k \psi(p \wedge q)$, and (38) follows from the corresponding equation for χ.

Alternatively, suppose that χ is $>_k \psi$ at both p and q. Then at both points, we must have $\chi = x$ and $\psi <_k x$, so that $\chi(p \wedge q) \geq_k x \geq_k \psi(p) \wedge \psi(q)$ and $\psi(p \wedge q) \geq_k \psi(p) \wedge \psi(q)$.

The remaining case is where $\chi(p) \leq_k \psi(p)$ and $\chi(q) >_k \psi(q)$. Now if $\chi(p \wedge q) >_k \psi(p \wedge q)$, then since $\psi(p \wedge q) <_k x$, we can conclude that $\psi(p) \wedge \psi(q) <_k x$ and therefore that $\psi(p) >_t \psi(p) \wedge \psi(q)$, so that $\psi(p) >_t \psi(q)$, since $\psi(q) \in B_x$. Now we also have that $\chi(p \wedge q) \geq_k \chi(p) \wedge \chi(q)$, or $x \geq_k \chi(p) \wedge x$. But $\chi(p) \geq_k x$, so this gives us $\chi(p) \geq_t x$. Thus $\chi(p) \geq_t \psi(q)$, and

$$\chi(p) \wedge \psi(q) = \psi(q) = \psi(p) \wedge \psi(q).$$

This allows us to conclude that $\psi(p \wedge q) \geq_k \chi(p) \wedge \psi(q)$ from the corresponding inequality for ψ.

Finally, we must consider the case where $\chi(p) \leq_k \psi(p)$ and $\chi(q) >_k \psi(q)$, but $\chi(p \wedge q) \leq_k \psi(p \wedge q)$. Here, we need to show that $\chi(p \wedge q) \geq_k \chi(p) \wedge \psi(q)$. But if $\chi(p) = x$, this is clear; if $\chi(p) >_k x$, then the result of conjoining $\chi(p)$ with any point in B_x is independent of the actual point in B_x selected, and we need only show that $\chi(p \wedge q) \geq_k \chi(p) \wedge x$. But since $\chi(q) = x$, this follows from the fact that χ is apparently closed.

Next, we must show that if $\chi(p) \geq_t \chi(q)$ and $\psi(p) \geq_t \psi(q)$, then $(\chi \cdot \psi)(p) \geq_t (\chi \cdot \psi)(q)$. This is easier.

If $\chi \geq_k \psi$ at both points or vice versa, the result follows immediately from the inequality on ψ or χ respectively. If $\chi(p) <_k \psi(p)$ and $\psi(q) <_k \chi(q)$, we must show that $\chi(p) \geq_t \psi(q)$. But this is a consequence of the fact that $\chi(p) \geq_t \chi(q) = x$, and $\psi(q) \in B_x$. The case where $\chi(p) >_k \psi(p)$ and $\psi(q) >_k \chi(q)$ is similar.

This completes the proof that if $\chi <_k \pi(\psi)$, then $\chi \cdot \psi$ is an apparently closed extension of ϕ with $\chi \cdot \psi <_k \psi$. It remains to show that if $\chi \lhd \pi(\psi)$ but $\chi \not<_k \pi(\psi)$, so that the other clause in the definition of $\lhd$ holds, then $\chi \lhd \psi$.

To see this, suppose that we have some z with $\chi(z) \neq \pi[\psi(z)]$ and $\sum\{\chi(p) | \chi(p) \neq \pi[\psi(p)]\} <_k \pi[\psi(z)]$. We will show that:

1. $\chi(z) \neq \psi(z)$, and
2. $\sum\{\chi(p) | \chi(p) \neq \psi(p)\} <_k \psi(z)$.

For the first, note that since χ is a truth assignment on B^x, we have $\chi(p) \geq_k x$ for all p. Now since $\chi \lhd \pi(\psi)$, we must have $x \leq_k \chi(z) <_k \pi[\psi(z)]$, so that $\pi[\psi(z)] = \psi(z)$ and therefore $\chi(z) \neq \psi(z)$.

For the second claim, we have:

$$\begin{aligned}
\sum\{\chi(p)|\chi(p) \neq \psi(p)\} &= \sum\{\chi(p)|\pi[\psi(p)] = \chi(p) \neq \psi(p)\} \\
&\quad + \sum\{\chi(p)|\pi[\psi(p)] \neq \chi(p) \neq \psi(p)\} \\
&\leq_k \sum\{\pi[\psi(p)]|\pi[\psi(p)] \neq \psi(p)\} + \sum\{\chi(p)|\pi[\psi(p)] \neq \chi(p)\} \\
&<_k x + \pi[\psi(z)] \\
&\leq_k x + \psi(z) = \psi(z).
\end{aligned}$$

Thus $\chi \lhd \psi$ on B, and ψ was not a $\lhd$-minimal apparently closed extension of ϕ. □

We now proceed with the proof of Theorem 11.10.

Proof. The proof proceeds by transfinite induction on the number of points at which B splits.

For the ground case, note that every bilattice splits at u and at $\perp$; if B splits only at these two points, then it must be monotonic. It follows that B is distributive, and therefore that any truth assignment has a unique k-minimal apparently closed extension.

For the inductive step, suppose that B has some ordinal number of splitting points, and that the theorem holds for any bilattice with fewer splitting points.

We note first that if the theorem is to fail, there cannot be any sequence of splitting points x_i that approaches u as a limit. Suppose that there were, and that ϕ were a truth assignment on B with two $\lhd$-minimal apparently closed extensions. Then for some finite i, these two minimal apparently closed extensions would be distinct in the reduced bilattice B^{x_i}, so that ϕ would have two $\lhd$-minimal apparently closed extensions in B^{x_i}. But B^{x_i} has fewer splitting points that B does.

We can therefore assume without loss of generality that B has some minimal nontrivial splitting point x. Now if ϕ is some truth assignment on B, the inductive hypothesis implies that ϕ will have a unique $\lhd$-minimal apparently closed extension on B^x, from which it follows that any two $\lhd$-minimal apparently closed extensions of ϕ will agree on B^x.

We claim that among the apparently closed extensions of ϕ that agree with any particular apparently closed extension ψ on B^x, there is a unique k-minimal (and therefore $\lhd$-minimal) one. To see this, consider the subbilattice B_x of points $y \in B$ with $y \leq_k x$. Note first that B_x has no nontrivial splitting points, and is therefore monotonic. Now restrict ϕ to B_x, restricting our language to the sublanguage L' of those sentences for which $\psi(x) \leq_k x$. It is clear that an extension of ϕ that agrees with ψ on B^x will be apparently closed on B if and only if ϕ is apparently closed with respect to L' on B_x, so that the set of apparently closed extensions of ϕ on B inherits a partial order from the set of apparently closed extensions on B_x. But this latter set has a minimal element, since B_x is distributive. Thus the set of apparently closed extensions of ϕ that agree with ψ *also* has a minimal element, and the theorem is proved. □

Lemma 11.12. *The relation $\approx_s$ is an equivalence relation on B.*

Proof. Immediate. □

Lemma 11.14. *The collection of stratification levels in a bilattice B inherits the k partial order of B.*

Proof. We must show that if $x \leq_k y$ for representatives x and y of $[x]_s$ and $[y]_s$ respectively, then either $[x]_s = [y]_s$ or $x' \leq_k y'$ for any $x' \in [x]_s$ and $y' \in [y]_s$.

If x and y are not stratification equivalent, there must exist a z that splits B with (say) $x \leq_k z$ but not $y \leq_k z$. Since B splits at z, we must therefore have $y \geq_k z$. Now for any x' that is stratification equivalent to x and y' that is stratification equivalent to y, we will have $x' \leq_k z$ and $z \leq_k y'$, so that $x' \leq_k y'$. □

Procedure 11.16 (Nonmonotonic inference I). Given as input a query q:

1. Set $\psi = \phi$. Set $F = \{(q, u)\}$. Each element of F is a pair (p, x), where p is a sentence to be investigated, and x is a point in the bilattice such that we are only interested in proofs of p that result in $\psi(p) >_s x$.
2. For some (p, x) in F, find a first-order proof of p from a set $U = \{p_i\}$ of sentences in the database with $\psi(U) \vee u \not\leq_t g_t[\psi(p)]$ and $\psi(U) >_s x$, using any conventional prover. If no such proof exists, stop and return $\psi(q)$.
3. Set $z = \psi(p)$, and $\psi(p) = \psi(p) + [\psi(U) \vee u]$.
4. If z is on a different stratification level from $\psi(p)$, then set $\psi(r) = \phi(r)$ for any r with $\psi(r) <_s \psi(p)$, and then set $\psi = \mathrm{cl}_\neg(\psi)$. Remove from F any pair (r, x) with $x <_s z$ and $r \neq q$.
5. If $\psi(U)$ is not on the top stratification level of the bilattice, add $(\neg \wedge p_i, \psi(U))$ to F. Return to step 2.

Theorem 11.17. *Let B be a canonically grounded, stratified bilattice. Then for any $\neg$-closed truth assignment ϕ and query q, the value returned by Procedure 11.16 will be $g_t[\mathrm{cl}(\phi)(q)]$.*

Proof. We need the following lemma:

Lemma A.6 *For any $\neg$-closed truth assignment ϕ and query q on a distributive (i.e., monotonic) bilattice B, the value returned by Procedure 11.16 will be $g_t[\mathrm{cl}(\phi)(q)]$.*

Proof. A monotonic bilattice has only one stratification level, so step 4 of the nonmonotonic procedure is irrelevant. In addition, we never need to modify F in the looping step 5. The nonmonotonic procedure therefore becomes:

1. Set $\psi = \phi$.
2. Find a first-order proof of q from a set $U = \{p_i\}$ of sentences in the database with $\psi(U) \vee u \not\leq_t g_t[\psi(q)]$ using any conventional prover. If no such proof exists, stop and return $\psi(q)$.
3. Set $z = \psi(q)$, and $\psi(q) = \psi(q) + [\psi(U) \vee u]$.
4. Return to step 2.

Swapping z and $\psi(q)$, we see that this is identical in form to the monotonic procedure.

The proof is not complete, however, since the definition of $\psi(U)$ is different for the monotonic and nonmonotonic cases. To show that the results are in fact the same, we must therefore show that

$$\sum_{\{p_i\} \models U} [\wedge_i \psi(p_i) + \psi(\wedge_i p_i)] \vee u = \sum_{\{p_i\} \models U} [\wedge_i \psi(p_i)] \vee u.$$

In a distributive bilattice this is immediate, since the sum on the left splits into:

$$\sum_{\{p_i\} \models U} [\wedge_i \psi(p_i)] \vee u + \sum_{\{p_i\} \models U} [\psi(\wedge_i p_i)] \vee u.$$

But the second term here is included in the first, since if $\{p_i\} \models U$, then $\wedge_i p_i \models U$. □

We now return to the proof of Theorem 11.17:

Proof. Once again, the proof is by induction on the number of splitting points in B. If B is monotonic, then the result follows from Lemma A.6.

For the inductive step, we may suppose as usual that B has some minimal nontrivial splitting point x. Now if $\mathrm{cl}(\phi)(q) >_k x$, the result follows immediately from the inductive step applied to the bilattice B^x. We can therefore assume without loss of generality that $\mathrm{cl}(\phi)(q) \leq_k x$.

Now let U be an element of $\pi_+(q)$, so that $U \models q$. We claim that either $\mathrm{cl}(\phi)(U) <_t u$, or $\mathrm{cl}(\phi)(U) \leq_k x$. To see this, note that the fixed-point result (20) implies that $\mathrm{cl}(\phi)(q) \geq_k \mathrm{cl}(\phi)(U) \vee u$. Now if $\mathrm{cl}(\phi)(U) >_s x$, since $\mathrm{cl}(\phi)(U) \vee u \leq_s x$, we must have $\mathrm{cl}(\phi)(U) \vee u \leq_t g_t(x)$, so that $\mathrm{cl}(\phi)(U) \leq_t g_t(x)$ and therefore $\mathrm{cl}(\phi)(U) \leq_t u$ by Lemma 11.5.

It follows from this that if we restrict our attention to the set $\pi_+^*(q)$ of U such that $U \models q$ and $\mathrm{cl}(\phi)(U) \leq_k x$, we have

$$g_t[\mathrm{cl}(\phi)(q)] = \sum_{U \in \pi_+^*(q)} \phi(U) \vee u. \tag{39}$$

Within this sum, it follows from arguments identical to those used in the proof of Corollary 10.3 that we need only consider elements of $\pi_+^*(q)$ satisfying the condition appearing in step 2 of the nonmonotonic procedure.

This shows that every term in (39) is in fact considered by Procedure 11.16; the difficulty is that there may be some set U with $U \models q$ that is used in step 2 even though $\mathrm{cl}(\phi)(U) \not\leq_k x$. The problem stems from the fact that our real interest is in whether or not $\phi(U) \vee u \leq_t g_t[\phi(p)]$ *in the closure*, but we can obviously only test for this using the current state of the truth assignment.

For any such U, we claim that after introducing U, the procedure will consider *some* sentence whose truth value jumps to a stratification level above $\phi(U)$. The theorem would then follow, since if this happens, the resetting step 4 will ensure that the truth value assigned to q is reset to $\phi(q)$ and recalculated.

To prove the claim, note that since $U \models q$ and $\mathrm{cl}(\phi)(U) \not\leq_k x$, we must have $\mathrm{cl}(\phi)(U) <_t u$. Thus

$$\bigwedge_{p_i \in U} \mathrm{cl}(\phi)(p_i) + \mathrm{cl}(\phi)(\wedge_i p_i) <_t u.$$

But $\wedge_i p_i \models p_j$ for any j, so $\mathrm{cl}(\phi)(\wedge_i p_i) \leq_t \mathrm{cl}(\phi)(p_j)$ for any j, and therefore $\wedge_i \mathrm{cl}(\phi)(p_i) + \mathrm{cl}(\phi)(\wedge_i p_i) \geq_t \mathrm{cl}(\phi)(\wedge_i p_i)$. Thus

$$\mathrm{cl}(\phi)(\wedge_i p_i) <_t u.$$

For notational simplicity, suppose that we set $p = \wedge_i p_i$, and that we denote by ψ the state of the truth assignment ϕ when the set U is considered. Now note that since ϕ is $\neg$-closed initially, it follows that $\phi(r) \approx_s \phi(\neg r)$ for all r when the procedure is invoked. Since the $\neg$-closure is taken again when the truth value of any sentence changes stratification level, we must have $\phi(r) \approx_s \phi(\neg r)$ throughout the execution of the procedure. Specifically, we must have $\psi(\neg p) \approx_s \psi(p)$.

Since U is considered at all by the nonmonotonic procedure, we must have $\psi(U) \not\leq_t u$, so that either $\psi(p) \not\leq_t u$ on the one hand, or $\wedge_i \psi(p_i) \not\leq_t u$ and $\psi(p) \not\geq_k \wedge_i \psi(p_i)$ on the other. In either case, since $\mathrm{cl}(\phi)(p) <_t u$ and $\mathrm{cl}(\phi)(p) \geq_k \psi(p)$, we must have $\mathrm{cl}(\phi)(p) >_s \psi(p)$. It follows from this that

$$\mathrm{cl}(\phi)(\neg p) >_s \psi(\neg p).$$

But now $\mathrm{cl}(\phi)(p) <_t u$, so $\mathrm{cl}(\phi)(\neg p) >_t u$, and

$$g_t[\mathrm{cl}(\phi)(\neg p)] \approx_s \mathrm{cl}(\phi)(\neg p) >_s \psi(\neg p).$$

But we know by the inductive hypothesis that the nonmonotonic procedure, if invoked to calculate $g_t(\neg p)$, will return the correct result. It follows that the introduction of $(\neg p, [\psi(p)]_s)$ into F in step 5 of the procedure will ensure that any invalid inferences drawn by the procedure will eventually be retracted. □

Theorem 11.18. *Suppose that $\phi(p) \neq u$ for a finite subset of our language L, that the bilattice B has a finite number of splitting points, and that the conventional prover used in Procedure 11.16 always terminates. Then the procedure itself terminates as well.*

Proof. The proof is essentially the same as that of Theorem 10.7. The arguments given there show that the number of sentences considered by the prover (including the conjunctions of sentences used in proofs of q) is finite, and therefore that the set F remains finite. It will follow that the procedure terminates if we can show that the set F is reduced by step 4 only finitely many times. But this is a consequence of the fact that the bilattice in question has only a finite number of stratification levels, and F is reduced only when the truth value assigned to some sentence increases stratification level. Since there are a finite number of sentences, and a finite number of stratification levels, the procedure must terminate. □

Theorem 11.19. *Fix z and y in a canonically grounded, stratified bilattice, with $z \geq_k y$ and $z \neq u$. Then z and y are on different stratification levels if and only if $g_t(z) \cdot \neg g_t(z) \not<_k y$ or $g_f(z) \cdot \neg g_f(z) \not<_k y$.*

Proof. We first show that if x is t-grounded and $x \cdot \neg x \neq u$, then x and $x \cdot \neg x$ are on different stratification levels. To see this, note that $g_f(x) = u$, since x is t-grounded, but $g_f(x \cdot \neg x) \neq u$. (If $g_f(x \cdot \neg x) = u$, since $x \cdot \neg x = \neg(x \cdot \neg x)$ it would follow that $g_t(x \cdot \neg x) = u$, and therefore $x \cdot \neg x = g_f(x \cdot \neg x) + g_t(x \cdot \neg x) = u$.) Thus $x \geq_k x \cdot \neg x$ but $g_f(x \cdot \neg x) >_k g_f(x)$, and B splits at a point between x and $x \cdot \neg x$.

Note that it follows from this that if $x \neq u$ is t-grounded, then x and $x \cdot \neg x$ are on different stratification levels. If $x \cdot \neg x \neq u$, we can apply the result of the last paragraph; if $x \cdot \neg x = u$, we can simply use the fact that every bilattice splits at u, so that u is on a stratification level by itself.

Now suppose that $g_t(z) \cdot \neg g_t(z) \not<_k y$. We can obviously assume without loss of generality that $y \neq u$, from which it follows that $g_t(z) \neq u$, so that

$$z \geq_s g_t(z) >_s g_t(z) \cdot \neg g_t(z). \tag{40}$$

Since $\geq_s$ respects the k partial order, we also know that $g_t(z) \cdot \neg g_t(z) \not<_s y$ but, since the s partial order is total, this implies $g_t(z) \cdot \neg g_t(z) \geq_s y$. Combining this with (40) gives us $z >_s y$.

Alternatively, suppose that z and y are on different stratification levels, so that $z >_s y$. Now we know that either $g_t(z) \approx_s z$ or $g_t(z) = u$, so either $g_t(z) \approx_s z$ or $g_f(z) \approx_s z$. In the former case, we will have $g_t(z) >_s y$, so that $g_t(z) \geq_k y$. But $\neg x \approx_s x$ for all x, so $\neg g_t(z) \geq_k y$ as well, and $g_t(z) \cdot \neg g_t(z) \geq_k y$. The case where $g_f(z) \approx_s z$ is similar. □

Corollary 11.20. *Fix z and y in a canonically grounded product of stratified bilattices, with $z \geq_k y$ and $z \neq u$. Then z and y are on different stratification levels in some coordinate if and only if $g_t(z) \cdot \neg g_t(z) \not\leq_k y$ or $g_f(z) \cdot \neg g_f(z) \not\leq_k y$.*

Proof. Immediate. □

Theorem 12.4. *Let A be the ATMS bilattice. Then the interpretation given by $\pi[\,a\ .\ b\,] = a$ for an element $[\,a\ .\ b\,]$ of A is a justified interpretation of A.*

Proof. This is in fact simply a restatement of Theorem 6.15 using the definitions of Section 12.1. □

Lemma 12.5. *Let B be an arbitrary bilattice, and ϕ a truth assignment on B. Then the interpretation given by $\pi(x) = \emptyset$ for all $x \in B$ is a justified interpretation, and ϕ is consistent with this interpretation.*

Proof. This is clear. Since the definition of a consistent truth assignment is satisfied trivially, *any* truth assignment on B will be consistent with the given interpretation. That the interpretation is justified follows from this. □

Procedure 12.6 (Nonmonotonic inference II). Given as input a query q, a bilattice B, and a justified interpretation for B. We assume that ϕ is a $\neg$-closed, consistent truth assignment on B.

1. Set $\psi = \phi$, and $F = \{(q, u)\}$.
2. For some (p, x) in F, find a first-order proof of p from a set $U = \{p_i\}$ of sentences in the database with $\psi(U) \vee u \not\leq_t g_t[\psi(p)]$ and $\psi(U) >_s x$, using any conventional prover. If no such proof exists, stop and return $\psi(q)$.
3. Set $z = \psi(p)$, and $\psi(p) = \psi(p) + [\psi(U) \vee u]$.
4. If z is on a different stratification level from $\psi(p)$, then set

$$\psi(r) = \phi(r) + \sum\{\psi(U) \vee u | U \in \pi[\psi(r)]\} \tag{41}$$

for any r with $\psi(r) \leq_s z$. Set $\psi = \mathrm{cl}_\neg \psi$.

5. If $\psi(U)$ is not on the top stratification level of the bilattice, add $(\neg \wedge p_i, \psi(U))$ to F. Return to step 2.

Theorem 12.7 (Nonmonotonic backward inference). *Let B be a canonically grounded product of stratified bilattices, and π a justified interpretation for B. Then for any $\neg$-closed truth assignment ϕ consistent with this justification and query q, the value returned by Procedure 12.6 will be $g_t[\mathrm{cl}(\phi)(q)]$.*

Proof. We need to show that the differences between Procedure 12.6 and its predecessor 11.16 do not affect the proof of Theorem 11.17.

In step 4, we now compare r's stratification level to p's *original* stratification level, as opposed to the new value. Since a proof depending on a truth value on one stratification level can never change a truth value on a higher stratification level, this is justified.

The resetting step now reads essentially:

4. If the stratification level of p changed, then reconsider the truth value of every sentence that depended on p for its proof.

This is precisely what is needed to make the proof of Theorem 11.17 work. □

Theorem 12.8. *Suppose that $\phi(p) \neq u$ for a finite subset of our language L, that the bilattice B has a finite number of splitting points, and that the conventional prover used in Procedure 12.6 always terminates. Then the procedure itself terminates as well.*

Proof. The proof is unchanged from the proof of Theorem 11.18. □

B Code

B.1 LISP description of the first-order bilattice

```
;; Typical bilattice description.

;; basic bilattice stuff.  Expects five functions:
;;      (mvl-plus x y) (mvl-dot x y) (mvl-and x y) (mvl-or x y) (mvl-not x)
;; [supplied functions are for first-order logic bilattice]

(defconst true 'true "t in bilattice")
(defconst false 'false "f in bilattice")
(defconst unknown 'unknown "u in bilattice")
(defconst bottom 'bottom "contradiction in bilattice")

(defconst std-cutoff `((,true . t-ge)))
(defconst std-success `((,true . t-ge)))

(defun mvl-plus (x y)
  (cond ((eq x unknown) y)
        ((or (eq y unknown) (eq y x)) x)
        (t bottom)))

(defun mvl-dot (x y)
  (cond ((eq x bottom) y)
        ((or (eq y bottom) (eq y x)) x)
        (t unknown)))

(defun mvl-and (x y)
  (cond ((eq x true) y)
        ((or (eq y true) (eq y x)) x)
        (t false)))

(defun mvl-or (x y)
  (cond ((eq x false) y)
```

```
        ((or (eq y false) (eq y x)) x)
        (t true)))

(defun mvl-not (x)
  (cond ((eq x true) false)
        ((eq x false) true)
        (x)))

(defun stash-value (ignore) true)
(defun unstash-value (ignore) unknown)

;; special function for comparing two values

(defun mvl-eq (x y) (eq x y))

(defun pi (ignore) nil)                ; justification information
```

B.2 MRS description of the full adder

```
;; inputs are on, off and on
(val (in 1 f1) on)                     ; P1
(val (in 2 f1) off)                    ; P2
(val (in 3 f1) on)                     ; P3

;; descriptions of various gates
(type x1 xorg)                         ; P4
(type x2 xorg)                         ; P5
(type a1 andg)                         ; P6
(type a2 andg)                         ; P7
(type o1 org)                          ; P8

;; interconnections
(conn (in 1 f1) (in 1 x1))             ; P9
(conn (in 1 f1) (in 1 a1))             ; P10
(conn (in 2 f1) (in 2 x1))             ; P11
(conn (in 2 f1) (in 2 a1))             ; P12
(conn (in 3 f1) (in 2 x2))             ; P13
(conn (in 3 f1) (in 1 a2))             ; P14
(conn (out 1 x1) (in 1 x2))            ; P15
(conn (out 1 x1) (in 2 a2))            ; P16
(conn (out 1 a2) (in 1 o1))            ; P17
(conn (out 1 a1) (in 2 o1))            ; P18
(conn (out 1 x2) (out 1 f1))           ; P19
(conn (out 1 o1) (out 2 f1))           ; P20

(<= (val $y $z)                        ; what connections mean
    (and (conn $x $y) (val $x $z)))    ; P21

(<= (val (out 1 $x) on)                ; behavior of and gates (P22 P23)
    (and (type $x andg) (val (in 1 $x) on) (val (in 2 $x) on)))
(<= (val (out 1 $x) off)
    (and (type $x andg) (val (in $n $x) off)))

(<= (val (out 1 $x) off)               ; or gates (P24 P25)
    (and (type $x org) (val (in 1 $x) off) (val (in 2 $x) off)))
(<= (val (out 1 $x) on)
    (and (type $x org) (val (in $n $x) on)))

(<= (val (out 1 $x) off)               ; xor gates (P26 P27 P28)
    (and (type $x xorg) (val (in 1 $x) $z) (val (in 2 $x) $z)))
(<= (val (out 1 $x) on)
    (and (type $x xorg) (val (in 1 $x) on) (val (in 2 $x) off)))
(<= (val (out 1 $x) on)
    (and (type $x xorg) (val (in 1 $x) off) (val (in 2 $x) on)))
```

B.3 LISP code for the justification lattice

```
(defconst max-j '(()) "maximal justification")
(defconst min-j nil "minimal justification")

(defun contains (x y) (every y #'(lambda (z) (memq z x))))      ;x contains y

(defun j-plus (j1 j2) (jl-simplify (append j1 j2)))
(defun j-dot (j1 j2)
  (jl-simplify
    (mapcan #'(lambda (x)
                (mapcar #'(lambda (y) (j-simplify (append x y))) j1)) j2)))

(defun j-simplify (l)                   ; remove multiple entries
  (do ((i l (cdr i)))
      ((null i) l)
    (rplacd i (delq (car i) (cdr i)))))

(defun jl-simplify (l)                  ; remove subsumed entries
  (do ((i l (cdr i)))
      ((null i) l)
    (if (some l #'(lambda (x)
                    (and (not (eq x (car i))) (contains (car i) x))))
        (setq l (delq (car i) l 1)))))

;; special function for comparing two values

(defun j-eq (j1 j2)
  (and (= (length j1) (length j2))
       (every j1 #'(lambda (x) (some j2 #'(lambda (y) (j-same x y)))))))

(defun j-same (x y)
  (and (= (length x) (length y))
       (every x #'(lambda (z) (memq z y)))))
```

B.4 LISP code for the ATMS bilattice

```
;;; bilattice for atms

(defconst true (cons max-j min-j) "t in bilattice")
(defconst false (cons min-j max-j) "f in bilattice")
(defconst unknown (cons min-j min-j) "u in bilattice")
(defconst bottom (cons max-j max-j) "contradiction in bilattice")

(defconst std-cutoff `((,unknown . t-gt)))
(defconst std-success `((,true . t-ge)))

(defun mvl-plus (x y)
  (cons (j-plus (car x) (car y)) (j-plus (cdr x) (cdr y))))

(defun mvl-dot (x y)
  (cons (j-dot (car x) (car y)) (j-dot (cdr x) (cdr y))))

(defun mvl-and (x y)
  (cons (j-dot (car x) (car y)) (j-plus (cdr x) (cdr y))))

(defun mvl-or (x y)
  (cons (j-plus (car x) (car y)) (j-dot (cdr x) (cdr y))))

(defun mvl-not (x)
  (cons (cdr x) (car x)))

(defun stash-value (p) `(((,(prop-name p))) . nil))
(defun unstash-value (ignore) unknown)
```

```
(defun mvl-eq (x y)
  (and (j-eq (car x) (car y)) (j-eq (cdr x) (cdr y))))

(defun pi (x) (car x))
```

B.5 Alternative description of the full adder

```
;;; full adder with justification information
;;; (gate information self-justified)

(mvl-stash '(val (in 1 f1) on) true)              ; P1
(mvl-stash '(val (in 2 f1) off) true)             ; P2
(mvl-stash '(val (in 3 f1) on) true)              ; P3

(mvl-stash '(type x1 xorg))                       ; P4  (P4 - P8 are assumptions)
(mvl-stash '(type x2 xorg))                       ; P5
(mvl-stash '(type a1 andg))                       ; P6
(mvl-stash '(type a2 andg))                       ; P7
(mvl-stash '(type o1 org))                        ; P8

(mvl-stash '(conn (in 1 f1) (in 1 x1)) true)      ; P9
(mvl-stash '(conn (in 1 f1) (in 1 a1)) true)      ; P10
(mvl-stash '(conn (in 2 f1) (in 2 x1)) true)      ; P11
(mvl-stash '(conn (in 2 f1) (in 2 a1)) true)      ; P12
(mvl-stash '(conn (in 3 f1) (in 2 x2)) true)      ; P13
(mvl-stash '(conn (in 3 f1) (in 1 a2)) true)      ; P14
(mvl-stash '(conn (out 1 x1) (in 1 x2)) true)     ; P15
(mvl-stash '(conn (out 1 x1) (in 2 a2)) true)     ; P16
(mvl-stash '(conn (out 1 a2) (in 1 o1)) true)     ; P17
(mvl-stash '(conn (out 1 a1) (in 2 o1)) true)     ; P18
(mvl-stash '(conn (out 1 x2) (out 1 f1)) true)    ; P19
(mvl-stash '(conn (out 1 o1) (out 2 f1)) true)    ; P20

(mvl-stash '(<= (val $y $z)                       ; P21
    (and (conn $x $y) (val $x $z))) true)

(mvl-stash '(<= (val (out 1 $x) on)               ; P22
    (and (type $x andg) (val (in 1 $x) on) (val (in 2 $x) on))) true)
(mvl-stash '(<= (val (out 1 $x) off)              ; P23
    (and (type $x andg) (val (in $n $x) off))) true)

(mvl-stash '(<= (val (out 1 $x) off)              ; P24
    (and (type $x org) (val (in 1 $x) off) (val (in 2 $x) off))) true)
(mvl-stash '(<= (val (out 1 $x) on)               ; P25
    (and (type $x org) (val (in $n $x) on))) true)

(mvl-stash '(<= (val (out 1 $x) off)              ; P26
    (and (type $x xorg) (val (in 1 $x) $z) (val (in 2 $x) $z))) true)
(mvl-stash '(<= (val (out 1 $x) on)               ; P27
    (and (type $x xorg) (val (in 1 $x) on) (val (in 2 $x) off))) true)
(mvl-stash '(<= (val (out 1 $x) on)               ; P28
    (and (type $x xorg) (val (in 1 $x) off) (val (in 2 $x) on))) true)
```

B.6 LISP version of the default bilattice

```
;;; bilattice for default logic

(defconst true 'true "t in bilattice")
(defconst false 'false "f in bilattice")
(defconst unknown 'unknown "u in bilattice")
(defconst bottom 'bottom "contradiction in bilattice")

(defconst std-cutoff `((,unknown . t-gt)))
```

```
(defconst std-success `((,true . t-ge)))

(defconst star '* "*")
(defconst dt 'dt "dt")
(defconst df 'df "df")

(putprop bottom 0 'index)
(putprop false 1 'index)
(putprop true 2 'index)
(putprop star 3 'index)
(putprop df 4 'index)
(putprop dt 5 'index)
(putprop unknown 6 'index)

(declare (special def-name def-plus def-dot def-and def-or def-not))

(defun mvl-plus (x y)
  (aref def-name (aref def-plus (get x 'index) (get y 'index))))

(defun mvl-dot (x y)
  (aref def-name (aref def-dot (get x 'index) (get y 'index))))

(defun mvl-and (x y)
  (aref def-name (aref def-and (get x 'index) (get y 'index))))

(defun mvl-or (x y)
  (aref def-name (aref def-or (get x 'index) (get y 'index))))

(defun mvl-not (x)
  (aref def-name (aref def-not (get x 'index))))

(defun stash-value (ignore) true)
(defun unstash-value (ignore) unknown)

;; special function for comparing two values

(defun mvl-eq (x y) (eq x y))

(defun pi (ignore) nil)
```

B.7 The arrays describing the default bilattice

```
;;; arrays containing information about default bilattice functions

(setq def-name (make-array 7)
      def-plus (make-array '(7 7))
      def-dot (make-array '(7 7))
      def-and (make-array '(7 7))
      def-or (make-array '(7 7))
      def-not (make-array 7))

(fillarray def-name (list bottom false true star df dt unknown))

(fillarray def-plus                 ; plus  b  f  t  * df dt  u
           '(0 0 0 0 0 0 0          ;    b  b  b  b  b  b  b  b
             0 1 0 1 1 1 1          ;    f  b  f  b  f  f  f  f
             0 0 2 2 2 2 2          ;    t  b  b  t  t  t  t  t
             0 1 2 3 3 3 3          ;    *  b  f  t  *  *  *  *
             0 1 2 3 4 3 4          ;   df  b  f  t  * df  * df
             0 1 2 3 3 5 5          ;   dt  b  f  t  *  * dt dt
             0 1 2 3 4 5 6))        ;    u  b  f  t  * df dt  u

(fillarray def-dot                  ;  dot  b  f  t  * df dt  u
           '(0 1 2 3 4 5 6          ;    b  b  f  t  * df dt  u
```

```
            1 1 3 3 4 5 6          ;   f  f  f  *  * df dt  u
            2 3 2 3 4 5 6          ;   t  t  *  t  * df dt  u
            3 3 3 3 4 5 6          ;   *  *  *  *  * df dt  u
            4 4 4 4 4 6 6          ;  df df df df df df  u  u
            5 5 5 5 6 5 6          ;  dt dt dt dt dt  u dt  u
            6 6 6 6 6 6 6))        ;   u  u  u  u  u  u  u  u

(fillarray def-and                 ; and  b  f  t  * df dt  u
          '(0 1 0 1 1 1 1          ;   b  b  f  b  f  f  f  f
            1 1 1 1 1 1 1          ;   f  f  f  f  f  f  f  f
            0 1 2 3 4 5 6          ;   t  b  f  t  * df dt  u
            1 1 3 3 4 3 4          ;   *  f  f  *  * df  * df
            1 1 4 4 4 4 4          ;  df  f  f df df df df df
            1 1 5 3 4 5 6          ;  dt  f  f dt  * df dt  u
            1 1 6 4 4 6 6))        ;   u  f  f  u df df  u  u

(fillarray def-or                  ;  or  b  f  t  * df dt  u
          '(0 0 2 2 2 2 2          ;   b  b  b  t  t  t  t  t
            0 1 2 3 4 5 6          ;   f  b  f  t  * df dt  u
            2 2 2 2 2 2 2          ;   t  t  t  t  t  t  t  t
            2 3 2 3 3 5 5          ;   *  t  *  t  *  * dt dt
            2 4 2 3 4 5 6          ;  df  t df  t  * df dt  u
            2 5 2 5 5 5 5          ;  dt  t dt  t dt dt dt dt
            2 6 2 5 6 5 6))        ;   u  t  u  t dt  u dt  u

                                   ; not  b  f  t  * df dt  u
(fillarray def-not '(0 2 1 3 5 4 6)) ;    b  t  f  * dt df  u
```

Acknowledgement

This work has been supported by DARPA under grant number N00039-86-C-0033, by ONR under grant number N00014-81-K-0004, by NSF under grant number DCR-8620059, and by the Rockwell Palo Alto Laboratory. I would like to thank the staff and students of the Logic Group at Stanford for providing so stimulating an environment in which to work, and the anonymous reviewers for constructive suggestions. I would specifically like to thank Hector Levesque, Vladimir Lifschitz, John McCarthy, Narinder Singh, David Smith and especially Karen Myers for many useful discussions and suggestions.

References

[1] K. Apt, H. Blair, and A. Walker. Towards a theory of declarative knowledge. In *Proceedings of the 1986 Workshop on Foundations of Deductive Databases and Logic Programming*, 1986.

[2] N. D. Belnap. How a computer should think. In *Proceedings of the Oxford International Symposium on Contemporary Aspects of Philosophy*, pages 30–56, 1975.

[3] N. D. Belnap. A useful four-valued logic. In G. Epstein and J. Dumm, editors, *Modern Uses of Multiple-valued Logic*, pages 8–37. D. Reidel Publishing Company, Boston, 1977.

[4] W. F. Clocksin and C. S. Mellish. *Programming in Prolog*. Springer-Verlag, Berlin, 1981.

[5] J. de Kleer. An assumption-based truth maintenance system. *Artificial Intelligence*, 28:127–162, 1986.

[6] J. de Kleer and B. C. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32:97–130, 1987.

[7] J. P. Delgrande. A formal approach to learning from examples. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 315–322, 1987.

[8] A. P. Dempster. A generalization of Bayesian inference. *J. Roy. Stat. Soc. B*, 30:205–247, 1968.

[9] J. Doyle. A truth maintenance system. *Artificial Intelligence*, 12:231–272, 1979.

[10] R. Duda, P. Hart, and N. Nilsson. Subjective Bayesian methods for rule-based inference systems. In *Proceedings 1976 National Computer Conference*, pages 1075–1082. AFIPS, 1976.

[11] H. B. Enderton. *A Mathematical Introduction to Logic.* Academic Press, New York, 1972.

[12] D. W. Etherington. *Reasoning with Incomplete Information: Investigations of Non-Monotonic Reasoning.* PhD thesis, University of British Columbia, Vancouver, Canada, 1986.

[13] D. W. Etherington and R. Reiter. On inheritance hierarchies with exceptions. In *Proceedings of the Third National Conference on Artificial Intelligence*, pages 104–108, 1983.

[14] A. V. Gelder. Negation as failure using tight derivations for general logic programs. In *Proceedings of the 1986 Workshop on Foundations of Deductive Databases and Logic Programming*, 1986.

[15] M. Gelfond. On stratified autoepistemic theories. Technical report, University of Texas at El Paso, 1986.

[16] M. R. Genesereth. An overview of meta-level architecture. In *Proceedings of the Third National Conference on Artificial Intelligence*, pages 119–124, 1983.

[17] M. R. Genesereth. The use of design descriptions in automated diagnosis. *Artificial Intelligence*, 24:411–436, 1985.

[18] M. L. Ginsberg. Analyzing incomplete information. Technical Report 84-17, KSL, Stanford University, 1984.

[19] M. L. Ginsberg. Non-monotonic reasoning using Dempster's rule. In *Proceedings of the Fourth National Conference on Artificial Intelligence*, pages 126–129, 1984.

[20] M. L. Ginsberg. Implementing probabilistic reasoning. In *Proceedings 1985 Workshop on Uncertainty in Artificial Intelligence*, pages 84–90, Los Angeles, CA, 1985.

[21] M. L. Ginsberg. Bilattices. Technical Report 86-72, KSL, Stanford University, 1986.

[22] M. L. Ginsberg. Counterfactuals. *Artificial Intelligence*, 30:35–79, 1986.

[23] M. L. Ginsberg. Possible worlds planning. In *Proceedings of the 1986 Workshop on Planning and Reasoning about Action*, pages 213–243, Timberline, Oregon, 1986. Morgan Kaufmann.

[24] M. L. Ginsberg. A circumscriptive theorem prover. *Artificial Intelligence*, 39:209–230, 1989.

[25] M. L. Ginsberg and D. E. Smith. Reasoning about action I: A possible worlds approach. *Artificial Intelligence*, 35:165–195, 1988.

[26] G. Grätzer. *General Lattice Theory.* Birkhäuser Verlag, Basel, 1978.

[27] S. Hanks and D. McDermott. Nonmonotonic logics and temporal projection. *Artificial Intelligence*, 33:379–412, 1987.

[28] T. Imielinski. Results on translating defaults to circumscription. *Artificial Intelligence*, 32:131–146, 1987.

[29] G. L. S. Jr. *Common Lisp: The Language.* Digital Press, Billerica, MA, 1984.

[30] T. S. Kuhn. *The Structure of Scientific Revolutions.* University of Chicago Press, Chicago, 1962.

[31] V. Lifschitz. Pointwise circumscription: Preliminary report. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 406–410, 1986.

[32] V. Lifschitz. Formal theories of action. In *Proceedings of the 1987 Workshop on the Frame Problem in Artificial Intelligence*, Lawrence, Kansas, 1987.

[33] J. McCarthy. Epistemological problems of artificial intelligence. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pages 1038–1044, Cambridge, MA, 1977.

[34] J. McCarthy. Circumscription – a form of non-monotonic reasoning. *Artificial Intelligence*, 13:27–39, 1980.

[35] J. McCarthy. Applications of circumscription to formalizing common sense knowledge. *Artificial Intelligence*, 28:89–116, 1986.

[36] D. McDermott. Non-monotonic logic II. *Journal ACM*, 29:33–57, 1982.

[37] D. McDermott. A critique of pure reason. *Computational Intelligence*, 3:151–160, 1987.

[38] D. McDermott and J. Doyle. Non-monotonic logic I. *Artificial Intelligence*, 13:41–72, 1980.

[39] R. Moore. Semantical considerations on nonmonotonic logic. *Artificial Intelligence*, 25:75–94, 1985.

[40] N. J. Nilsson. Probabilistic logic. *Artificial Intelligence*, 28:71–87, 1986.

[41] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[42] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32:57–95, 1987.

[43] R. Reiter and G. Criscuolo. On interacting defaults. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, pages 270–276, 1981.

[44] R. Reiter and J. de Kleer. Foundations of assumption-based truth maintenance systems: Preliminary report. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 183–188, 1987.

[45] S. Russell. The compleat guide to MRS. Technical Report STAN-CS-85-1080, Stanford University, June 1985.

[46] E. Sandewall. A functional approach to non-monotonic logic. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 100–106, 1985.

[47] D. S. Scott. Some ordered sets in computer science. In I. Rival, editor, *Ordered Sets*, pages 677–718. D. Reidel Publishing Company, Boston, 1982.

[48] G. Shafer. *A Mathematical Theory of Evidence.* Princeton University Press, Princeton, NJ, 1976.

[49] Y. Shoham. Chronological ignorance. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 389–393, 1986.

[50] E. H. Shortliffe. *Computer-based Medical Consultations: MYCIN.* American Elsevier, New York, 1976.

[51] L. A. Zadeh. Fuzzy logic and approximate reasoning. *Synthese*, 30:407–428, 1975.