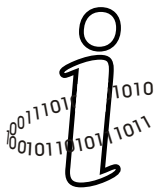
Faculty of Mathematics, Physics, and Informatics
Comenius University, Bratislava

Logic program semantics via an argumentation semantics.

Monika Adamová, Ján Šefránek

TR-2012-033



Technical Reports in Informatics

Logic program semantcs via an argumentation semantics

Monika Adamová, Ján Šefránek

Comenius University, Bratislava, Slovakia, monika.adamova@gmail.com;
sefranek@ii.fmph.uniba.sk

Abstract. There are various semantics designed for argumentation frameworks. They enable to assign a meaning e.g. to odd-length cycles. Our main motivation is to transfer semantics proposed by Baroni, Giacomin and Guida for argumentation frameworks with odd-length cycles to logic programs with odd-length cycles through default negation. The developed construction is even stronger. For a given logic program an argumentation framework is defined. The construction enables to transfer each semantics of the resulting argumentation framework to a semantics of the given logic program.

Keywords: argumentation framework; extension; logic program; odd cycle; semantics

1 Introduction

Relations between abstract argumentation frameworks and logic programs were studied since the fundamental paper by Dung [10]. Among typical research problems are, e.g., a characterization of extensions of abstract argumentation framework in terms of answer sets or other semantics of logic programs, a construction of new semantics of logic programs, based or inspired by extensions of argumentation frameworks, encoding extensions in answer set programming.

Our main motivation is to transfer semantics proposed in [3] for argumentation frameworks with odd-length cycles to logic programs with odd-length cycles through default negation. According to our knowledge, only CF2 extensions of [3], were studied from different logic programming points of view, see, e.g., [14,17]. In [14] an ASP-encoding of CF2 is presented and in [17] a characterization of CF2 in terms of answer set models is proposed.

Our goal is to propose some new semantics of logic programs via transferring semantics of argumentation frameworks. We propose a uniform method, which for a given logic program transfers arbitrary argumentation semantics to a semantics of the logic program. The method enables to define for the given logic program a corresponding argumentation framework. As the next step, an arbitrary semantics of the resulting argumentation framework is transferred to a semantics of the given logic program. The main contribution of our paper is this uniform method with a rather interesting comoputational properties.

The paper is structured as follows. Basics of SCC-recursive semantics of [3] is sketched after technical preliminaries. Then, in Section 4, a straightforward transfer of an argumentation semantics to a logic program semantics is described. However, the presented solution is counterintuitive for some cases. An improved method, based on unfolded form of a given program is presented in Section 5. After that, in Section 6, our method is compared to methods of [10] and [4]. Possibilties of semantic characterization of logic programs with odd-length cycles through default negation are disccussed, too.

A representation of an argumentation framework A by a logic program P is described in Section 6. It is shown that for an arbitrary argumentation semantics holds that extensions of the original argumentation framework A coincide with extensions of the argumentation framework constructed for P using our method. Finally, related work is overviewed and main contributions, open problems and future goals are summarized in Conclusions.

This paper is an essentially modified version of WLP2011-paper [2].

2 Preliminaries

Some basic notions of argumentation frameworks and logic programs are introduced in this section.

Argumentation frameworks An *argumentation framework* [10] is a pair $AF = (AR, atatcks)$, where AR is a set (of arguments) and $attacks \subseteq AR \times AR$ is a binary relation. Let be $a, b \in AR$; if $(a, b) \in atatcks$, it is said that a attacks b.

Let be $S \subseteq AR$. It is said that S is *conflict-free* if for no $a, b \in S$ holds $(a, b) \in attacks$.

A set of arguments $S \subseteq AR$ attacks $a \in AR$ iff there is $b \in S$ s.t. $(b, a) \in attacks$.

A conflict-free set of arguments S is *admissible* in AF iff for each $a \in S$ holds: if there is $b \in AR$ s.t. $(b, a) \in attacks$, then S attacks b. An admissible set of arguments counterattacks each attack against its members.

Dung defined some semantic characterizations (extensions) of argumentation frameworks as sets of conflict-free and admissible arguments, which satisfy also some other conditions.

A *preferred extension* of AF is a maximal admissible set in AF. A conflict-free $S \subseteq AR$ is a *stable extension* of AF iff S attacks each $a \in AR \setminus S$.

The *characteristic function* F_{AF} of an argumentation framework AF assigns sets of arguments to sets of arguments, where $F_{AF}(S) = \{a \in AR \mid \forall b \in AR\ b \text{ attacks } a \Rightarrow S \text{ attacks } b\}$.

The *grounded* extension of an argumentation framework AF is the least fixed point of F_{AF} (F_{AF} is monotonic).

A *complete extension* is an admissible set S of arguments s.t. each argument, which is acceptable with respect to S, belongs to S.

A set of extensions assigned by a semantics $\mathcal{S}$ to an argumentation framework AF is denoted by $\mathcal{E}_S(AF)$. A *semantics* $\mathcal{S}$ of AF is a mapping σ_S, which assigns

$\mathcal{E}_S(AF)$ to AF. We consider only coflict-free semantics in the following sense: if $\mathcal{S}$ is a semantics, $M \in \mathcal{E}_S(AF)$, then M is a conflict-free set of arguments.

Logic programs Only propositional normal logic programs are considered in this paper. Let $\mathcal{A}$ be a set of atoms. The set of default literals is $\mathcal{D} = not\ \mathcal{A} = \{not\ A \mid A \in \mathcal{A}\}$. A literal is an atom or a default literal. The set of all literals is denoted by $\mathcal{L}$. A rule r is an expression of the form

$$A \leftarrow A_1, \ldots, A_k, not\ B_1, \ldots, not\ B_m; \text{ where } k \geq 0, m \geq 0 \tag{1}$$

A is called the head of the rule and denoted by $head(r)$. The set of literals $\{A_1, \ldots, A_k, not\ B_1, \ldots, not\ B_m\}$ is called the body of r and denoted by $body(r)$. $\{A_1, \ldots, A_k\}$, called the positive part of the body, is denoted by $body^+(r)$ and $\{B_1, \ldots, B_m\}$ is denoted by $body^-(r)$. Notice that $body^-(r)$ differs from the negative part $\{not\ B_1, \ldots, not\ B_m\}$ of the body. If R is a set of rules, then $head(R) = \{head(r) \mid r \in R\}, body^+(R) = \{body^+(r) \mid r \in R\}, body^-(R) = \{body^-(r) \mid r \in R\}$.

A (normal) program is a finite set of rules. We will often use only the term program.

An interpretation is a consistent subset of $\mathcal{L}$, i.e., $I \subseteq \mathcal{L}$ s.t. for no atom A holds that $\{A, not\ A\} \subseteq I$. An interprettion I is total iff for each atom A either $A \in I$ or $not\ A \in I$. A rule r is satisfied in an interpretation I iff $head(r) \in I$ whenever $body(r) \subseteq I$. An interpretation I is a model of P iff each rule $r \in P$ is satisfied in I.

Our method transfers an argumentation semantics to a logic program semantics. The logic program semantics is specified in terms of *sets of atoms derivable* in the corresponding logic program. We follow the approach of Dimopoulos and Torres [7]: the derivation dependends on a set of default literals.[1] In the next paragraphs we will adapt some basic definitions from [7].

An *assumption* is a default literal. Let Δ be a set of assumptions. $\Delta^{\curvearrowright^P}$ is a set of atoms, derivable from Δ w.r.t. a program P; here is a precise definition:

Let a set of assumptions Δ and a program P be given. The program P_Δ is obtained from P by deleting elements of Δ from the bodies of the rules. The program P^+_Δ is obtained from P_Δ by deleting all rules with bodies containing assumptions. Finally, $\Delta^{\curvearrowright^P} = \{A \in \mathcal{A} \mid P^+_\Delta \models A\}$).

A set of assumptions Δ is conflict-free w.r.t. a program P iff $\Delta \cup \Delta^{\curvearrowright^P}$ is an interpretation. This view on interpretations enables an easy connection to the approach and results of [4] and a comfortable switching from two-valued to three-valued interprattaions and back.

It is said that an atom A is *derived* from Δ using rules of P iff $A \in \Delta^{\curvearrowright^P}$. Stable model semantics of logic programs play a background role in our paper, so, we introduce a definition of stable model. An interpretation $S = \Delta \cup \Delta^{\curvearrowright^P}$ is a stable model of P iff S is total interpretation [7].

If P is an empty program, then an empty interpretation $\Delta \cup \Delta^{\curvearrowright^P}$ is its stable model, i.e., both Δ and $\Delta^{\curvearrowright^P}$ are empty sets of literals.

[1] A similar viewpoint is accepted also in [10] and [4].

3 SCC-recursive semantics

Assymmetries in handling even and odd cycles in argumentation semantics are analysed in [3]. We present only a sketchy view of their approach. An argumentation framework AF may be conceived as an oriented graph G_{AF} with arguments as vertices and the attack relation as the set of edges.

Example 1 *Consider* $AF = (\{a, b, c\}, \{(a, b), (b, c), (c, a)\})$. *The graph representation of AF contains an odd-length cycle.*

Stable semantics does not assign an extension to such argumentation framework. However, there are two stable extensions for a cycle of length four.

Assymetries between semantic treatment of odd-length and even-length cycles are present also in other "classic" argumentation semantics proposed in [10].[2] *This motivated the research and solutions of [3].* □

A general recursive schema for argumentation semantics is proposed in [3]. The approach si based on strongly connected components.

Definition 1 *Let an argumentation framework* $AF = \langle AR, attacks \rangle$ *be given. A binary relation of path equivalence, denoted by* $PE_{AF} \subseteq (AR \times AR)$, *is defined as follows.*

- $\forall a \in AR, (a, a) \in PE_{AF}$,
- $\forall a \neq b \in AR, (a, b) \in PE_{AF}$ *iff there is a path from* a *to* b *and a path from* b *to* a.

The strongly connected components *of AF are the equivalence classes of arguments (vertices) under the relation of path-equivalence. The set of the strongly connected components of AF is denoted by* $SCCS_{AF}$. □

If we consider $SCCS_{AF}$ then the graph G_{AF} may be viewed as an acyclic one. Recursive semantics over $SCCS_{AF}$ are defined in a constructive way – an incremental process of adding arguments into an extension is specified. A symmetric handling of odd and even cycles is enabled by this construction.

Notions of parents and ancestors for graphs with strongly connected componets are defined in an obvious way. Initial components (components without parents) provide a basis for a construction of an extension. We start at the initial component and proceed via oriented edges to next components. If we construct an extension E and a component C is currently processed, the process consists in a choice of a subset of C, i.e. a choice of $E \cap C$ (according to the given semantics – the semantics specifies how choices depend on choices made in ancestors of C). A base function is assumed, which is applied to argumentation frameworks with exactly one component and it characterizes a particular argumentation semantics.

[2] We will use the attribute "classiic" for admissible, preferred, stable, grounded and complete semantics.

A notion of SCC-recursive argumentation semantics formalizes the intuitions presented above. SCC-recursive characterization of traditional semantics is provided. Finally, some new semantics, AD1, AD2, CF1 and CF2, are defined in [3].

AD1 and AD2 extensions preserve the property of admissibility. However, the requirement of maximality is relaxed, so this solution is different as compared to the preferred semantics. An alternative is not to require admissibility of sets of arguments and insist only on conflict-freeness. Maximal conflict-free sets of arguments are selected as extensions in semantics CF1 and CF2. For details and differences see [3]. ASP-encodings of AD1, AD2, CF1 and CF2 are presented in [1].

4 Transfer of argumentation semantics

Basic ideas of our method are explained intuitively as follows. We build an *argumentation framework over the rules* of a logic program. Rules play the role of arguments. An attack relation over such arguments is introduced. After that some arguments (i.e., rules) are accepted/rejected on the basis of a given argumentation semantics: accepted rules form an extension of the argumentation semantics. A corresponding semantics for logic program is introduced as a set of literals *derivable* from accepted rules. Note that this method enables a transfer of an arbitrary argumentation semantics to the given logic program.

An argumentation framework over the rules of a program P is defined in the following. After that we will proceed towards derivations in P, based on an argumentation semantics.

Definition 2 *Let a program P be given. Then an* argumentation framework over P *is $AF_P = \langle AR, attacks \rangle$, where*

$AR = \{r \in P\}$ and $attacks = \{(r_1, r_2) \mid A = head(r_1), body^+(r_1) = \emptyset, A \in body^-(r_2)\}$. □

Example 2 *Let be $P = \{r_1 : a \leftarrow;\ r_2 : b \leftarrow not\ a.\}$. Then attacks relation of AF_P is $\{(r_1, r_2)\}$.*

If $P = \{r_1 : a \leftarrow not\ b.\ r_2 : b \leftarrow not\ a.\}$, then $attacks = \{(r_1, r_2), (r_2, r_1)\}$. □

Now, let us introduce the derivations. They are used for computing sets of atoms, which form a semantic counterpart of a selected argumentation semantics for a given logic programs.

Let a program P be given, AF_P be an argumentation framework over P. Suppose that a set of rules $R \subseteq P$ is an extension according to an argumentation semantics $\mathcal{S}$. Some intuitions in the next example.

Example 3 *Let a program $P = \{r_1 : a \leftarrow, r_2 : b \leftarrow not\ a, r_3 : c \leftarrow not\ b, r_4 : d \leftarrow not\ c\}$ be given.*

Our method consists of four steps. First, an argumentation framework AF_P over P is constructed. Second, a set of extensions of AF_P (i.e. a set of sets of rules of P) according to a chosen argumentation semantisc $\mathcal{S}$ is computed. Third, for each set of rules R is computed a set A of atoms, well-defined consequences of R. Finally, if R and A are consistent in a well-defined sense (our goal is to derive from R only atoms which do not occur as negated in the bodies of rules in R), then A represents a semantic counterpart of the chosen argumentation semantics for the logic program P.

Follow it on our example. Precise definitions are introuced after the example.

We get $AF_P = (\{r_1, r_2, r_3, r_4\}, \{(r_1, r_2), (r_2, r_3), (r_3, r_4)\})$. Consider the preferred semantics. The only preferred extension of AF_P is the set of rules $R = \{r_1, r_3\}$. The set of atoms $A = \{a, c\}$ can be derived from R. It is important to verify that R and A are consistent in the following sense: neither not a, nor not c occur in $body^-(R)$. Finally, if the consistency is verified, we decide that $A = \{a, c\}$ is derived *in P according to the preferred semantics and it represents a counterpart of preferred semantics of the logic program P.* □

We now proceed to formal definitions.

Definition 3 *A set of rules $R \subseteq P$ is* enabled *in a program P by an argumentation semantics S iff $R \in \mathcal{E}_S(AF_P)$. If R satisfies this condition, it is denoted by $Rule\_in_S^P$ (or by a shorthand $Rule\_in$, if a given semantics and a given program are clear from the context).* □

A set of rules R ($Rule\_in_S^P$) is enabled by S according to Definition 3, if R is an S-extension of AF_P. The following definition introduces an importatnt notion of a set of atoms consistent with a set of rules.

Definition 4 *Let M be an arbitrary set of atoms and $R \subseteq P$ be an arbitrary subset of a program P.*

It is said that M is consistent with *R iff $\forall A \in M\ \neg\exists r \in R\ A \in body^-(r)$.* □

Now, a fundamental task is to point out a way from rules enabled by an argumentation semantics to a corresponding set of atoms. The set of atoms, denoted by $In\_AS_S^P$ (or simply In_AS), is a candidate for a semantic characterization of the given logic program P.

Definition 5 *Let AF_P be an argumentation framework over a program P, $\mathcal{S}$ be an argumentation semantics of AF_P and $Rule\_in_S^P$ be a set of rules of P enabled by the semantics $\mathcal{S}$.*

Then $In\_AS_S^P$ is the least set of atoms $\{A \mid \exists r \in Rule\_in_S^P, head(r) = A, \forall b \in body^+(r) : b \in In\_AS_S^P\}$. □

Definition 5 specifies how to compute In_AS. First, $head(r)$ is included into In_AS for each $r \in Rule\_in_S^P$ s.t. $body^+(r) = \emptyset$. After that In_AS is iteratively recomputed for all $r \in Rule\_in_S^P$ with non-empty $body^+(r)$. Notice that this is

a process of $T_{Rule\_in_S^P}$ iteration applied to a set of assumptions (a set of default literals).

Finally, it is necessary to use consistent $In\_AS_S^P$ in order to define a sound semantic characterization of the given logic program P. This characterization is called the set of atoms *derived* in P according to a semantics $\mathcal{S}$.

Definition 6 *If $In\_AS_S^P$ is consistent with $Rule\_in_S^P$, then it is said that $In\_AS_S^P$ is the set of atoms* derived *in P according to semantics $\mathcal{S}$.* □

Example 4 *Remind Example 3. We will analyze it in terms of Definitions 3 – 6.*

$P = \{r_1 : a \leftarrow, r_2 : b \leftarrow not\ a, r_3 : c \leftarrow not\ b, r_4 : d \leftarrow not\ c\}$.

$AF_P = (\{r_1, r_2, r_3, r_4\}, \{(r_1, r_2), (r_2, r_3), (r_3, r_4)\})$.

$\mathcal{E}_S(AF_P) = \{\{r_1, r_3\}\}$*, where S is the preferred semantics. It means, $\{r_1, r_3\}$ is the only set of rules, enabled by the preferred semantics according to Definition 3.*

$In\_AS = \{a, c\}$ *according to Definition 5. The set of atoms $\{a, c\}$ is consistent with the set of rules $\{r_1, r_3\}$ according to the Definition 4. Finally, according to Definition 6 is $\{a, c\}$ derived in P according to the preferred semantics.* □

The next example shows why the condition of consistency is important.

Example 5 *Let P be $\{r1 : a \leftarrow b, r2 : b \leftarrow not\ a\}$. Then $AF_P = (\{a, b\}, \emptyset)$, preferred extension of AF_P is P and the set of consequences of P according to Definition 5 is $In\_AS = \{a, b\}$. But In_AS is inconsistent with P.*

It is counterintuitive to consider $\{a, b\}$ as a semnatic characterization of P. The drawbacks of the presented set of definitions are analyzed below. □

Derivation of atoms according to Definition 6 coincides with the definition of derivation in Section 2.

Proposition 1 *Let an argumentation semantics S be given. Let be $R = Rule\_in_S$. A set of atoms derived in P according to the semantics S is $\Delta^{\sim_R}$ for some Δ.*

Proof:
Let be $R = Rule\_in_S$ and In_AS be the corresponding derived set of atoms.

Suppose that $\Delta = \{not\ A \mid \exists r \in R\ A \in body^-(r)\}$. It holds that $B \in \Delta^{\sim_R}$ iff $R^+_\Delta \models B$. Obviously, $R^+_\Delta \models B$ holds iff $B \in In\_AS$. □

The attacking argument in the attack relation is constrained in this section to the rules with non-empty $body^+(r)$. An advantage of this decision is that the attack relation is recognizable on syntactic level. No additional computation is needed: attacks of rules with non-empty $body^+(r)$ are conditional, they depend on a derivation of positive literals in $body^+(r)$. On the other hand, this decision does not consider hidden attacks and leads to some counterintuitive consequences, see the next example. The bug is improved in the next section.

Example 6 *Let P be $\{r_1 : a \leftarrow not\ b,\ r_2 : b \leftarrow c, not\ d.\},\ r_3 : c \leftarrow .\}$, then in AF_P is attacks $= \emptyset$. If S is the preferred semantics, then $\{\{r_1, r_2, r_3\}\} = \mathcal{E}_S(AF_P)$, $P = Rule\_in_S^P$ is enabled by the preferred semantics.*

Further, it holds that $In\_AS_S^P = \{a, b, c\}$ according to Definition 5. But $In\_AS_S^P$ is not consistent with $P = Rule\_in_{\mathcal{S}}$, hence no atom is derived in P according to the preferred semantics.

Obviously, $\{r_2, r_3\}$ could be an intuitive preferred extension. It means that our construction (our decision about the attacks relation) do not generate all intuitive semantic characterizations of a logic program corresponding to an argumentation semantics. □

Examples 5 and 6 show that there are logic programs without a semantic counterpart of a preferred extension of argumentation framework over those programs.This is a feature which does not correspond to the fact that every argumentation framework has a preferred extension.

Similalry, note that stable extensions of AF_P are not in general stable models of P.

Example 7 *Consider the program $P = \{r_1 : a \leftarrow p, not\ b, r_2 : b \leftarrow q, not\ a, r_3 : p \leftarrow\}$.*

The stable model of P is $\{p, a\}$, but a semantic counterpart of the stable extension of AF_P does not exist. Rules $P = \{r_1, r_2, r_3\}$ form a stable extension of AF_P, but $In\_AS = \{a, b, p\}$ is not consistent with $Rule\_in = P$, □

This observation is a consequence of the given design decision concerning the attack relation. We are going to solve the problem in the next section.

5 Transfer via unfolding

The approach described in the previous section is insensitive w.r.t. hidden attacks between rules. Examples 6 and 7 indicate that this is a bug, if we want an intuitively satisfying transfer of an argumentation semantics to a corresponding logic program semantics. Hidden attacks of rules with non-empty positive parts of bodies can be uncovered by the unfolding operation, see the next example.[3]

Example 8 *Remind Example 6 with $P = \{r_1 : a \leftarrow not\ b,\ r_2 : b \leftarrow c, not\ d.\ r_3 : c \leftarrow .\}$. A buggy behaviour of the approach of the previous section was illustrated on AF_P.*

Unfolded form (equivalent) of P is the program $Q = \{q_1 : a \leftarrow not\ b;\ q_2 : b \leftarrow not\ d;\ q_3 : c \leftarrow\}$,

Consider the argumentation framework over Q, $AF_Q = (\{q_1, q_2, q_3\}, \{(q_2, q_1)\})$. Hence, the (only) preferred extension of AF_Q is $R = \{q_2, q_3\}$. The (only) stable model of R is $\{b, c\}$.

[3] We are inspired by a use of unfolding in [13,19]. Unffolding is used there for a specification of preferred answer sets.

We are aiming to propose $\{b, c\}$ as the semantic counterpart of the preferred semantics in P. To that end we continue with some basic facts about unfolding and then proceed to our construction and its evaluation. □

We will use the term *unfolding* for a transformation method based on the Principle of Generalized Partial Evaluation (GPPE) [8].

Definition 7 (Unfolding) *Let a program P and an atom A be given. Let Q be the set of all rules $q_j \in P, j = 1, \ldots, n$ s.t. $A \in body^+(q_j)$ for each j. Consider the set of all rules $r_i, i = 1, \ldots, k$ s.t. $head(r_i) = A$.*

Then we define the program $P' = (P \setminus Q) \cup \{head(q_j) \leftarrow \mathcal{B}_i^j, (body^-(r_i) \cup body^-(q_j)) \mid i \in \{1, \ldots, k\}, j \in \{1, \ldots, n\}$, where $\mathcal{B}_i^j = (body^+(q_j) \setminus A) \cup body^+(r_i)\}$.

It is said that P' is the result of an unfolding transformation *of P.* □

Definition 8 (Unfolding sequence, unfolded program) *If $\sigma = \langle P_1, \ldots, P_m \rangle$ is a sequence of logic programs s.t. for each $i < m$ holds that P_{i+1} is the result of unfolding transformation of P_i, then σ is called an* unfolding sequence.

If P is a program s.t. there is no unfolding sequence of length greater than 1 with P as the first member, it is said that P is an unfolded program. *If Q is an unfolded program and there is an unfolding sequence $\langle P, \ldots, Q \rangle$, then Q is an* unfolded form *of P.* □

Let $SM(Q)$ be the set of all stable models of a program Q, If P' is an unfolded form of P then $SM(P) = SM(P')$, see [8].[4]

Remark 1 *Unfolded programs may contain tautological rules and rules with such atonms in positive parts of bodies, which do not occur in heads of the program. If we use another notion of unfolded program, reduced by some elimination principles of [5] (compare with loop detection of [9]), we may get a more efficient computation of an unfolded equivalent of a given program. However,this is not in focus of our paper.* □

We propose a construction as follows. A starting point is a logic program P and our goal is to transfer an arbitrary argumentation semantics to P.

To this end P is transformed to its unfolded form Q and an argumentation framework over Q is constructed. Definition 2 of attack relation is used also here. If a rule with non-empty positive part of its body is in an unfolded program, it is either a tautological rule or a rule with false body. Those rules are not executable and it is not intuitive consider them as attacking rules. Hence, we may suppose that an attacking rule has the empty positive body.

Similarly, we use without a change also Definition 3 of rules enabled by an argumentation semantics. A subset $R \subseteq Q$ of rules enabled by an argumentation

[4] Equivalences w.r.t. to other semantics hold, too, but we are interested only in stable model semantics, see Definition 9.

semantics $\mathcal{S}$, i.e., an extension of AF_Q w.r.t. $\mathcal{S}$ is selected and a stabble model M of R is computed (we will show that there is only one such stable model).

Finally, a semantic counterpart of $\mathcal{S}$ for P is built over M.

Let us start with some examples motivating some hints for a specification of a semantic counterpart of an argumentation semantics for a logic program. After that follows a formal elaboration.

Example 9 *Remind Example 7:* $P = \{r_1 : a \leftarrow p, not\ b, r_2 : b \leftarrow q, not\ a, r_3 : p \leftarrow\}$. *It was shown that the method of Section 4 does not guarantee corersponedence of the stable model of* P *and the stable extensions of* AF_P.

Consider Q, *the unfolded form of* P: $\{q_1 : a \leftarrow not\ b; q_2 : b \leftarrow q, not\ a; q_3 : p \leftarrow\}$. *The only stable extension of* AF_Q *is* $R = \{q_1, q_3\}$. *The only stable model of* R *is* $S = \{not\ b\} \cup \{a, p\}$, *where* $\{not\ b\}^{\curvearrowright_R} = \{a, p\}$. *Observe that* $\{q\} = body^+(P)$. *It holds that atoms from positive parts of bodies of an unfolded program are false in each stable model of the program. Hence, we can assume* $not\ q$ *and* $\{not\ b, not\ q\} \cup \{not\ b, snotq\}^{\curvearrowright_R}$ *is (the only) stable model of* P. □

There is yet further reason for adding default literals into a semantic counterpart of an extension.

Example 10 *Let* P *be an unfolded program* $\{r_1 : a \leftarrow not\ b;\ r_2 : b \leftarrow not\ a;\ r_3 : c \leftarrow not\ b;\ r_4 : c \leftarrow not\ a;\ r_5 : d \leftarrow not\ c\}$.

Stable extensions of AF_P *are* $\{R = \{r_1, r_3\}, R' = \{r_2, r_4\}$ *and stable models of* R *and* R' *are* $\{not\ b\} \cup \{a, c\}$ *nd* $\{not\ a\} \cup \{b, c\}$. *respectively, where* $\{a, c\} = \{not\ b\}^{\curvearrowright_R}$ *and* $\{b, c\} = \{not\ a\}^{\curvearrowright_{R'}}$.

Consider now r_5: d *is false both in* R *and* R'. *If a goal is to transfer stable models of* R (R') *to stable models of* P *it is needed to assume also not* d. □

Examples 9 and 10 illustrated that a completion is required, if we want to get a stable model of P, because we are aiming at a total interpretation. On the other hand, some argumentation semantics do not require two-valued counterparts on the side of logic programs.

Another useful hint is motivated by the next example. An argumentation framework without stable extension, but with preferred extension is presented. We need to accept a three-valued point of view, when constructing a countepart of this semantics for logic programs.[5]

Example 11 *Let* P *be an unfolded program* $\{r = p \leftarrow not\ p\}$. AF_P *is* $(\{r\}, \{(r, r)\})$. *The preferred extension of* AF_P *is* $\emptyset \subseteq P$, *the stable model of the empty program is empty. If we want to suggest a semantic characterization of* P, *which is a counterpart of the preferred extension of* AF_P, *we have to consider 3 valued interpretations, in order to avoid an assignment of true to not* p. □

It is useful to show now that a set of rules enabled by an argumentation semantics has only one stable model.

[5] Semantic counterparts of "classic" argumetation semantics were charcterized already in [10,11,4].

Proposition 2 *Let $R \subseteq Q$ be a set of rules enabled by an argumentation semantics $\mathcal{S}$. Then R has the only stable model.* □

Proof: Suppose that $R \in \mathcal{E}_S(AF_Q)$. Only conflict-free semantics are assumed, i.e. $head(R) \cap body^-(R) = \emptyset$. Suppose for simplicity that tautological rules and rules with atoms in their bodies, which are not defined by a rule are (equivalently) eliminated.

Let Δ be $body^-(R)$. Then $\Delta^{\backsim_R} = \{head(r) \mid r \in P, body^+(r) = \emptyset\}$. Therefore, $M = \Delta \cup \Delta^{\backsim_R}$ is a total interpretation in the language of R, each atom of R occcurs in M either in positive form or negated. It is also the only stable model of R, because there is no different set of assumptions Δ' generating via R a total interpretation.

If rules with atoms in its bodies are not eliminated, Δ contains also default negations *not* A s.t. A occurs in a body of a rule, but it does not occur in a head of a rule with the empty positive body. □

Our examples showed that a careful stance is needed when argumentation semantics $\mathcal{S}$ is transformed into a semantic counterpart of $\mathcal{S}$ for the given logic program P. We will use a function $\alpha(S, P)$ which assigns a set of default negations to a given argumentation semantics S and logic program S. For the goals of this paper α assigns an empty set of default negations to each argumentation semantics except of stable semantics (or except of each two valued semantics). In that case α adds to default negations of the stable model of R default negations of all atoms occuring in P, but not in R.

Definition 9 (Counterpart) *Let P be a program, Q its unfolded form, AF_Q be an argumentation framework over Q and $\mathcal{S}$ be an argumentation semantics.*

Suppose that $R \subseteq Q$ is a set of rules enabled by $\mathcal{S}$ and $\Delta \cup \Delta^{\backsim_R}$ is the stable model of R. We assume also a function $\alpha(S, P)$, which assigns a set of default literals to an argumentation semantics $\mathcal{S}$ and program P.

Then $(\Delta \cup \alpha(S, P)) \cup \Delta^{\backsim_R}$ is called a semantic counterpart *of $\mathcal{S}$ for the program P.* □

In Eexample 9 the stable model of the stable extension R was also a stable model of P. We will show that in general the semantic counterpart of a stable extension of AF_Q is a stable model of P, if Q is an unfolded form of P.

Let $At(P)$ be the set of atoms occuring in P.

Proposition 3 *Let P be a logic program, Q its unfolded form, $\mathcal{S}$ be the stable argumentation semantics and $R \subseteq Q, R \in \mathcal{E}_S(AF_Q)$ be a stable extension of AF_Q.*

If $\Delta \cup \Delta^{\backsim_R}$ is the stable model of R, then $(\Delta \cup \alpha(S, P)) \cup \Delta^{\backsim_R}$ is a stable model of P, where $\alpha(S, P)$ is $\{not\ A \mid\ A \in At(Q \setminus R)\}$, □

Proof: Let $R \subseteq Q$ be a stable extension of AF_Q, i.e. $R = \{r \in Q \mid r \text{ is not attacked by } R\}$. Further, let M be the stable model of R, i.e. $M = \Delta \cup \Delta^{\backsim_R}$ is for some Δ a total interpretation of R.

We will show that for some set of assumptions Θ is $\Delta \cup \Theta \cup \Delta^{\sim_R}$ a stable model of P. Let Θ be $\{not\ A \mid\ A \in At(Q \setminus R)\}$.

Consider $q \in Q \setminus R$, i.e. there is $r \in R$ s.t. $not\ head(r) \in body^-(q)$. Hence, rules of $Q \setminus R$ are not applicable w.r.t. $Q \cup \Delta$, all heads of rules from $Q \setminus R$ have to be considered as false in any superset of M. Similarly for each atom $A \in At(Q \setminus R)$ s.t. $A \notin head(Q \setminus R)$.

It follows that $(\Delta \cup \Theta) \cup \Delta^{\sim_R}$ is a stable model of Q, consequently aalso a stable mlodel of P. □

Semantic counterparts of some argumentation semantics are discussed briefly in the next section.

6 Discussion

Main goal of this section is to provide a brief characterization of some logic program semantics, transferred from the "classic" argumentation semantics and to oultine the topic of semantic characterization of logic programs with odd-length cycles through default negations. A more detailed characterization is a goal of our future research. In this paper the counterparts to admissible, preferred, grounded and complete argmentation semantucs are provided indirectly, via a mapping to notions and results of [4].

Semantic counterparts of admissible, preferred, stable, grounded, complete argumenation semantics were studied already in [10,**?**,4]. A correspondence between stable (well founded) models of a logic program P and stable (grounded) extensions of the argumentation framework $AF_{napif}(P)$, assigned to P is shown in [10]. Dung in [11] proposed some new semantics of logic programs. The semantics were inspired by (and correspond to) "classic" argumentation semantics. Here is a list of semantics pairs, the first member of the pair is an argumentation semantics, the second is the corresponding logic program semanics: admissible set – admissible scenario [11]; preferred extension : preferred extension [11]; complete extension : complete scenario [11]; grounded extension : well founded model [10].

Results of [4] concerning relations between logic program sematics and argumentation semantics are dependent of [11]. However, the first part of this section presents the construction of [4]. The reason is that there is a simple assignment of our conceptual apparatus to notions of [4] and, consequently, the results of [4], i.e., the resilts of [11] can be inherited also by our framework.

Our goal is to express analogues of Proposition 3 for admissible, preferred, grounded and complete extensions, i.e., to show for a logic program P and its unfoldded form Q that if $R \subseteq Q$ is an admissible (preferred, grounded, complete) set of arguments in AF_Q then the stable model of R specifies its semantic counterpart for the logic program P.

Our construction is mapped onto the construction of [4] and it is briefly argued that their results about semantic counterparts of argumentation semantics hold also for our approach.

An argumentation-theoretic framework over a deductive system is used in [4] for a characterization of different nonmonotonic semantics. Basic argumentation-theoretic semantics are defined for assumption-based frameworks and a set of nonmonotonic semantics is characterized in terms of argumentation semantics.

We introduce here only the specialization of the abstract argumentation-theoretic framework over a deductive system for logic programs, which was constructed in [4].

Deductive system corresponding to a logic program P is a pair $(\mathcal{V}, \mathcal{R})$, where $\mathcal{V}$ consists of the set of all literals $\mathcal{L}$ and of all rules, which can be constructed in terms of $\mathcal{L}$. $\mathcal{R}$ is the set of all inference rules with premisses created by a rule and all literals in its body, while the consequent of the given rule is the head of the rule in the premisses.

An argumentation-theoretic framework over such deductive system is a triple $(P, \mathcal{H}, c)$, where $\mathcal{H} = 2^{\mathcal{D}}$ is the set of all sets of assumptions constructible in the given language and c is a mapping, which assigns atom A to a default negation $not\ A$.

Let T be $P \cup \Delta$, where Δ is a set of assumptions. A deduction from T is a sequence $\sigma = \langle \beta_1, \ldots, \beta_k \rangle$, where $k > 0$, $\beta_i \in T$ or it is a consequence of an inference rule with all premisses occuring before β_i in σ. It is denoted by $T \vdash L$ that there is a deduction of a literal L from T.

Δ is a conflict-free set of assumptions iff $P \cup \Delta \nvdash A, not\ A$. A maximal conflict-free asuumption is defined in a sraighforward way. A set of assumtions Δ attacks an assumption $not\ A$ iff $P \cup \Delta \vdash A$ and Δ attacks a set of assumptions Δ' iff it attacks some $not\ A \in \Delta'$.

The proofs of results similar to Proposition 3 are based on the following observation.

If R is an admissible set of arguments in our approach (i.e., a set of rules) and $\Delta = body^{-}(R)$ then Δ is an admissible set of assumptions: it does not attack itself and if some Δ' attacks Δ then Δ atacks Δ'. Further, $A \in \Delta^{\curvearrowright_R}$ iff $R \cup \Delta \vdash A$ and our attack relation of AF_P is equivalent to the attack of [4], which was defined above. Therefore, also conditions for preferred, grounded and complete semantics of AF_Q are inherited by stable models of R, i.e. by a semantics of P.

A transfer of results gained in [4] to an identification of semantic counterparts of argumentation-theoretic semantics of AF_P for a logic program$P$ is based on the presented observation. A detailed exposition of this result will be presented in a next paper.

In the second part of this section the topic of odd cycles and of transferring argumentation semantics of [3] to logic program semantics is discussed.

Programs with ood-length cycles throuigh default negation do not have stable models. However, there is a reserach devoted to attempts to priovide a reasonable semantic characterization of such programs. We do not refer to that research, our goal is only to outline how our method enables a trnsfer of semantics proposed in [3].

Our method enables a transfer of an arbitrary argumentation semantics to a logic program semantics, consequently, it can serve as a tool for transferring semantics AD1,AD2, CF1 and CF2 [3] and, thus, to rpovide a semantic characterization of odd cycles.

First, an example, which shows that "classic" argumentation semantics have some the problems with odd cycles.

Example 12 *Consider program* $P_1 = \{r_1 : a \leftarrow not\ b, r_2 : b \leftarrow not\ a\}$ *with a negative cycle of even length and* $P_2 = \{r_1 : a \leftarrow not\ b, r_2 : b \leftarrow not\ c, r_3 : c \leftarrow not\ a\}$ *with an odd length negative cycle.*

There is no stable model of P_2*, but* P_1 *has two stable mdels. Preferred, stable and complete argumentation semantics assign also two extensions to* AF_{P_1}*. On the other hand, they assign one (empty) or no extension to* AF_{P_2}*. This assymetry is considered in [3] as a drawback.* □

Recursive semantics of [3] are aiming to overcome that drawback. An interesting solution of the problem of odd cycles may provide a new perspective on semantic characterization of logic programs, too.

Example 13 *Let us continue with Example 12.*

Note that AF_{P_2} *consists of the only component, the odd cycle* $(r_1, r_2), (r_2, r_3), (r_3, r_1)$*.* $CF1$ *assigns three extensions* $\{r_1\}, \{r_2\}, \{r_3\}$ *to* AF_{P_2}*. Connsequently, our construction enables to transfer three sets of atoms* $\{a\}, \{b\}, \{c\}$ *as a semantic characterizations of the logic program* P_2*.*

Observe that no one of those characterizations is a two-valued model of P_2*. However, a three-valued or paraconsistent model is possible.* □

Consider also other example.

Example 14 *Let be* $P = \{r_1 : a \leftarrow not\ a, r_2 : b \leftarrow not\ a\}$*. The argumentation framework* AF_P *has according to the semantics* $CF2$ *extension* $\{r_2\}$*, consequently* $\{b\}$ *is transferred to* P*.*

A three-valued or paraconsistent interpretation is needed, too. □

Our futurre goal is to investigate more deeply semantics trasferred from AD1, AD2, CF1, CF2 or other argumentation semantics to logic program semantics.

7 Representation result

In this section we apply a changed view. An argumentation framework $\mathcal{A}$ is assumed and its representation by a simple logic program $P^{\mathcal{A}}$ is constructed. Then we can construct an argumentation framework $\mathcal{B} = AF_{P^{\mathcal{A}}}$ over the rules of $P^{\mathcal{A}}$ using our method.

We will present a kind of representation result – an argumentation semantics of $\mathcal{A}$ is preserved under transformations to $\mathcal{B}$ via $P^{\mathcal{A}}$. Suppose that an argumentation framework $\mathcal{A}$ is given. Conisder its extension E under the argumentation

semantics $\mathcal{S}$. We will transfer $\mathcal{A}$ to a logic program $P^{\mathcal{A}}$. If the argumentation framework $AF_{P^{\mathcal{A}}}$ is constructed then the semantic counterpart of an extension of $AF_{P^{\mathcal{A}}}$ under $\mathcal{S}$ in $P^{\mathcal{A}}$ is again E.

Definition 10 *Let an argumentation framework $AF = \langle AR, attacks\rangle$ be given. We represent AF by a logic program P^{AF} as follows*

- *for each $a \in AR$ there is exactly one rule $r \in P^{AF}$ s.t. $head(r) = \{a\}$*
- *$body^-(r) = \{b \mid b \in AR, (b,a) \in attacks\}$, $body^+(r) = \emptyset$.*

□

If body of a rule is empty, then the corresponding argument is not attacked in AF. A selection of a conflict-free $R \subseteq P^{AF}$ starts from the facts and subsequently are considered only those attacking arguments (negative literals in bodies of P^{AF}), which do not occur as heads in the rules already included in R. Additional conditions on R are specified by an argumentation semantics $\mathcal{S}$.

Example 15 *Let be $AF = (AR, attacks)$, where $AR = \{a,b,c,d,e\}$ and $attacks = \{(a,b),(c,b),(c,d),(d,c),(d,e),(e,e)\}$. P^{AF}, the logic program representing AF is as follows:*

$$
\begin{aligned}
r_1 &: \; b \leftarrow not\ a, not\ c \\
r_2 &: \; a \leftarrow \\
r_3 &: \; c \leftarrow not\ d \\
r_4 &: \; d \leftarrow not\ c \\
r_5 &: \; e \leftarrow not\ e, not\ d
\end{aligned}
$$

□

Programs representing an argumentation framework look like lists: to each argument in the head of a rule is assigned a list of arguments attacking the argument in the head of the rule.

Notice that there are logic programs, which cannot represent an argumentation framework. On the other hand, if a logic program represents an argumentation framework, it is done in a unique way – there is exactly one argumentation framework represented by the program.

Example 16 *$P_1 = \{a \leftarrow not\ b, b \leftarrow not\ a\}$ is a logic program, which represents the argumentation framework $AF = \langle\{a,b\}, \{(a,b),(b,a)\}\rangle$.*

$P_2 = \{a \leftarrow not\ b\}$ cannot be a representation of any argumentation framework. There is no rule in P_2 with b in its head (and each argument must be in the head of a rule).

Theorem 4 *Let AF be an argumentation framework, $AF = (AR, attacks)$, P^{AF} be the logic program representing AF. Let In_AS be a set of atoms, derivable in P^{AF} according to a semantics S.*

Then In_AS is an extension of AF according to the semantics S.

Proof:
For each argument $a \in AR$, there is exactly one rule $r \in P^{AF}$ s.t. $head(r) = a$. A function $\Psi : R \to AR$, where $R \subseteq P^{AF}$, assigns to each rule $r \in R$ the argument $a \in AR$, which occurs in the head of r. $\Psi^{-1} : AR \to R$ is an inverse function which assigns to an argument the rule with the argument in the head.

$In\_AS = \{a \mid \exists r \in Rule\_in, head(r) = a\}$ follows from the fact that $body^+(r) = \emptyset$ for each rule r. Hence, $In\_AS = \Psi(Rule\_in)$.

Let us denote P^{AF} simply by P. It follows from the definition that for each $(a, b) \in attacks$ there is a pair $(r_1, r_2) \in attacks_P$, where $AF_P = \langle AR_P, attacks_P \rangle$. Notice that $a \in head(r_1)$ and $b \in head(r_2)$. (AF_P is a framework over the rules of the program P). If $(a, b) \in attacks$ then *not* a occurs in the body of a rule with b in the head. Similarly, for all $(r_1, r_2) \in attacks_P$ there is $(x, y) \in AF$ s.t. $head(r_1) = x, y \in body^-(r_2)$. Therefore, the only difference between the frameworks AF and AF_P is that the vertices of both frameworks are renamed according to the function Ψ.

Therefore, $In\_AS = \Psi(Rule\_in) = \mathcal{E}_S(AF)$. □

8 Related work

First, we compare from the computational point of view arguments of our argumentation framework AF_P to arguments of [10] and the attack relation of AF_P to the attack relation of [4].

In [10] arguments assigned to a logic program are pairs (Δ, k), where $k \in \Delta^{\sim_P}$ is a ground atom or pairs of the form $(\{notk\}, notk)$. In the general case of first-order language the set of arguments is undecidable. In the case of a propositional language, the corresponding decision problem is computationally demanding. On the other hand, arguments of AF_P are rules of P.

Similarly, the decision problem concerning the attack relation of [4] is computationally more demanding than ours, where only pairs of rules are compared.

Of course, an extra computational cost of a transformation to an unfolded progarm is needed except of the relatively simple identification of arguments and attacks in AF_P. On the other hand, if we replace GPPE by loop detection, see [9], we can get the transformation in polynomial time. Additionally, as regards the computational complexity of our approach, if a set of rules R is an extension of an argumenttion framework AF_Q and Q is an unfolded program with eliminated atoms in bodies, then R is stratified and only two levels are necessery for its stratification. Computational aspects of our approach will be analyzed more precisely in a next paper.

Let us proceed to other related works. We are familiar with the following types of results.

– Some researchers construct a new semantics of logic programs, inspired by extensions of argumentation frameworks. This goal is close to ours.
– A correspondence of an argumentation semantics and a logic program semantics is described. Particularly, they specify a characterization of extensions of abstract argumentation framework in terms of answer sets or other semantics of logic programs.
– Finally, encoding extensions of argumentation frameworks in answer set programming is another type of research. This research usually leads to an implementation of argumentation semantics.

Argumentation framework is constructed and studied in terms of logic programs in [18]. Arguments are expressed in a logic programming language, conflicts between arguments are decided with the help of priorities on rules.

The correspondence between complete extensions in abstract argumentation and 3-valued stable models in logic programming was studied in [20].

The project "New Methods for Analyzing, Comparing, and Solving Argumentation Problems", see, e.g., [12,14], is focused on implementations of argumentation frameworks in Answer-Set Programming, but also other fundamental theoretical questions are solved. CF2 semantics is studied, too. An Answer Set Programming Argumentation Reasoning Tool (ASPARTIX) is evolved.

The Mexican group [6,15,16,17] contributes to research on relations of logic programing and argumentation frameworks, too. Their attention is devoted to characterizations of argumentation semantics' in terms of logic programming semantics. Also a characterization of CF2 is provided in terms of answer set models or stratified argumentation semantics, which is based on stratified minimal models of logic programs.

Our main goal, in the context of presented remarks, is to "import" semantics from argumentation frameworks to logic programs. However, also other results about relations of both areas are relevant for us.

This section contains only some sketchy remarks, a more detailed analysis and comparison is planned.

9 Conclusions

A method for transferring an arbitrary argumentation semantics to a logic program semantics was developed. The method consists in defining an argumentation framework over the rules of a program. Extensions of the argumentation framework are sets of rules. A set of consequences of those rules is an interpretation, which provides the corresponding semantic characterization of the program.

This method allows a semantic characterization of programs with odd-length (negative) cycles.

Acknowledgements: This paper was supported by the grant 1/1333/12 of VEGA.

References

1. M. Adamová: Representácia abstraktného argumentačného frameworku logickým programom; Master Thesis, Comenius University, 2011
2. M. Adamová, J. Šefránek: Transfer of semantics from argumentation frameworks to logic programming. A preliminary report. Proceedings of WLP/INAP 2011.
3. Baroni, P., Giacomin, M., Guida, G.: SCC- recursiveness: a general schema for argumentation semantics. Artificial Intelligence, 168 (1-2), 2005, 162-210
4. A. Bondarenko, P.M. Dung, R.A. Kowalski, F. Toni: An abstract, argumentation-theoretic approach to default reasoning. Artif. Intell. 93: 63-101 (1997)
5. S. Brass, J.Dix: Characterizations of the Stable Semantics by Partial Evaluation. Journal of Logic Programming, 32(3), 207-228, 1997
6. J. L. Carballido, J. C. Nieves, and M. Osorio.: Inferring Preferred Extensions by Pstable Semantics. Iberoamerican Journal of Artificial Intelligence, 13(41):3853, 2009
7. Y. Dimopoulos, A. Torres: Graph theoretical structures in logic programs and default theories, Theoretical Computer Science 170, pages 209-244, 1996.
8. J.Dix: A Classification Theory of Semantics of Normal Logic Programs: II. Weak Properties. Fundamenta Informaticae, 22 (3): 257-288 (1995)
9. J. Dix, U.Fuhrbach, I. Niemelä: Nonmonotonic Reasoning: Towards Efficient Calculi and Implemenations. Handbook of Automated Reasoning, 2001
10. P. M. Dung: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artificial Intelligence 77, pages 321-357, 1995.
11. Dung, P.M.: An Argumentation Theoretic Foundation of Logic Programming. Journal of Logic Programming, vol. 22, No. 2, 1995, 151-177
12. Eggly, U., Gaggl, A., Woltran, S.: ASPARTIX: Implementing Argumentation Frameworks Using Answer-Set Programming. Proc. of the 24th International Conference on Logic Programming (ICLP 2008), 734-738. Springer LNCS 5366, 2008.
13. A. Gabaldon: A Selective Semantics for Logic Programs with Preferences, Proc. of NRAC, 2011.
14. S. A. Gaggl, S. Woltran: Cf2 Semantics Revisited. Frontiers in Artificial Intelligence and Applications, pages 243-254. IOS Press, 2010.
15. J. C. Nieves, M. Osorio, and C. Zepeda. Expressing Extension-Based Semantics based on Stratified Minimal Models. In H. Ono, M. Kanazawa, and R. de Queiroz, editors, Proc. of WoLLIC 2009, Tokyo, Japan, 305-319. Springer Verlag, 2009.
16. J. C, Nieves and I. Gomez-Sebastia: Extension-Based Argumentation Semantics via Logic Programming Semantics with Negation as Failure. Proc. of the Latin-American Workshop on Non-Monotonic Reasoning, CEUR Workshop Proceedings vol 533, 31-45, Apizaco, Mexico, November 5-6, 2009.
17. Osorio, M., Nieves, J.C., Gmez-Sebastia, I.: CF2-extensions as Answer-set Models. Proceedings of the COMMA2010 Conference. Pages 391-402.
18. H. Prakken, G. Sartor: Argument-based logic programming with defeasible priorities. Journal of Applied Non-classical Logics 7: 25-75 (1997), special issue on 'Handling inconsistency in knowledge systems'.
19. J. Šefránek and A.Šimko: Warranted derivation of preferred answer sets, Proc.WLP/INAP 2011.
20. Wu, Y., Caminada, M., Gabbay, D.: Complete Extensions in Argumentation Coincide with Three-Valued Stable Models in Logic Programming. Studia Logica 93(2-3):383-403 (2009)