Argumentation-based Proof for an Argument in a Paraconsistent Setting ⋆

Iara Carnevale de Almeida[2] and José Júlio Alferes[1]

[1] CENTRIA, Universidade Nova de Lisboa
2829-516 Caparica, Portugal
jja@di.fct.unl.pt
[2] CITI, Departamento de Informática, Universidade de Évora
Colégio Luis Verney; 7000-671 Évora, Portugal
ica@di.uevora.pt

Abstract. The paradigm of argumentation has been used in the literature to assign meaning to knowledge bases in general, and logic programs in particular. With this paradigm, rules of logic program are viewed as encoding arguments of an agent, and the meaning of the program is determined by those arguments that somehow (depending on the specific semantics) can defend themselves from the attacks of others arguments, named acceptable arguments. In previous work we presented an argumentation based declarative semantics allowing paraconsistent reasoning and also dealing with sets of logic programs that argue and cooperate among each other. In this paper we focus on the properties of this semantics in what regards paraconsistency and propose a procedure for proving an argument according to that semantics.

1 Introduction

In logic programming, several ways to formalise argumentation-based semantics have been studied for logic programs. Intuitively, argumentation-based semantics treat the evaluation of a logic program as an argumentation process, i.e. a goal G is true if at least one argument for G cannot be successfully attacked. The ability to view logic programming as a non-monotonic knowledge representation language, in equal standing with other non-monotonic logics, brought to light the importance of defining clear declarative semantics for logic programs, for which proof procedures (and attending implementations) are then defined (e.g. [8, 9, 15, 2, 20, 12, 18, 7, 14, 10]).

In [5] we proposed an argumentation based semantics for sets of logic programs that are able to cooperate and argue with each other. In it each program relies on a set of other programs with which it has to agree in order to accept an argument, and a set of programs with which it can cooperate to build arguments. Besides this distributed nature, the semantics in [5] also allows for paraconsistent forms of argumentation. In fact, it was also a goal of that work to be able to deal with mutually inconsistent, and even inconsistent, knowledge bases. Moreover, when in presence of contradiction we

⋆ The work was partially supported by the Brazilian CAPES, and by the European Commission within the 6th Framework Programme project REWERSE, number 506779.

wanted to obtain ways of agent reasoning, ranging from consistent (in which inconsistencies lead to no result) to paraconsistent. For achieving this, we considered strong and weak arguments.

The paraconsistency in the argumentation also yield a refinement of the possible status of arguments: besides the justified, overruled and defensible arguments as in [16], justified arguments may now be contradictory, based on contradiction or non-contradictory. Moreover, in some applications it might be interesting to change easily from a paraconsistent to a consistent way of reasoning (or vice-verse).

In this paper we focus on the properties of that semantics in what regards paraconsistency which are interesting by themselves, and independent from its distributed nature. With this purpose, we restrict our attention to the special case of the semantics in [5], where only a single logic programs is in the set of programs. Moreover, we provide a notion of proof for an argument for that semantics in that class.

In the next section we present a version of the proposed declarative semantics simplified for the case of a single program, study some of its most significant properties regarding paraconsistency, and illustrate it in one example. We then define the proof method for it, and end with some conclusions. Due to lack of space all proofs have been removed from this version of the paper, and they can be found in a longer version available as a technical report from the first author.

2 Paraconsistent Argumentation Semantics

As motivated in the introduction, in our framework [5] the knowledge base of an agent is modelled by a logic program. More precisely, we use *Extended Logic Program with denials* (ELPd), itself an extension of Extended Logic Programs [11] for modelling the knowledge bases. Besides default and explicit negation, as usual in extended logic programs, we allow a program to have denials of the form

$$\bot \leftarrow L_1, \ldots, L_l, not\ L_{l+1}, \ldots, not\ L_n\ (0 \leq l \leq n)$$

where each of the L_is is an objective literal (i.e. an atom A in the language of the program, or an explicitly negated atom $\neg A$). In other words, denial are simply rules where the head is the special, reserved, symbol $\bot$.

An argument for some objective literal L is a *complete well-defined sequence* concluding L over a *set of rules* of the knowledge base Kb. By *complete* here we mean that all rules required for concluding L are in the sequence. By *well-defined sequence* we mean a (minimal) sequence of rules concluding L as follows: the head of the last rule in the sequence is an objective literal L; furthermore, if some atom L' (ignoring default literals) appears in the body of a rule then there must be a rule before this one with L' in the head; moreover, the sequence must not be circular and only use rules that are strictly necessary.

Definition 1 (Complete Well-defined Sequence). *Let P be an ELPd, and L an objective literal in the language of P. A* well-defined sequence *for L over a set of rules S is a finite sequence $[r_1; \ldots; r_m]$ of rules r_i from S of the form $L_i \leftarrow Body_i$ such that*

- *L is the head of the rule r_m, and*

– *an objective literal L' is the head of a rule r_i ($1 \leq i < m$) only if L' is not in the body of any r_k ($1 \leq k \leq i$) and L' is in the body of some rule r_j ($i < j \leq m$).*

We say that a well-defined sequence for L is complete *if for each objective literal L' in the body of the rules r_i ($1 \leq i \leq m$) there is a rule r_k ($k < i$) such that L' is the head of r_k.*

By the conclusions *of a sequence we mean the set of all objective literals in the head of some rule of the sequence, and by the* assumptions *we mean the set of all default literal in bodies.*

For dealing with consistent and paraconsistent reasoning, we define strong and weak arguments, based on strong and weak sets of rules, the former being simply the rules in the Kb . A *weak set of rules* results from adding to all rule bodies the default negation of the head's complement, and of $\bot$, thus making the rules weaker (more susceptible to being contradicted/attacked). Intuitively, if there is a potential inconsistency, be it by proving the explicit complement of a rules head or by proving $\bot$ then the weak argument is attacked, whereas the strong is not.

Definition 2 (Strong and Weak Arguments). *Let P be an ELPd, and L a literal in its language. Let the* weak set of rules of P *be defined as*

$$R_P^w = \{\ L \leftarrow Body, not\ \neg L, not\ \bot \mid L \leftarrow Body \in P\ \}$$

A strong *(resp.* weak*)* argument *of P for L, A_L^s (resp. A_L^w), is a complete well-defined sequence for L over P (resp. R_P^w).*

Let A_L^w and A_L^s be two arguments of P. A_L^w is the weak argument corresponding *to A_L^s, and vice-verse, if both use exactly the same rules of the original program P (the former by having rules R_P^w and the latter from P alone).*

We say that A_L is an argument *of P for L if it is either a strong argument or a weak one of P for L. We also say that A_L^k is a* k-argument of P *for L (where k is either s, for strong arguments, or w, for weak ones).*

After defining how arguments are built, we now move on to defining the attacking relation between these arguments. By using two kinds of arguments, strong and weak arguments as just exposed, we may rely on a single kind of attack. Indeed the different kinds of attacks usually considered in argumentation semantics for extended logic programs, *undercuts* and *rebuts* as in [15], can be captured by a single notion of attack. If an argument for an objective literal L (denoted by A_L) has a default negation $not\ L'$ in it, any argument for L' attacks (by undercut) A_L. The rebut attacking relation states that an argument also attacks another one when both arguments have complementary conclusions (i.e. one concludes L and the other $\neg L$). It is easy to see that with strong and weak arguments, *rebut can be reduced to undercut*: rebutting reduces to undercut attacks to weak arguments.

In our definition of attacks care must be taken in what regards arguments for $\bot$. By simply using undercut attacks any argument for $\bot$ attacks every weak argument. However, it does not make sense to attack arguments for objective literals if they do not

lead to *falsity*. Informally, an objective L literal leads to *falsity*if there is an argument A_L such that $A_\perp$ is built based on such an argument, e.g.

$$A^s_\perp : A^s_L + [\perp \leftarrow L, not\ L']$$

We only consider objective literals that are in the body of the rule for $\perp$ because these literals immediately lead to *falsity*. We assume that the involvement of other objective literals are not as strong as those in the body of the denial[3]. Then objective literals are *directly conflicting with* $A_\perp$ if the following holds:

Definition 3 (Directly Conflict with $A_\perp$). *Let $A_\perp$ be an argument for $\perp$, '$\perp \leftarrow Body$' be the rule in $A_\perp$ and $\{L_1, \ldots, L_n\}$ be the set of all objective literals in $Body$. The set of objective literals* directly conflicting with $A_\perp$ *is*

$$DC(A_\perp) = \{\perp\} \cup \{L_1, \ldots, L_n\}.$$

Definition 4 (Attack). *Let P be an ELPd. An argument A_L of P for L* attacks *an argument $A_{L'}$ of P for L' iff*

- *L is the symbol $\perp$, $not\ \perp$ belong to the body of some rule in $A_{L'}$, and $L' \in DC(A_L)$; or*
- *L is an objective literal different from $\perp$, and $not\ L$ belongs to the body of some rule in $A_{L'}$.*

Since attacking arguments can in turn be attacked by other arguments, comparing arguments is not enough to determine their acceptability w.r.t. the set of overall arguments. What is also required is a definition that determines the acceptable arguments on the basis of all the ways in which they interact, by proposing arguments and so opposing them. A subset S of proposed arguments of P is acceptable only if the set of all arguments of P does not have some valid opposing argument attacking the proposed arguments in S. As in [8, 15], we demand acceptable sets to contain all such arguments. Two questions remain open: how to obtain opposing arguments and, among these, which are valid?

An opposing argument for a proposed argument which makes an assumption, say $not\ L$, is simply an argument for a conclusion L. For an opposing argument A^o to be valid for attacking a proposed argument A^p in S, S should not have another argument that, in turn, attacks A^o (i.e. another argument that reinstates[4] A^p). In this case, we say that S cannot defend itself against A^o. This motivation points to a definition of acceptable sets of arguments S^i in P such as a set S is *acceptable* if it can attack all opposing arguments. So, we can say that a proposed argument A^p is acceptable w.r.t. a set S of acceptable arguments if and only if each opposing argument A^o attacking A^p is (counter-)attacked by an argument in S.

[3] We further assume they can be detected in a process of "belief revision", e.g. [3]. However, a discussion of this issue is beyond the scope of this proposal.

[4] The key observation is that an argument A that is attacked by another argument B can only be acceptable if it is *reinstated* by a third argument C, i.e by an acceptable argument C that attacks B.

However, it is still necessary to determine how strong arguments and weak arguments should interact w.r.t. such a set S of arguments. Based on the idea of reinstatement, both attacked and counter-attacking arguments should be of the same kind. For instance, if a proposing argument is strong (resp. weak) then every counter-attack against its opposing argument should be strong (resp. weak). A similar reason can be applied for opposing arguments. Therefore, proposed (resp. opposing) arguments should be of the same kind.

In the remainder of this paper we will use the notation p and o to distinguish the proposed argument from the opponent one, i.e. p (resp. o) is a (strong or weak) proposed (resp. opponent) argument. Since there are four possibilities of interaction between a proposed argument, A^p, and an opposing argument, A^o, the definition of arguments' acceptability (and the corresponding characteristic function) is generalised by parametrising the possible kinds of arguments, viz. strong arguments and weak arguments.

Definition 5 (Acceptable Argument). *Let P be an ELPd, p (resp. o) be the kind (strong or weak) of the proposed (resp. opposing) argument, $\mathcal{A}rgs^p(P)$ ($\mathcal{A}rgs^o(P)$) be the set of all arguments in P of kind p (resp. o) , and $S \subseteq \mathcal{A}rgs^p(P)$. An argument $A_L \in \mathcal{A}rgs^p(P)$ is an* acceptable$_{p,o}$ argument *w.r.t. S iff each argument $A_{L'} \in \mathcal{A}rgs^o(P)$ attacking A_L is attacked by an argument $A_{L''} \in S$.*

Note that this proposal is in accordance with the 'Compositional Principle' of [20]: "If an argument SA is a sub-argument of argument A, and SA is not acceptable w.r.t. a set of arguments S, then A is also not acceptable w.r.t. S". We now formalise the concept of acceptable arguments with a fixpoint characteristic function $p\,o$ of P:

Definition 6 (Characteristic Function). *Let P be an ELPd, and p (resp. o) be the kind (strong or weak) of the proposed (resp. opposing) argument of P, and $S \subseteq \mathcal{A}rgs^p(P)$. The* characteristic function $p\,o$ *of P and over S is:*

$$F_P^{p,o} : 2^{\mathcal{A}rgs(P)} \rightarrow 2^{\mathcal{A}rgs(P)}$$
$$F_P^{p,o}(S) = \{Arg \in \mathcal{A}rgs(P) \mid Arg \text{ is } acceptable_{p,o} \text{ w.r.t. } S\}.$$

It can be proven that this function is monotonic, and so it has a least fixpoint that can be obtained iteratively as usual:

Proposition 1. *Define for any P the following transfinite sequence of sets of arguments:*

- $S^0 = \emptyset$
- $S^{i+1} = F_P^{p,o}(S^i)$
- $S^\delta = \bigcup_{\alpha<\delta} S^\alpha$ *for limit ordinal δ*

Given that $F_P^{p,o}$ is monotonic, there must exist a smallest λ such that S^λ is a fixpoint of $F_P^{p,o}$, and $S^\lambda = lfp(F_P^{p,o})$.

Note that $lfp(F_P^{p,o})$ is well-behaved, i.e. arguments in it are $acceptable_{p,o}$ w.r.t. the set of all argument of P. By definition $lfp(F_P^{p,o})$ is minimal, which guarantees that it does not contain any argument of which acceptance is not required. Moreover, when $F_P^{p,o}$ is finitary the iterative process above is guaranteed to terminate after an enumerable number of steps.

Proposition 2. *$F_P^{p,o}$ is $finitary$ if each argument in S is attacked by at most a finite number of arguments in S.*

By knowing the set of all acceptable$_{p,o}$ arguments of P, we can split all arguments from $\mathcal{A}rgs(P)$ into three classes: justified arguments, overruled arguments and defensible arguments. Our definition of overruled is different from [15]'s proposal. In its proposal, the restriction applies that overruled arguments cannot be also justified and so [15]'s argumentation semantics is always consistent. Since we aim to obtain a paraconsistent way of reasoning, the status of an argument is defined as follows:

Definition 7 (Justified, Overruled or Defensible Argument). *Let P be an ELPd, p (resp. o) be the kind (strong or weak) of an argument of P, and $F_P^{p,o}$ be the characteristic function p o of P. An argument A_L^p is*

- justified$_P^{p,o}$ *iff it is in $lfp(F_P^{p,o})$*
- overruled$_P^{p,o}$ *iff the A_L^o corresponding to A_L^p is attacked by a justified$_P^{p,o}$ argument*
- defensible$_P^{p,o}$ *iff it is neither a justified$_P^{p,o}$ nor an overruled$_P^{p,o}$ argument*

We denote the $lfp(F_P^{p,o})$ by $JustArgs_P^{p,o}$.

We may also iteratively obtain overruled arguments based on the greatest fixpoint of the characteristic function which, by monotonicity of the characteristic function is guaranteed to exist and can also be obtained iteratively as usual. In fact:

Lemma 1. $gfp(F_P^{o,p}) = \{A_{L_1}^o : \neg(\exists A_{L_2}^p \in lfp(F_P^{p,o}) \mid A_{L_2}^p \text{ attacks } A_{L_1}^o)\}$

Lemma 2. $lfp(F_P^{p,o}) = \{A_{L_1}^p : \neg(\exists A_{L_2}^o \in gfp(F_P^{o,p}) \mid A_{L_2}^o \text{ attacks } A_{L_1}^p)\}$

Then, the following holds:

Theorem 1. *A_L^p is overruled$_P^{p,o}$ iff the A_L^o corresponding to A_L^p is not in $gfp(F_P^{o,p})$.*

Due to space limitations we do not detail here general properties when some other weaker restriction are imposed. Instead, we discuss some properties of $JustArgs_P^{p,o}$ and comparisons. Since p (resp. o) denote the kind of a proposed (resp. an opposing) argument, i.e. strong argument or weak argument, assume that p (resp. o) in $\{s, w\}$. Both $JustArgs_P^{w,w}$ and $JustArgs_P^{w,s}$ are both conflict-free[5] and non-contradictory[6]. Thus, every argument in both $JustArgs_P^{w,w}$ and $JustArgs_P^{w,s}$ is non-contradictory, i.e. it is not related to a contradiction at all. Furthermore, $F_P^{w,w}$ has more defensible arguments than $F_P^{w,s}$. Therefore, we obtain a consistent way of reasoning in Ag if we apply $F_P^{w,w}$ over $\mathcal{A}rgs(P)$.

In contrast, $JustArgs_P^{s,s}$ and $JustArgs_P^{s,w}$ may be contradictory. However, to evaluate the acceptability of available arguments without considering the presence of *falsity* or both arguments for L and $\neg L$, the proposed arguments should be strong ones, and every opposing argument is a weak argument. Since $F_P^{s,w}$ respects the 'Coherence Principle' of [13, 1], i.e. given that every opposing argument is a weak one, it can be

[5] A set S of arguments is conflict-free if there is no argument in S attacking an argument in S.

[6] A set S of arguments is non-contradictory if neither an argument for *falsity* nor both arguments for L and $\neg L$ are in S.

attacked by any proposed argument for its explicit negation. Therefore, we obtain a paraconsistent way of reasoning if we apply $F_P^{s,w}$ over $\mathcal{A}rgs(P)$.

Moreover, a justified$_P^{s,w}$ argument of an agent can be related to a contradiction with respect to $JustArgs_P^{s,w}$ as follows. We first define that an argument that reinstate another argument is its *counter-attack*:

Definition 8 (Counter-Attack). *Let P be an ELPd, S a set of arguments from P, A_L be an argument in S, and $A_{L'}$ be an argument of P attacking A_L. A* counter-attack *for A_L against $A_{L'}$ is an argument in S that attacks $A_{L'}$. $CA_{A_L}(A_{L'}, S)$ is the set of all counter-attacks for A_L against $A_{L'}$ in S*

Definition 9 (Relation to a Contradiction). *Let P be an ELPd. A justified$_P^{s,w}$ argument A_L^s is*

- contradictory$_P^{s,w}$ *if $JustArgs_P^{s,w}$ is contradictory w.r.t. L, or there exists a contradictory$_P^{s,w}$ argument $A_\perp^s$ and $L \in DC(A_\perp^s)$; or*
- based-on-contradiction$_P^{s,w}$ *if for all $A_{L'}^w$ attacking A_L^s there exists a contradictory$_P^{s,w}$ or based-on-contradiction$_P^{s,w}$ argument in $CA_{A_L}(A_{L'}^w, JustArgs_P^{s,w})$, or there exists an L' in the head of some rule in A_L^s, different from L and $\perp$, such that $JustArgs_P^{s,w}$ is contradictory w.r.t. L'; or*
- non-contradictory$_P^{s,w}$ *iff it is neither contradictory$_P^{s,w}$ nor it is based-on-contradiction$_P^{s,w}$.*

Proposition 3. *A justified$_P^{s,w}$ argument A_L^p is non-contradictory$_P^{s,w}$ if for no head L' of a rule in A_L^s, $JustArgs_P^{s,w}$ is contradictory w.r.t. L', and every counter-attack for A_L^s is a non-contradictory$_P^{s,w}$ argument.*

A *truth value* of an agent's conclusion in a (consistent or paraconsistent) way of reasoning is as follows:

Definition 10 (Truth Value of a Conclusion). *Let P be an ELPd, and $L \in \mathcal{H}(P)$, and $k \in \{s, w\}$. A literal L over P is*

- false$_P^{k,w}$ *iff every k-argument for L is overruled$_P^{k,w}$*
- true$_P^{k,w}$ *iff there exists a justified$_P^{k,w}$ argument for L. Moreover, L is*
  - contradictory$_P^{k,w}$ *if L is the symbol $\perp$ or there exists a justified$_P^{k,w}$ argument for $\neg L$*
  - based-on-contradiction$_P^{k,w}$ *if it is both true$_P^{k,w}$ and false$_P^{k,w}$*
  - non-contradictory$_P^{k,w}$*, otherwise*
- undefined$_P^{k,w}$ *iff L is neither true$_P^{k,w}$ nor false$_P^{k,w}$ (i.e. there is no justified$_P^{k,w}$ argument for L and at least one k-argument for L is not overruled$_P^{k,w}$).*

Example 1 (Privacy of Personal Life – PPL). Usually, any person deserves privacy with respect to her personal life. However, when such a person behaves in a way that is not acceptable (e.g. selling drugs) she will suffer the consequences. The first consequence is the focus of media attention on her personal life and consequent loss of privacy. The personal life of such a person might be exposed by the "results" of media attention (e.g.

photos, reports, and so on), unless there is a law that protects her against it. The above description can be expressed by the following extended logic programming rules.

$$focusOfMediaAttention(X) \leftarrow person(X),\ \neg acceptableBehavior(X).$$
$$\neg acceptableBehavior(X) \leftarrow involved(X,Y),\ againstSociety(Y).$$
$$\neg hasPrivacy(X) \leftarrow focusOfMediaAttention(X).$$
$$personalLifeExposed(X) \leftarrow \neg hasPrivacy(X), not\ protectedByLaw(X).$$
$$hasPrivacy(X) \leftarrow person(X),\ not\ \neg hasPrivacy(X).$$

In contrast, it is considered an absurdity that someone may lose her privacy when she is involved in some event for which there is no evidence of being public (e.g. someone starting a long-term treatment for drugs dependency). The absurdity in the rule below is represented as a denial:

$$\bot \leftarrow \neg hasPrivacy(X),\ event(X,Y),\ not\ publicEvent(Y).$$

Moreover, modern society normally tries to protect children, and so their privacy is guaranteed until evidence appears of some unusual behaviour (e.g. by having unacceptable behaviour).

$$hasPrivacy(X) \leftarrow child(X),\ not\ unusualChild(X).$$
$$unusualChild(X) \leftarrow child(X), \neg acceptableBehavior(X).$$
$$person(X) \leftarrow child(X).$$

However, famous persons are inherently the focus of media attention:

$$focusOfMediaAttention(X) \leftarrow famousPerson(X).$$
$$person(X) \leftarrow famousPerson(X).$$

Assume an agent Ag with the knowledge above, plus some facts about Potira and Ivoti [7]. Potira is a famous child, and Ivoti is a famous soccer player in treatment for drugs dependency:

$$child(potira). \qquad famousPerson(potira).$$
$$famousPerson(ivoti). \quad event(ivoti, treatmentForDrugsDependency).$$

Figure 1 illustrates, with obvious abbreviations, the possible attacks of arguments for "privacy of Potira's life" over $\mathcal{A}rgs(PPL)$. The notation for that figure is as follows: Arguments are represented as nodes. A solid line from argument A to argument B means "A attacks B", a dotted line from A to B means "A is built based on B", and a line with dashes means "A reinstates B". A round node means "it is an acceptable argument" and a square node means "it is not an acceptable argument", which are w.r.t. the set of arguments of $\mathcal{P}$. Then we can presume both the status of the arguments and the truth value of the conclusions of PPL.

[7] The following names are from Native South American, more specifically from the Tupi-Guarani family, Potira and Ivoti both mean "flower".

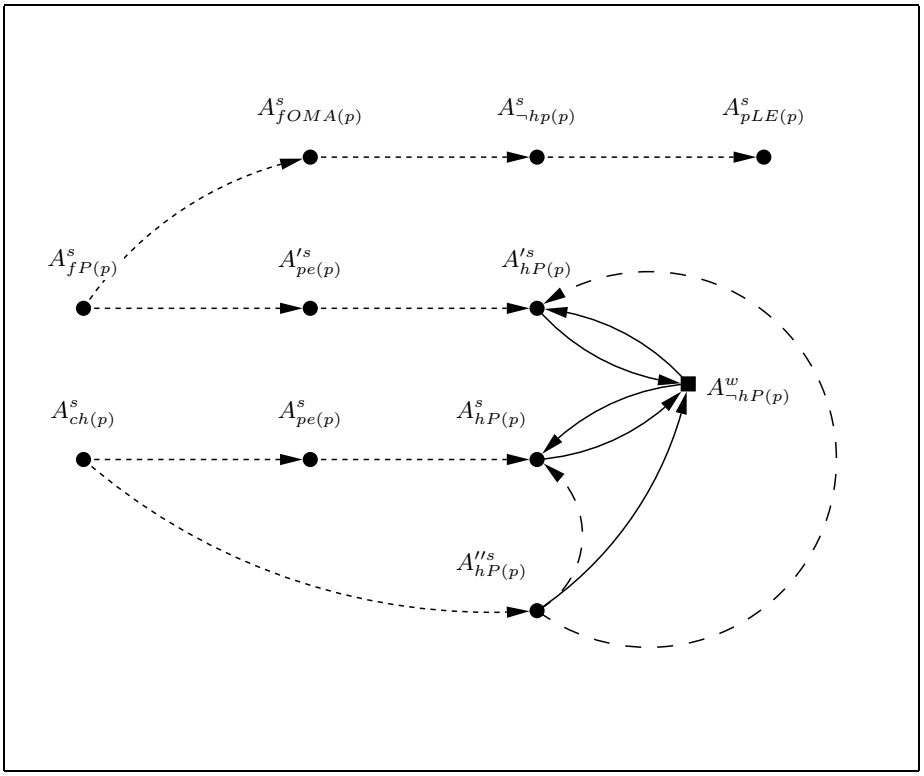

Fig. 1. Acceptable arguments in $Args(PPL)$ for Potira

The argument for "Potira has no Privacy" ($A^s_{\neg hp(p)}$) and also the arguments for "Potira has privacy" ($A^s_{hp(p)}$, $A'^s_{hp(p)}$, $A''^s_{hp(p)}$) are contradictory$^{s,w}_{PPL}$; the argument "Portira has her personal life exposed" ($A^s_{pLE(p)}$) is either based-on-contradiction$^{s,w}_{PPL}$ and overruled$^{s,w}_{PPL}$. The other arguments are non-contradictory$^{s,w}_{PPL}$. Therefore, the truth values for conclusions about Potira are as follows:

- $fP(p)$, $ch(p)$, $fOMA(p)$ and $pe(p)$ are non-contradictory$^{s,w}_{PPL}$;
- $hp(p)$ and $\neg hp(p)$ are both (true$^{s,w}_{PPL}$ and) contradictory$^{s,w}_{PPL}$ and false$^{s,w}_{PPL}$; and
- $pLE(p)$ is both based-on-contradiction$^{s,w}_{PPL}$ and false$^{s,w}_{PPL}$.

Moreover, the truth values for conclusions regarding Ivoti are as follows:

- $fP(i)$, $pe(p)$ and $fOMA(i)$ are non-contradictory$^{s,w}_{PPL}$;
- $hp(i)$ and $\neg hp(i)$ are both contradictory$^{s,w}_{PPL}$ and false$^{s,w}_{PPL}$; and
- "There is *falsity* in PPL" (i.e. $\bot$) is both (true$^{s,w}_{PPL}$ and) contradictory$^{s,w}_{PPL}$ and false$^{s,w}_{PPL}$. Then
- $ev(i, TFDD)$ and $pLE(i)$ are both based-on-contradiction$^{s,w}_{PPL}$ and false$^{s,w}_{PPL}$.

3 A proof for an argument

Though the declarative semantics just exposed may rely on an iterative procedure, its usage for computing arguments may not always be appropriate. This is specially the case when we are only interested in the proof for a (query) argument, rather than all acceptable arguments, as is obtained by the iterative process. Such a query oriented proof procedure can be viewed as conducting a "dispute between a proponent player and an opponent player" in which both proponent and opponent exchange arguments. In its simplest form, the dispute can be viewed as a sequence of alternating arguments:

$$PR_0, OP_0, PR_1, \ldots, PR_i, OP_{i+1}, PR_{i+2}, \ldots$$

The proponent puts forward an initial argument PR_0. For every argument PR_i put forward by the proponent, the opponent attempts to respond with an attacking argument OP_{i+1} against PR_i. For every attacking argument OP_{i+1} put forward by the opponent, the proponent attempts to counter-attack with a proposed argument PR_{i+2} against OP_i. To win the dispute, the proponent needs to have a proposed argument against every opposing argument of the opponent. Therefore, a winning dispute can be represented as a dialogue tree, which represents the top-down, step-by-step construction of a proof tree. We follow [15]'s proposal, which defines a proof for an argument A_L as a dialogue tree for A_L. However, our definition of dialogue tree is in accordance with the acceptability of the arguments of an ELPd P (see Def. 5):

> A proposed argument $A_L \in \mathcal{A}rgs^p(P)$ is acceptable if all of its opposing arguments in $\mathcal{A}rgs^o(P)$ are attacked by acceptable arguments from $\mathcal{A}rgs^p(P)$.

To define a dialogue tree for an argument A_L we need first a definition of *dialogue for an argument*. A dialogue for A_L is a sequence of PR and OP moves of proposed arguments and opposing arguments, such that the first PR move is the argument A_L. Each OP (resp. PR) move of a dialogue consists of an argument from $\mathcal{A}rgs^o(P)$ (resp. $\mathcal{A}rgs^p(P)$) attacking the previous proposed (resp. opposing) argument in such a dialogue. Intuitively, we can see that every PR move wants the conclusion of A_L to be acceptable, and each OP move only wants to prevent the conclusion of A_L from being acceptable. In the case of PR moves, we can further say that if we impose a restriction that proposing arguments cannot be used more than once in a sequence of moves of a dialogue, then the dialogue will have a finite sequence of PR and OP moves. Therefore, none of the proposed arguments can be used more than once in the same dialogue, but any of the opposing arguments can be repeated as often as required to attack a proposed argument.

Definition 11 ($dialogue^{p,o}_{A_L}$**).** *Let P be an ELPd, p (resp. o) be the kind (strong or weak) of a proposed (resp. an opposing) argument of P, and $\mathcal{A}rgs^p(P)$ and $\mathcal{A}rgs^o(P)$ be the set of p-arguments and o-arguments of P, respectively. A* dialogue *p o (in P) for an argument $A_L \in \mathcal{A}rgs^p(P)$, called $dialogue^{p,o}_{A_L}$, is a finite non-empty sequence of m moves $move_i = A_{L_i}$ ($1 \leq i \leq m$) such that*

1. $move_1 = A_L$

2. *for every* $1 < i \leq m$, A_{L_i} *attacks* $A_{L_{i-1}}$ *and*
   - *if* i *is odd then* $A_{L_i} \in \mathcal{A}rgs^p(P)$ *and there is no odd* $j < i$ *such that* $A_{L_j} = A_{L_i}$, *or*
   - *if* i *is even then* $A_{L_i} \in \mathcal{A}rgs^o(P)$.

We say that $move_i$ *is odd if* i *is odd; otherwise,* $move_i$ *is even.*

A dialogue for A_L *succeeds* if its last move is a PR move. In this proposal, we want to guarantee that a dialogue tree for an argument A_L is finitary (cf. Prop. 2). Nevertheless, we only consider grounded finite ELPd in order to relate the declarative semantics (presented in the previous section) to this proposal of operational semantics. By considering this, every dialogue in such a dialogue tree finishes because there will always be a last move PR (resp. OP) in such a dialogue, so no opposing (resp. proposed) argument against it exists. For non-grounded (infinite) programs, there may be (failed) dialogues with an infinite sequence of moves. In such a case, these dialogues should be considered failures, and the argument for such a dialogue tree should be deduced as defensible. The main problem of such an approach is detecting an infinite sequence of moves in a dialogue. However, the following definition will consider cases of both 'grounded finite ELPd' and 'non-grounded (infinite) ELPd'.

Definition 12 (The Status of a dialogue). *Let* P *an ELPd. A dialogue* $p\ o$ *(in* P*) for an argument* $A_L \in \mathcal{A}rgs^p(P)$ *is* completed *iff its last move is* m*, and*

- *if* m *is odd then there is no argument in* $\mathcal{A}rgs^o(P)$ *attacking* A_{L_m}, *or*
- *if* m *is even then there is no argument in* $\mathcal{A}rgs^p(P) - S_p$ *attacking* A_{L_m} *where* S_p *is the set of all* A_{L_j} *in the sequence such that* j *is odd*

(or it is infinite*). A completed dialogue is* failed *iff its last move is* odd *(or it is infinite); otherwise, it* succeeds.

Note that a dialogue$^{p,o}_{A_L}$ in P and the $lfp(F^{p,o}_P)$ "grow up" in different ways. In the former, an argument A in the last move, $move_f$, is not attacked by any argument in $\mathcal{A}rgs(P)$. Since A attacks the previous move, $move_{f-1}$, we can say that the argument B in $move_{f-2}$ was reinstated by A. Thus, each $move_i$ ($1 \leq i < f-1$) is reinstated by $move_{i+2}$. The latter evaluates argument A as acceptable in the first iteration of the characteristic function $F^{p,o}_P$. In the second iteration, A reinstates B, so that B is acceptable and might reinstate other arguments in all following iterations. We can further say that dialogue$^{p,o}_{A_L}$ decreases (in a top-down way) and $lfp(F^{p,o}_P)$ increases (in a bottom-up way) the set of evaluated arguments.

Proposition 4. *Let* $move_m = A_L$ *be the last move of a succeeded dialogue*$^{p,o}_{A'_L}$ *in* P. *Then,* $A_L \in F^{p,o}_P(\emptyset)$.

A dialogue tree DT for A_L is held between a proposed argument PR and its opposing argument OP against PR, where the root of DT is A_L. The dialogue tree DT considers all possible ways in which A_L can be attacked because each branch of DT is a dialogue for A_L, i.e. every single dialogue for A_L is built because we should consider the overall arguments in $\mathcal{A}rgs(P)$ to deduce the status of A_L. The dialogue tree DT for an argument A_L succeeds if every dialogue of DT succeeds.

Definition 13 ($DT^{p,o}_{A_L}$). *Let P be an ELPd, p (resp. o) be the kind (strong or weak) of the proposed (resp. opposing) argument of $\mathcal{A}rgs(P)$, and $\mathcal{A}rgs^p(P)$ (resp. $\mathcal{A}rgs^o(P)$) be the set of p-arguments (o-arguments) of P. A* dialogue tree *$p\ o$ (in P) for $A_L \in \mathcal{A}rgs^p(P)$, called $DT^{p,o}_{A_L}$, is a finite tree of moves $move_i = A_{L_i}$ ($i > 0$) such that*

1. *each branch of $DT^{p,o}_{A_L}$ is a $dialogue^{p,o}_{A_L}$, and*
2. *for all i, if $move_i$ is*
   - *even then its only child is a p-argument attacking $A_{L_i} \in \mathcal{A}rgs^o(P)$, or*
   - *odd then its children are all o-arguments attacking $A_{L_i} \in \mathcal{A}rgs^p(P)$*

A $DT^{p,o}_{A_L}$ succeeds *iff all branches (i.e. all $dialogue^{p,o}_{A_L}$) of the tree succeeds.*

Based on the second condition of Definition 13, we might obtain more than one dialogue tree for an argument. This occurs because only one proponent's move is built for each opponent's move of a dialogue tree. For instance,

Example 2. Let $P = \{p \leftarrow not\ a;\ a \leftarrow not\ b, not\ c;\ a \leftarrow not\ d;\ b;\ c \leftarrow not\ g;\ g\}$. There are two possible $DT^{s,s}_{A_p}$ in P: the first dialogue tree does not succeed because there is a last move which is an o-argument, viz $[a \leftarrow not\ b]$; the second one also does not succeed because every last move is an o-argument, viz $[g]$ and $[a \leftarrow not\ d]$.

At this point we can relate, for grounded finite programs, the results from a $DT^{p,o}_{A_L}$ to the status of the argument A_L (see Def. 7), as follows:

Proposition 5. *An argument A^p_L in a grounded finite P is*

- *justified$^{p,o}_P$ iff there exists a successful $DT^{p,o}_{A_L}$*
- *overruled$^{p,o}_P$ iff for all $DT^{p,o}_{A_L}$: there exists a $move_2 = A^o_{L'}$ such that $DT^{o,p}_{A_{L'}}$ succeeded*
- *defensible$^{p,o}_P$ iff it is neither justified$^{p,o}_P$ nor overruled$^{p,o}_P$.*

The following example illustrate the concepts presented in Proposition 5.

Example 3. Let $P2 = \{a \leftarrow not\ b;\ \neg a;\ b;\ \neg b;\ c;\ \bot \leftarrow not\ c\}$. On the top of Figure 2, it is illustrated the possible $DT^{w,w}_{A_L}$ in $P2$. Note that each dialogue tree does not succeed because its last move is an o-argument. Nevertheless, all arguments are defensible$^{w,w}_{P2}$ because none of these last moves are justified$^{w,w}_{P2}$. On the bottom of the Figure 2 it is also illustrated the possible $DT^{s,w}_{A_L}$ in $P2$. In such a case, all arguments are justified$^{s,w}_{P2}$.

Proposition 6. *A justified$^{s,w}_P$ argument A^s_L in a finite ground P is*

- contradictory$_{s,w}$ *iff L is the symbol $\bot$, or different from $\bot$ and there exists at least a successful $DT^{s,w}_{A_{\neg L}}$; or*
- based-on-contradiction$_{s,w}$ *iff A^s_H is not contradictory$_{s,w}$ and*
  - *there exists a contradictory^{s,w} $A^s_{L'}$ (with a rule $L' \leftarrow Body$) such that $L \in Body$, or*
  - *there exists an L'' in the head of a rule in A^s_L such that $A^s_{L''}$ is contradictory^{s,w}, or*

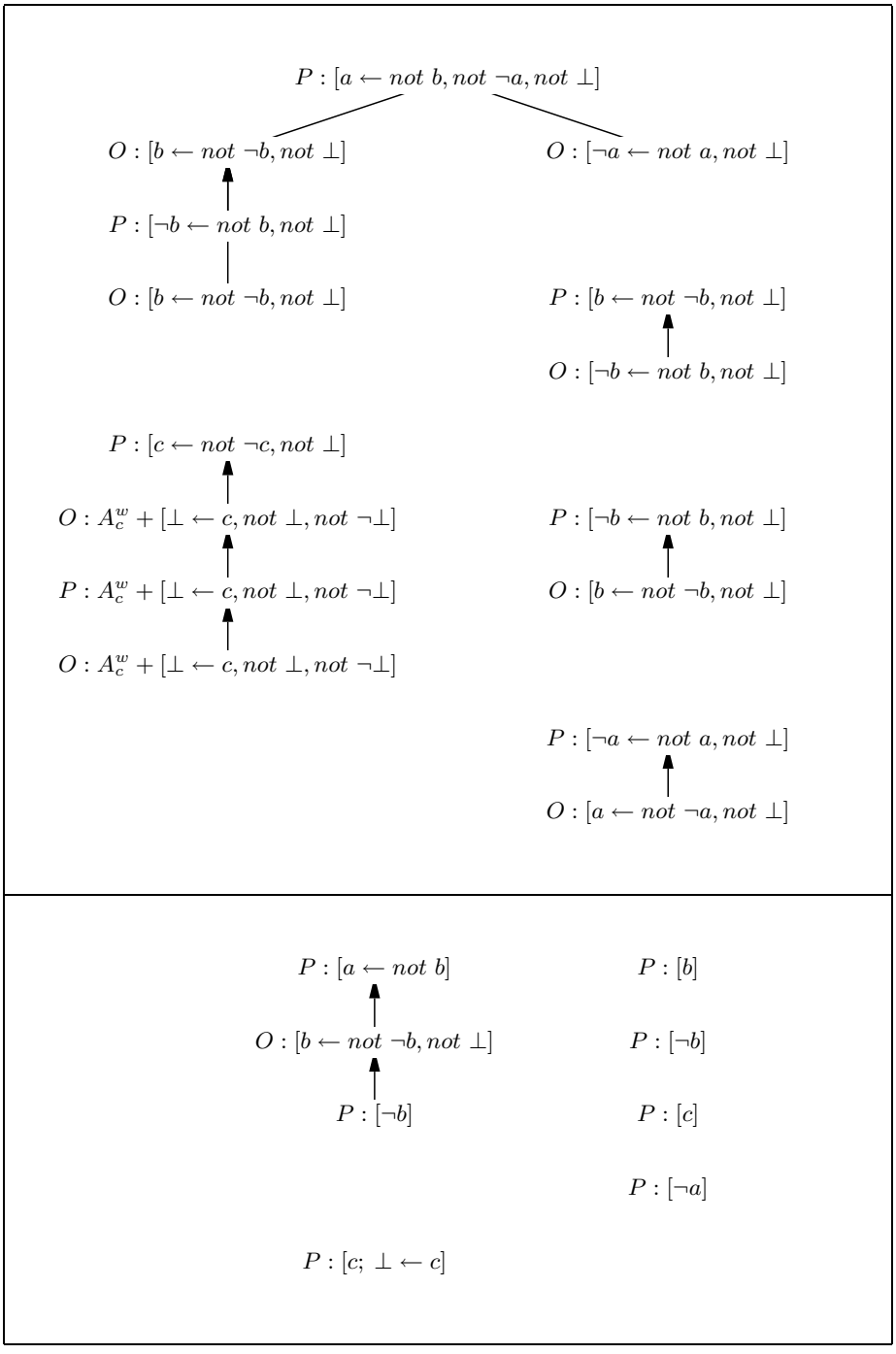

Fig. 2. Some $DT^{w,w}_{A_L}$ and $DT^{s,w}_{A_L}$ in $\{a \leftarrow not\ b; \neg a; b; \neg b; c; \bot \leftarrow not\ c\}$

  - *for all* $dialogue^{s,w}_{A_H}$ *in* $DT^{s,w}_{A_H}$*: the last move has not a non-contradictory*$_{s,w}$ *argument; or*
- non-contradictory$_{s,w}$*, otherwise.*

To conclude about the truth value of an objective literal L we evaluate more than one dialogue tree of each argument for such L:

Proposition 7. *An objective literal* H *is*

- $true^{p,o}_P$ *iff there exists a successful* $DT^{p,o}_{A_H}$*. Thus,* H *is*
  - *contradictory*$^{p,o}_P$ *iff for all successful* $DT^{p,o}_{A_H}$*:* A^p_H *is contradictory*$^{p,o}_P$*, or*
  - *based-on-contradiction*$^{p,o}_P$ *iff for all successful* $DT^{p,o}_{A_H}$*:* A^p_H *is based-on-contradiction*$^{p,o}_P$*, or*
  - *non-contradictory*$^{p,o}_P$ *iff there exists a successful* $DT^{p,o}_{A_H}$ *such that* A^p_H *is non-contradictory*$^{p,o}_P$*;*
- $false^{p,o}_P$ *(in* P*) iff* $\forall DT^{p,o}_{A_H}$*:* A^p_H *is overruled*$^{p,o}_P$*;*
- $undefined^{p,o}_P$ *(in* P*) iff* $\forall DT^{p,o}_{A_H}$*:* A^p_H *is neither justified*$^{p,o}_P$ *nor overruled*$^{p,o}_P$*.*

Example 4. Following Example 3, all literals of $P2$ are justified$^{s,w}_{P2}$. However, all literals of $P2$ are undefined$^{w,w}_{P2}$.

4 Conclusions and Further Work

Our argumentation semantics is based on the argumentation metaphor, in the line of the work developed in [9, 15, 2, 18] for defining semantics of single extended logic programs. In these argumentation-based semantics, rules of a logic program are viewed as encoding arguments of an agent. More precisely, an argument for an objective literal L is a sequence of rules that "proves" L, if all default literals (of the form $not\ L'$) in the body of those rules are assumed true. In other words, arguments encoded by a program can attack – by undercut – each other. Moreover, an argument for L attacks – by rebut – another argument if this other argument assumes its explicit negation (of the form $\neg L$). The meaning of the program is so determined by those arguments that somehow (depending on the specific semantics) can defend themselves from the attacks of other arguments.

We generalise [15]'s definition of argument by proposing two kind of arguments, viz. strong arguments and weak arguments. By having two kinds of arguments, viz. strong arguments and weak arguments, the attack by undercut is not needed. Simply note that rebut are undercut attacking weak arguments. Therefore, rebut is not considered in our proposal since, as already shown in [17, 6, 18], it can be reduced to undercut by considering weaker versions of arguments. [2] also defines a methodology for transforming non-exact, defensible rules into exact rules with explicit non-provability conditions and shows that this transformation eliminates the need for rebuttal attacks and for dealing with priorities in the semantics.

Similar to [9, 15] we formalise the concept of acceptable arguments with a fixpoint operator. However, the acceptability of an argument might have different results and it depends on which kind of interaction between (strong and weak) arguments is chosen. Therefore, our argumentation semantics assigns different levels of acceptability to an

argument for an objective literal L and so it can be justified, overruled, or defensible. Moreover, a justified argument for L can be contradictory, based on contradiction, or non contradictory. Consequently, a truth value of L can be true (and contradictory, based on contradiction, or non contradictory), false or undefined.

Since our argumentation semantics is parametrised by the kind of interaction between arguments, we obtain results from a consistent way of reasoning to a paraconsistent way of reasoning. A consistent way of reasoning neither concludes that L nor $\neg L$ are true, even if one of these is a fact. A paraconsistent way of reasoning can conclude L is true even if it also concludes that $\neg L$ is true. Given that we consider denials in the agent's knowledge base – in a conflicting situation – a consistent way of reasoning cannot conclude that a given L is true if L is related with the presence of the *falsity*; a paraconsistent way of reasoning might conclude L even it is related with *falsity*. Furthermore, our argumentation semantics (and the corresponding proof procedure) succeeds in detecting conflicts in a paraconsistent extended logic program with denials, i.e. it handles with contradictory arguments and with the presence of *falsity*.

For this proposal we have made two implementations, both in XSB System (by resorting to tabling) [19] which computes the argumentation Prolog implementation over an agent's knowledge base. One bottom-up implementation of the semantics, following closely its declarative definition; another of query-driven proof procedures for the semantics. The proof procedure has also been implemented by using the toolkit Interprolog [4], a middle-ware for Java and Prolog which provides method/predicate calling between both.

As we mentioned, the original semantics, defined in [5], is a generalisation of the one presented here to a distributed argumentation-based negotiation semantics. As future work we intend to generalise this (centralised) proof procedure to a distributed proof procedure seeing the negotiation process as a forest of dialogue trees, rather than a single tree as here.

References

1. J. J. Alferes, C. V. Damásio, and L. M. Pereira. A logic programming system for non-monotonic reasoning. *Journal of Automated Reasoning*, 14(1):93–147, 1995.
2. A. Bondarenko, P. M. Dung, R. Kowalski, and F. Toni. An abstract, argumentation-theoretic approach to default reasoning. *Journal of Artificial Intelligence*, 93(1–2):63–101, 1997.
3. L. M. Pereira e M. Schroeder C. V. Damásio. Revise: Logic programming and diagnosis. In U. Furbach J. Dix and A. Nerode, editors, *4th International Conference (LPNMR'97)*, volume LNAI 1265 of *Logic Programming and NonMonotonic Reasoning*, pages 353–362. Springer, July 1997.
4. M. Calejo. Interprolog: Towards a declarative embedding of logic programming in java. In J. J. Alferes and J. Leite, editors, *9th European Conference (JELIA 2004)*, LNAI, pages 714–71. Springer, 2004. Toolkit available at http://www.declarativa.com/InterProlog/.
5. Iara Carnevale de Almeida and José Júlio Alferes. An argumentation-based negotiation framework. In K. Inoue, K. Satoh, and F Toni, editors, *VII International Workshop on Computational Logic in Multi-agent Systems (CLIMA)*, volume 4371 of *LNAI*, pages 191–210. Springer, 2006. Revised Selected and Invited Papers.

6. Iara de Almeida Móra and José Júlio Alferes. Argumentative and cooperative multi-agent system for extended logic programs. In F. M. Oliveira, editor, *XIVth Brazilian Symposium on Artificial Intelligence*, volume 1515 of *LNAI*, pages 161–170. Springer, 1998.
7. P. Dung, P. Mancarella, and F. Toni. *Computational Logic: Logic Programming and Beyond – Essays in Honour of Robert A. Kowalski*, volume 2408, chapter Argumentation-based proof procedures for credulous and sceptical non-monotonic reasoning, pages 289–310. Springer, 2002.
8. P. M. Dung. An argumentation semantics for logic programming with explicit negation. In *10th International Conference on LP (ICLP)*, pages 616–630. MIT Press, 1993.
9. P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Journal of Artificial Intelligence*, 77(2):321–357, 1995.
10. P. M. Dung, R. Kowalski, and F. Toni. Argumentation-theoretic proof procedures for default reasoning. Technical Report. Available at http://www.doc.ic.ac.uk/˜ft/PAPERS/arg03.pdf, May 2003.
11. M. Gelfond and V. Lifschitz. Logic programs with classical negation. In Warren and Szeredi, editors, *7th International Conference on LP (ICLP)*, pages 579–597. MIT Press, 1990.
12. R. P. Loui. Process and policy: Resource-bounded non-demonstrative reasoning. *Journal of Computational Intelligence*, 14:1–38, May 1998.
13. L. M. Pereira and J. J. Alferes. Well founded semantics for logic programs with explicit negation. In *European Conference on Artificial Intelligence (ECAI)*, pages 102–106. John Wiley & Sons, 1992.
14. J. L. Pollock. Defeasible reasoning with variable degrees of justification. *Journal of Artificial Intelligence*, 133:233–282, 2002.
15. H. Prakken and G. Sartor. Argument-based extended logic programming with defeasible priorities. *Journal of Applied Non-Classical Logics*, 7:25–75, 1997.
16. H. Prakken and G. A. W. Vreeswijk. *Handbook of Philosophical Logic*, volume 4, chapter Logics for Defeasible Argumentation, pages 218–319. Kluwer Academic, 2 edition, 2002.
17. Michael Schroeder, Iara de Almeida Móra, and José Júlio Alferes. Vivid agents arguing about distributed extended logic programs. In Ernesto Costa and Amilcar Cardoso, editors, *Progress in Artificial Intelligence, 8th Portuguese Conference on Artificial Intelligence (EPIA)*, volume 1323 of *LNAI*, pages 217–228. Springer, 1997.
18. R. Schweimeier and M. Schroeder. Notions of attack and justified arguments for extended logic programs. In F. van Harmelen, editor, *15th European Conference on Artificial Intelligence*. IOS Press, 2002.
19. T. Swift and et all. Xsb - a logic programming and deductive database system for unix and windows. Technical report, XSB project, 2003. Toolkit available at *http://xsb.sourceforge.net/*.
20. G. A. W. Vreeswijk. Abstract argumentation systems. *Journal of Artificial Intelligence*, 90(1–2):225–279, 1997.