Computing argumentation in logic programming *

Antonios C. Kakas
Department of Computer Science
University of Cyprus
75 Kallipoleos Street, Nicosia P.O. Box 537, Cyprus
antonis@turing.cs.ucy.ac.cy

Francesca Toni
Department of Computing
Imperial College of Science, Technology and Medicine
180 Queen's Gate, London SW7 2BZ, UK
ft@doc.ic.ac.uk

November 27, 1998

Abstract

In recent years, argumentation has been shown to be an appropriate framework in which logic programming with negation as failure as well as other logics for non-monotonic reasoning can be encompassed. Many of the existing semantics for negation as failure in logic programming can be understood in a uniform way using argumentation. Moreover, other logics for non-monotonic reasoning that can also be formulated via argumentation can be given new semantics, by a direct extension of the logic programming semantics.

In this paper we develop an abstract computational framework where various argumentation semantics can be computed via different parametric variations of a simple basic proof theory. This proof theory is given in terms of derivations of trees where each node in a tree contains an argument (or attack) against its corresponding parent node. The proposed proof theory, defined here for the case of logic programming, generalises directly to other logics for non-monotonic reasoning that can also be formalised via argumentation.

The abstract proof theory forms the basis for developing concrete top-down proof procedures for query evaluation. These proof procedures are obtained by adopting specific search strategies and ways of computing attacks in the particular argumentation framework. For logic programming these procedures can be seen as a generalisation of the Eshghi-Kowalski abductive proof procedure that in turn generalises SLDNF.

*This is an extended and revised version of the earlier paper [46].

1 Introduction

Recently, it has been shown [4, 12, 20, 21, 22, 26, 27] that Logic Programming (LP) with Negation as Failure (NAF) as well as many other logics for non-monotonic reasoning, e.g. Reiter's default logic [39], Moore's autoepistemic logic [31] and McDermott's non-monotonic modal logics [30], can be understood, in a uniform way, within an argumentation framework. In particular, many existing semantics for LP with NAF can be given an equivalent formulation in argumentation terms, e.g. stable models [17], partial stable models [40] and preferred extensions [11], stationary expansions [37] and complete scenaria [11], well-founded model [48], stable theories [24] and acceptability semantics [26] (and semantics equivalent to these). All of these argumentation semantics for LP carry through to any other logic for non-monotonic reasoning that can also be understood in argumentation terms. For example, the stable model semantics for LP corresponds to the conventional semantics for these formalisms. Other argumentation semantics for LP can be generalised to these formalisms, thus providing new semantics, alternative to the conventional ones.

In this paper, we develop a general computational model for computing the argumentation-theoretic formulation of (most) semantics for LP. We define abstract proof theories in such a way that they can be directly adapted to other non-monotonic formalisms, when they are expressed in argumentation terms.

In the argumentation-based approach to the semantics of LP, a logic program P is seen as a theory in a background monotonic logic (the logic of Horn clauses) that can be extended by suitable subsets, Δ, of a given set of "hypotheses" $\mathcal{H}$ (c.f. [34]), consisting of all negative (variable-free) literals of the form $not\, p$ that can be constructed in the language of P. A set of hypotheses Δ can be seen as an "argument" supporting the consequences of the extended theory $P \cup \Delta$ derived in the underlying monotonic logic, where the (negative) hypotheses in Δ are understood as (ground) atomic facts. In general, not all subsets of $\mathcal{H}$ are suitable to extend P. For example, a subset Δ may be "self-conflicting", in the sense that $P \cup \Delta$ entails an atom p, with $not\, p$ in Δ. More interestingly, two different sets of hypotheses (or arguments) Δ and Δ' can be "conflicting" with each other with only one of them forming an "allowed extension" of P.

The various LP semantics can be understood via different criteria for selecting appropriate subsets of $\mathcal{H}$ to provide allowed extensions of P. These different criteria can all be defined in terms of a common notion of "attack" between subsets of $\mathcal{H}$. Informally, a set of hypotheses Δ attacks another set Δ' if some consequence of the theory $P \cup \Delta$ is in conflict with Δ' by entailing an atom p, with $not\, p$ in Δ'. The different selection criteria are then based upon comparing candidate sets of hypotheses with attacking sets of hypotheses. For example, the stable model semantics can be understood in terms of "conflict-free" sets of hypotheses that attack all hypotheses external to them. The partial stable model (or preferred extension) semantics can be understood in terms of conflict-free sets of hypotheses Δ that are able to attack back any set of hypotheses attacking Δ. The well-founded model and acceptability semantics can be understood in terms of recursively defined selection criteria, whose satisfaction by a particular set of hypotheses depends on whether other (attacking) sets of hypotheses might or might not satisfy the same criteria. Other formalisms for non-monotonic reasoning can be understood in argumentation terms similarly, with appropriate notions of hypotheses, arguments, conflict and attack.

The main aim of this paper is to develop an abstract proof theory providing, by appropriate instantiations of its parameters, proof theories for different argumentation semantics, corresponding to the existing semantics for NAF. In particular, we will investigate the computation of: partial stable model, stable theory, acceptability and well-founded model semantics (and semantics equivalent to these). The instance of the proof theory for partial stable models can also be used to compute stable models for those cases when the set of all partial stable models coincides with the set of all stable models. In LP, this is the case, in particular, when the given logic program is stratified [2] or, more generally, order-consistent [42].

The abstract proof theory thus provides a common basic computational framework for the various argumentation semantics. We use the abstract proof theory as a basis for developing top-down, goal-oriented, abstract proof procedures for argumentation. These proof theories and procedures are defined in terms of derivations of trees, whose nodes are sets of hypotheses such that every node attacks its parent node. The proof procedures are obtained from the proof theories by specifying special selection strategies (of the nodes in the trees) and particular ways of computing the attack relation. Specific proof procedures for the different argumentation semantics can then be obtained by appropriately instantiating the parameters of the abstract proof procedures in the same way proof theories for different argumentation semantics can be obtained by instantiating the basic proof theory. Proof procedures for other non-monotonic logics could be developed from the abstract proof theories in an analogous by appropriately modifying the computation of the attack relation.

Both proof theories and procedures developed in this paper are constructive in the sense that they can be used not only to decide whether a certain given extension is allowed by the particular semantics at hand, but also to construct allowed extensions in which a given query holds. Therefore, the proof procedures can be used for query evaluation at the same level as that of SLDNF. They (aim to) construct a set of hypotheses Δ which is allowed by the semantics and such that a given query is entailed by the extension of the program with Δ. Although most of the semantics of NAF are formulated using the propositional Herbrand grounding of the program, the top-down proof procedures developed in this paper are directly applicable to the non-propositional (first-order) form of the logic program, similarly to SLDNF.

The origin and motivation of the proof theories and procedures we propose can be traced back to the seminal abductive proof procedure for LP presented in [14]. The specific proof theories and procedures for computing partial stable models, stable theories and acceptability generalise the procedure in [14] in the sense that whatever is computed by this is also computed by our proof procedures for these semantics. Since the procedure of [14] is a generalisation of SLDNF, in the same sense, so are our proof procedures. The proof procedure for computing the well-founded model semantics is also a generalisation of SLDNF, in the same sense.

In this paper we give relatively little attention to the issue of efficiency of the computation. At this stage, we are primarily interested in developing a basic uniform computational model for argumentation semantics from which top-down proof procedures can be derived. Efficiency considerations should then follow.

The paper is organised as follows. In the next section we review the argumentation-based formulation of the semantics for LP. In section 3, we re-interpret these semantics in terms of trees of sets of hypotheses. In section 4, we present the abstract proof theory and its instances for the various LP semantics we intend to compute. The formulation of these proof theories is essentially independent from the LP formalism. In section 5, we describe an abstract proof procedure for LP, and its specific instances for its various semantics. Section 6 analyses aspects of the specific proof procedures relevant to their implementation. Section 7 compares our approach with others and section 8 concludes and discusses possible future work.

2 Argumentation semantics for logic programming

In this section we review how preferred extension [11] (or equivalently partial stable model [40], as proved in [25]), stable theory [24], acceptability [26] and well-founded [48] semantics for LP, can be formulated within an abstract argumentation framework [4, 5, 12, 20, 26, 27].

In general, an argumentation framework consists of a *theory* P in some background *monotonic logic*, a set of candidate *hypotheses* $\mathcal{H}$, and a (binary) *attacking relation* (*attacks*) between sets of hypotheses, satisfying the following fundamental properties:

- no set of hypotheses attacks the empty set of hypotheses $\emptyset$,
- attacks is *monotonic*, i.e., for all $A, A', \Delta, \Delta' \subseteq \mathcal{H}$, if A attacks Δ then

  if $A \subseteq A'$ then A' attacks Δ, and

  if $\Delta \subseteq \Delta'$ then A attacks Δ',
- attacks is *compact*, i.e., for all $A, \Delta \subseteq \mathcal{H}$, if A attacks Δ then there exists a finite $A' \subseteq A$ such that A' attacks Δ.

In this paper we will assume that the attacking relation is defined only on finite subsets of $\mathcal{H}$. Different instances of this abstract argumentation framework correspond to different non-monotonic formalisms. For the special case of LP, P is a logic program, the set of hypotheses is

$\mathcal{H}=\{not\, p \,|\, p$ is a variable-free atom in the Herbrand base of $P\}$,

and the background monotonic logic is the classical logic of (definite) Horn programs. These programs are obtained by replacing each negative literal in the program P and in the hypotheses set $\mathcal{H}$ by a new positive atom [14]. We will denote these programs by the same symbol P, and the consequence relation of this logic by $\models$. (Unless otherwise specified, we will assume that programs containing variables represent the set of all their variable-free instances over the given Herbrand universe.)

Given a program P, a conjunction of (variable-free) literals G is a *non-monotonic consequence* of P, with respect to some semantics, iff there exists a set of hypotheses Δ, allowed by the semantics, such that $P \cup \Delta \models G$.

Different semantics for LP allow different sets of hypotheses. All semantics we consider can be formulated in terms of the following single attacking relation between sets of (variable-free) negative literals (cf. [21, 22]).

Definition 1 A set of hypotheses A *attacks* another set Δ (*on* an hypothesis $not\, p$) iff $P \cup A \models p$ for some $not\, p$ in Δ.

Note that this definition satisfies all properties listed above. In particular, by compactness of $\models$, the attacking relation is compact. However, note that there might be infinitely many (finite) attacks against any set of hypotheses, as illustrated by the following example.

Example 2 Consider (the ground instantiation of) the program

$$\begin{array}{l} p \leftarrow q(x),\, not\, r(x) \\ q(succ(x)) \\ q(0) \\ r(x) \end{array}$$

and the set $\Delta_0 = \{not\, p\}$. There are infinitely many (finite) attacks against Δ_0, i.e. $\{not\, r(0)\}$, $\{not\, r(succ(0))\}$, $\{not\, r(succ(succ(0)))\}$, etc.

In the reminder of this section, we will define the various argumentation semantics for LP and illustrate them by means of examples, pointing at some of their main features relevant for the development of their proof theories.

Preferred extensions and admissibility semantics

Preferred extensions [11] and partial stable models [40] are obtained from maximal admissible sets of hypotheses [11]. Within the abstract argumentation framework, admissible sets are specified as follows:

Definition 3 A set of hypotheses Δ is *admissible* iff for all sets of hypotheses A, if A attacks Δ, then Δ attacks $A - \Delta$.

For the equivalence of this definition of admissibility and the original definition in [11] see [21, 22].

Intuitively, Δ is admissible iff Δ *defends itself* against each attack A. Note that a set of hypotheses that attacks itself can not be admissible, because there is no attack against the empty set.

In this paper we will concentrate on the computation of admissible sets of hypotheses, rather than preferred extensions. In fact, every admissible set of hypotheses is "contained" in some preferred extension (see [11, 5]), and thus, in order to determine whether a given query holds with respect to the preferred extension and partial stable model semantics, it is sufficient to determine whether the query holds with respect to the semantics of admissible sets.

Example 4 Consider the program

$$\begin{array}{l} s \leftarrow not\, r \\ r \leftarrow not\, s. \end{array}$$

$\Delta_1 = \{not\, s\}$ is admissible, since every attack against Δ_1 is a superset of $\Delta_2 = \{not\, r\}$, which is attacked by Δ_1 itself. Similarly, Δ_2 is admissible. However, $\Delta_1 \cup \Delta_2$ is not admissible, since it attacks itself. Therefore, both Δ_1 and Δ_2 are maximal admissible, and correspond to preferred extensions of P.

This example shows that a program may admit several admissible (and preferred) sets of hypotheses which may be incompatible with each other. Any proof theory for the admissibility/preferred extension semantics would therefore need to allow a non-deterministic choice amongst such sets.

Example 5 Consider the program of example 4 extended by

$$q \leftarrow not\, s.$$

$\Delta_0 = \{not\, q\}$ is not admissible, since it does not attack the attack $\{not\, s\}$ against it. However, the set $\Delta_0 \cup D$, with $D = \{not\, r\}$, is admissible, because it attacks the attack $\{not\, s\}$ (and hence all attacks against $\Delta_0 \cup D$, since every such attack contains $not\, s$). Therefore, by extending Δ_0 with D, we construct the admissible (and preferred) set $\{not\, q, not\, r\}$.

The set D in this example plays the role of a **defence** for Δ_0. In general, as we will see later, the construction of an admissible set Δ can be done *incrementally*, starting from a given set of hypotheses Δ_0, by adding to Δ_0 suitable defences for it. The existence of several admissible (and preferred) sets of hypotheses reflects itself on the existence of several suitable defences for a given Δ_0, and imposes a non-deterministic choice amongst defences in the proof theory.

Example 6 Consider the program of example 4 extended by

$$\begin{aligned} t &\leftarrow not\, q \\ q &\leftarrow not\, s \\ q &\leftarrow not\, r. \end{aligned}$$

$\Delta_0 = \{not\, t\}$ is not admissible, but both $\Delta_0 \cup D_1$, with $D_1 = \{not\, r\}$, and $\Delta_0 \cup D_2$, with $D_2 = \{not\, s\}$, are admissible (and preferred).

In general, however, not every potential defence can be promoted to Δ_0, as illustrated by the following example.

Example 7 Consider the program

$$\begin{aligned} s &\leftarrow not\, r \\ r &\leftarrow not\, r. \end{aligned}$$

The set $\Delta_0 = \{not\, s\}$ is not admissible, since the attack $A = \{not\, r\}$ against Δ_0 is not attacked by Δ_0. The set $D = \{not\, r\}$ is a possible defence for Δ_0, since it attacks A. Moreover, D is contained in every possible attack against A. However, $\Delta_0 \cup D$ is not admissible since D, and therefore $\Delta_0 \cup D$, attacks itself (there is no admissible superset of Δ_0).

Stable theories and weak stability semantics

Stable theories are obtained from maximal weakly stable sets of hypotheses [24]. Within the abstract argumentation framework, these are specified as follows:

Definition 8 A set of hypotheses Δ is *weakly stable* iff for all sets of hypotheses A, if A attacks Δ, then $\Delta \cup A$ attacks $A - \Delta$.

For the equivalence of this definition of weak stability and the original definition in [24] see [21, 22].

Intuitively, the attack A against Δ can be used in conjunction with Δ to defend Δ against A. Note that, similarly to the case of admissible sets, a set of hypotheses that attacks itself can not be weakly stable.

Trivially, every admissible set of hypotheses is weakly stable, but not vice versa, as illustrated by example 7. We have already seen that $\Delta = \{not\, s\}$ is not admissible for the program in this example. However, Δ is weakly stable, since every attack A against Δ is a superset of $\{not\, r\}$, which is attacked by $\{not\, r\}$ itself and thus by $\Delta \cup A$. Δ is also maximal weakly stable and corresponds to a stable theory of P.

As for admissibility and preferred extension semantics, it can be proved that every weakly stable set of hypotheses is "contained" in some stable theory. Therefore, we will concentrate on the computation of weakly stable sets of hypotheses, rather than stable theories.

Acceptability semantics

Admissibility and weak-stability semantics are special cases of the more general argumentation semantics of acceptability. Formally, acceptability is defined as the least fixed point of an operator $\mathcal{F}$ on the set of all binary relations on the set of hypotheses $\mathcal{H}$, (see [26] for the details). This operator is extracted from the following informal recursive specification of acceptability. [1]

Definition 9 A set of hypotheses Δ is *acceptable to* another set of hypotheses Δ_0 iff for all sets of hypotheses A, if A attacks $\Delta - \Delta_0$, then there exists a set of hypotheses D such that D attacks $A - (\Delta_0 \cup \Delta)$ and D is **acceptable to** $A \cup \Delta_0 \cup \Delta$.
A set of hypotheses is **acceptable** iff it is acceptable to the empty set of hypotheses.

Hence acceptability is a binary relation on the set of hypotheses $\mathcal{H}$ with the central notion of "Δ is acceptable to Δ_0", where Δ_0 can be seen as a given choice of hypotheses (or context) under which the acceptability of Δ is to be examined.

The base case for the above recursive definition is given by

(BC1) a set of hypotheses Δ is acceptable to another set Δ_0 if there exists no set of hypotheses A attacking $(\Delta - \Delta_0)$.

In particular, since there are no attacks against the empty set of hypotheses, BC1 occurs when $\Delta - \Delta_0 = \emptyset$, giving

(BC2) Δ is acceptable to Δ_0 if $\Delta \subseteq \Delta_0$.

[1] In [26] this definition was further motivated by the (informal) specification "A set of hypotheses Δ is **acceptable to** another set of hypotheses Δ_0 iff for all sets of hypotheses A, if A attacks $\Delta - \Delta_0$, then A is **not acceptable to** $\Delta_0 \cup \Delta$", which gives definition 9 when we unfold its recursive "A is not acceptable to $\Delta_0 \cup \Delta$".

Note that a set of hypotheses Δ that attacks itself can not be acceptable, because there is no attack against the empty set.

The formulation of acceptability in definition 9 directly leads to (see appendix A for a formal proof):

Corollary 10 Weakly stable and admissible sets of hypotheses are acceptable.

Similarly to the case of admissible and weakly stable sets, the set D in the formulation of the acceptability semantics plays the role of a defence against the attack A. The proof theory will construct acceptable sets Δ by incrementally augmenting in a suitable way initial subsets of Δ with (hypotheses in) defences, in order to guarantee Δ's acceptability. Moreover, again the choice of suitable defences is (don't know) non-deterministic, and some possible defences might turn out to be unsuitable. However, in the case of acceptable sets there is an extra-level of (don't know) non-determinism, to separate which (parts of) chosen defences can be included in the final Δ. This is illustrated by the following

Example 11 Given the program

$$p \leftarrow not\, q$$
$$q \leftarrow not\, t$$

both $\Delta_1 = \{not\, p, not\, t\}$ and $\Delta_2 = \{not\, p\}$ are acceptable. Indeed, it is easy to see that Δ_1 is admissible, and therefore acceptable. Moreover, all attacks against Δ_2 contain $not\, q$, and can be shown to be not acceptable to Δ_2 by exploiting the fact that the attack $\{not\, q\}$ is not acceptable to Δ_2, proved as follows:

- $\{not\, q\}$ is acceptable to Δ_2 only if there exists a defence D, attacking $A - (\{not\, q\} \cup \Delta_2)$, where $A = \{not\, t\}$ attacks $\{not\, q\} - \Delta_2 = \{not\, q\}$, and such that D is acceptable to $\{not\, q\} \cup \Delta_2 \cup A$.
- But $A - (\{not\, q\} \cup \Delta_2) = \{not\, t\}$ and there is no attack against $\{not\, t\}$.

In this example, given the initial set $\Delta_2 = \{not\, p\}$, the potential defence $\{not\, t\}$ chosen to prove that the attack $\{not\, q\}$ against Δ_2 is not acceptable to Δ_2, might or might not be added to Δ_2 in order to generate an acceptable set, since in both cases the resulting set is acceptable.

Example 12 Given the program

$$p \leftarrow not\, q$$
$$q \leftarrow not\, r$$
$$r \leftarrow not\, s$$
$$s \leftarrow not\, q$$

$\Delta = \{not\, p\}$ is acceptable. Indeed, all attacks against Δ contain $not\, q$, and can be shown to be not acceptable to Δ by exploiting the fact that the attack $\Delta_1 = \{not\, q\}$ is not acceptable to Δ, proved as follows:

- Δ_1 is acceptable to Δ only if there exists a defence D, attacking $A-(\Delta_1 \cup \Delta) = \{not\,r\}$, where $A = \{not\,r\}$ attacks $\Delta_1 - \Delta$, and such that D is acceptable to $A \cup \Delta_1 \cup \Delta = \{not\,r,\ not\,q,\ not\,p\}$.
- Any such D must contain *not s*, and is acceptable to $A \cup \Delta_1 \cup \Delta$ only if there exists a defence D', attacking $A' - (A \cup \Delta_1 \cup \Delta)$, with $A' = \{not\,q\}$ attacking $D - (A \cup \Delta_1 \cup \Delta)$, and such that D' is acceptable to $A \cup \Delta_1 \cup \Delta \cup A' \cup D$.
- But $A' - (A \cup \Delta_1 \cup \Delta) = \emptyset$, and therefore no such D' exists.

(Similarly, $\Delta_2 = \{not\,r\}$ and $\Delta_3 = \{not\,s\}$ are not acceptable.)

In this example, given the (acceptable) set $\Delta = \{not\,p\}$, the defence $\{not\,r\}$, chosen to prove that the attack $\{not\,q\}$ against Δ is not acceptable to Δ, should not be added to Δ. It is needed at an intermediate level to defend against the attack $\{not\,q\}$ and, although acceptable to $\{not\,p,\ not\,q\}$, it is not acceptable to the empty set and hence can not be added to Δ.

Well-founded model semantics

Up to now we have considered semantics of LP which are credulous in nature, allowing, in general, more than one set of hypotheses, and therefore requiring a choice to determine non-monotonic consequences of P. An important skeptical semantics for LP, allowing a single set of hypotheses, is the well-founded model semantics [48]. Within the abstract argumentation framework, the well-founded model can be formally expressed as the least fixed point of an operator $\mathcal{F}$ on the set of all unary relations on the set of hypotheses $\mathcal{H}$ (see appendix A for the details) analogously to the case of acceptability. The informal recursive specification is given as follows.

Definition 13 A set of hypotheses Δ is *wf-acceptable* iff for all sets of hypotheses A, if A attacks Δ, then there exists a set of hypotheses D such that D attacks A and D is **wf-acceptable**.

As for acceptability, the set D plays the role of a defence for Δ.

The base case for wf-acceptability is given by:

(BC) Δ is wf-acceptable if no set of hypotheses A attacks Δ.

Consider the program in example 11. Both the empty set and the set $\Delta_0 = \{not\,t\}$ are wf-acceptable, since no set of hypotheses attacks them. (Note that the empty set is always wf-acceptable, because there is no attack against it.) Therefore, the set $\Delta_1 = \{not\,p\}$ is wf-acceptable, since it can be defended by the wf-acceptable set Δ_0. Indeed, every attack against Δ_1 must contain *not q*, which is attacked by Δ_0. Finally, the set $\Delta_0 \cup \Delta_1$ is wf-acceptable.

Consider the program in example 4. $\Delta_1 = \{not\,s\}$ is not wf-acceptable. Every attack against Δ_1 contains *not r*, which can be defended only by a set of hypotheses containing Δ_1. This means that for Δ_1 to be wf-acceptable we circularly need this to be wf-acceptable and hence Δ_1 can not belong to the least-fixed point of the associated operator $\mathcal{F}$. In other words, the wf-acceptability of Δ_1 can not be reduced to the base case BC of the definition, i.e. to a set of hypotheses which is not attacked by any other set. Similarly, $\Delta_2 = \{not\, r\}$ is not wf-acceptable. The only wf-acceptable set is the empty set.

A similar situation arises for the program in example 7. The set $\{not\, r\}$ is not wf-acceptable, since it attacks itself and every possible defence must contain $not\, r$ itself. Again the base case BC cannot be reached. Moreover, the set $\{not\, s\}$ is not wf-acceptable, since every set defending it must contain $not\, r$, which as we have just argued cannot be wf-acceptable.

The following theorem shows the relation of wf-acceptability to the well-founded model of [48] (see appendix A for its proof).

Theorem 14 Given a logic program P:

- If $M = (I^+, I^-)$ is the well-founded model of P, then $\Delta_M = \{not\, p \,|\, p \in I^-\}$ is maximal wf-acceptable.
- If Δ is wf-acceptable and (I^+, I^-) is the well-founded model of P, then $I^+_\Delta = \{p \,|\, p \in$ Herbrand base of P and $P \cup \Delta \models p\} \subseteq I^+$ and $I^-_\Delta = \{p \,|\, not\, p \in \Delta\} \subseteq I^-$.

As a direct corollary of the second item, if Δ is a maximal wf-acceptable set of hypotheses, then (I^+_Δ, I^-_Δ) is the well-founded model of P.

Since each wf-acceptable set of hypotheses is "contained" in the well-founded model, we can restrict attention to the computation of wf-acceptable sets of hypotheses.

The following provides a useful (and easy to prove) property of the well-founded semantics relevant to the definition of proof theory and procedure.

Lemma 15 A set of hypotheses Δ is wf-acceptable iff each set of hypotheses $\Delta' \subseteq \Delta$ is wf-acceptable.

The two recursive formulations of the sceptical well-founded model semantics and of the credulous acceptability semantics are very similar. The essential difference between these two semantics is that, whereas in the case of acceptability defences are justified in the context of sets of hypotheses (the original Δ, other defences, and even attacks), in the case of wf-acceptability all defences must be *universally* justified, irrespectively of the hypotheses they defend or attack, by eventually reducing them to defences with no attacks against them.

As a result, whereas the proof theory for acceptability (possibly) extends the initial set of hypotheses Δ_0 by means of some defence D (as illustrated in example 5), the proof theory for wf-acceptability does not extend Δ_0, but keeps generating (nested) defences until no attack exists against some defence. Notice that in this case both the final defence and each "earlier" defences are wf-acceptable and hence the union of all such defences is also wf-acceptable.

Other semantics

A number of other semantics for NAF in LP have been proposed. Some have been proved to correspond to some of the semantics considered so far and can be computed by adopting the methods we propose in the next sections. For example:

- You and Yuan's regular model semantics [49] is equivalent to partial stable model semantics [40] (see [49]),
- Torres' argumentation-theoretic semantics [47] is equivalent to stable theory semantics [24] (see [22]).

Other semantics can be defined in argumentation terms, using our notion of attack. For example, complete scenaria [11] and stationary expansions [37] correspond (see [12, 4, 5]) to **complete** sets of hypotheses, defined as sets of hypotheses that are admissible and contain every hypothesis they "defend" (see [12, 4, 5] for a formal definition). Since every admissible set of hypotheses is (by definition) contained in a complete set, the semantics corresponding to complete sets can be computed by adopting the methods we propose in the next sections for admissibility.

Stable models [17] can be also defined in argumentation terms via "stable" sets of hypotheses: (See [12, 4, 5] for the formal correspondence between stable models and stable sets of hypotheses.)

Definition 16 A set of hypotheses Δ is *stable* iff Δ does not attack itself and Δ attacks each hypothesis not in Δ [2].

Unlike all the other semantics discussed so far in the paper, the stable model semantics does not satisfy the property that "models" of "self-contained" subprograms can always be extended to "models" of the whole program. (Intuitively, a subprogram is "self-contained" iff no predicate "referred to" in the subprogram are defined in the full program outside the subprogram.) As a consequence, query evaluation under the stable model semantics is computationally expensive, as in general one needs to consider the whole program independently of the particular query at hand and of the "self-contained" subprograms affecting the evaluation of the query.

For example, let us denote the program in example 7 as P_1 and the program in example 11 as P_2, and consider $P = P_1 \cup P_2$. P_1 and P_2 are "self-contained", since all predicates "referred to" in P_1 (i.e. s and r) and P_2 (i.e. p, q and t) are defined exclusively by P_1 and P_2, respectively. P_2 admits one stable model, $M = \{q\}$, corresponding to the stable set of hypotheses $\Delta_M = \{not\,p,\, not\,t\}$. This stable model cannot be extended to any stable model of P, since P_1 and hence P admit no stable model. Therefore, while computing a stable model entailing $not\,p$, one can not restrict attention to P_2 only, but needs to take into account the unrelated subprogram P_1 of P.

Note that, in addition to being a stable model of P_2, M is also an admissible scenario and the well-founded model, and Δ_M is acceptable and weakly stable for P_2 as well as for P. Therefore, in any computation for the query $not\,p$ with respect to P, one can restrict attention to P_2 only, with no need to consider P_1.

In this paper, we will not consider explicitly the computation of stable model semantics. However, any proof procedure for computing the admissibility semantics can serve as a proof procedure for computing stable models in all cases where the set of all partial stable models is the same as the set of all stable models. This is the case, e.g., when the given logic program is stratified [2] or order-consistent [42]. Note that the programs in examples 7 and 12 are neither stratified nor order-consistent.

[2] A attacks $not\,p$ iff A attacks $\{not\,p\}$.

The program in example 4 is order-consistent but not stratified. Therefore, our proof procedure for computing the admissibility semantics gives a sound computation of the stable model semantics for these classes of programs where stable model semantics and partial stable model semantics coincide.

3 Argumentation via trees

The proof theories and procedures will be based on a formulation of the argumentation semantics in terms of and-trees. In the sequel, we will assume that the reader is familiar with basic tree-theoretic notions such as those of root, node, branch, subbranch (from one node down to another), child and parent of a node, ancestor and descendent of a node, leaf, subtree (rooted at a given node) and truncation (of a tree).

In the trees we will consider, nodes are sets of hypotheses with each node labelled either as "attack" or "defence". In the sequel, we will use the term node to refer both to a location in the tree and to the set of hypotheses at this location.

The definition of these trees is parametric with respect to a function *hyp-branch* that returns, for each given tree and each node of the tree, a set of hypotheses. Trees for admissible, weakly stable, wf-acceptable and acceptable sets of hypotheses are obtained by different choices of the function *hyp-branch*. For all such choices, the set of hypotheses returned by *hyp-branch* for a given tree and a given node in the tree is a subset of the union of all sets of hypotheses in the branch of the tree from the root down to, and excluding, the node itself. The subset returned by the function *hyp-branch* may be a strict subset of the set of all hypotheses in the branch (e.g. for the instances of the function *hyp-branch* for admissible and well-founded semantics). Therefore, the function *hyp-branch* is distinct from the tree-theoretic notion of branch.

In the remainder of the paper, unless otherwise stated, we will assume that a program P and its associated set of hypotheses $\mathcal{H}$ are given.

3.1 General trees

Definition 17 Let *hyp-branch* be any function such that given a tree with sets of hypotheses at its nodes and a node of the tree, returns a set of hypotheses. Then, a *pre-tree for a set of hypotheses* $\Delta \subseteq \mathcal{H}$ is a labelled tree $\mathcal{T}$ where

(PT1) each node is a set of hypotheses, labelled either as **attack** or **defence**, but not both;

(PT2) the root is Δ, labelled as defence;

(PT3) every node A labelled as attack has (exactly) one child D, labelled as defence, and D attacks $A - \mathit{hyp\text{-}branch}(\mathcal{T}, A)$;

(PT4) every node D labelled as defence has as children all sets of hypotheses A such that A attacks $D - \mathit{hyp\text{-}branch}(\mathcal{T}, D)$, and each such child A is labelled as attack;

(PT5) every branch of $\mathcal{T}$ is finite.

Hence, in each branch of a pre-tree, defence and attack nodes alternate: a defence node can only be followed by an attack node, and vice versa. For convenience, we will assign levels to nodes in pre-trees, with the root being of level 0 and each node that is child of a node of level i being of level $i+1$. Then, in a pre-tree nodes of even level are labelled as defence and nodes of odd level are labelled as attack.

Moreover, pre-trees can be infinite in breadth, since there may be infinitely many attacks against any given set of hypotheses (see example 2). However, by (PT5), every pre-tree is forced to be finite in depth.

Finally, every node of a pre-tree, except the root, attacks its parent node, and no node of the tree attacks any hypothesis in the set returned by *hyp-branch* for the parent of that node. Namely, the role of the function *hyp-branch* is to identify the part of nodes that should be attacked by their children.

Definition 18 For every node N in a pre-tree $\mathcal{T}$, $N - \mathit{hyp\text{-}branch}(\mathcal{T}, N)$ is the *weak part* of N. If N is an attack node, then any hypothesis in the weak part of N attacked by the (single) child of N in $\mathcal{T}$ is a *culprit* of N.

Note that the same set of hypotheses can attack more than one hypothesis in another set. Therefore, there might be more than one culprit in any attack node in (a subtree of) a pre-tree. It will be useful to assume the existence of a function *culprits* that records the set of culprits (chosen) in a subtree of a pre-tree.

Definition 19 A *culprit function* is any function *culprits* from the set of subtrees of pre-trees to the power set of $\mathcal{H}$, such that, for any given subtree $\mathcal{T}$ and any attack node A in $\mathcal{T}$, $\mathit{culprits}(\mathcal{T})$ contains exactly one culprit of A.

The function *culprits* will be used to record a particular choice of culprits at every attack node.

Condition (PT3) in definition 3.1 of pre-tree implies that no node labelled as attack may be a leaf of a pre-tree. Therefore, leaves of a pre-tree are necessarily defence nodes whose weak part cannot be attacked. In particular, a defence node D is a leaf if $D - \mathit{hyp\text{-}branch}(D) = \emptyset$.[3] We refer to the level of the defence node leaf of the longest branch of a pre-tree as the **depth** of the pre-tree. For example, a pre-tree consisting of the root only has depth 0. Since defence nodes have even level, every pre-tree is necessarily of even depth.

Figures 1 (a) and (b) present trees with roots the sets of hypotheses $\{not\, q\}$ and $\{not\, p\}$, respectively, with respect to the program P in example 11. Here and in all figures in the paper we omit explicit labels for defence and attack nodes. When an attack node contains more than one hypothesis we indicate its culprit, chosen by the given function *culprits*, by underlying it.

The tree in (a) is not a pre-tree for $\{not\, q\}$, for any choice of the function *hyp-branch*, because the attack node $\{not\, t\}$ is a leaf (thus condition (PT3) is violated). Furthermore, for some choice of *hyp-branch*, e.g. $\mathit{hyp\text{-}branch}(N) = \emptyset$ for every node N,[4] there exist attacks against (the weak part of) the root and defence node (of level 2) $\{not\, q\}$ that do not appear amongst their children in the tree, e.g. $\{not\, t, not\, q\}$ attacks $\{not\, q\}$ but is not a child of the root or of the given defence node (thus condition (PT4) is violated). Note that, whereas the tree could be expanded to satisfy condition (PT4), condition (PT3) cannot be satisfied, since there is no attack against the only possible culprit in the attack node $\{not\, t\}$.

[3] When no confusion arises, we drop the tree argument in the function *hyp-branch*.

[4] This choice of *hyp-branch* will be adopted for the well-founded model semantics, see section 3.1.1.

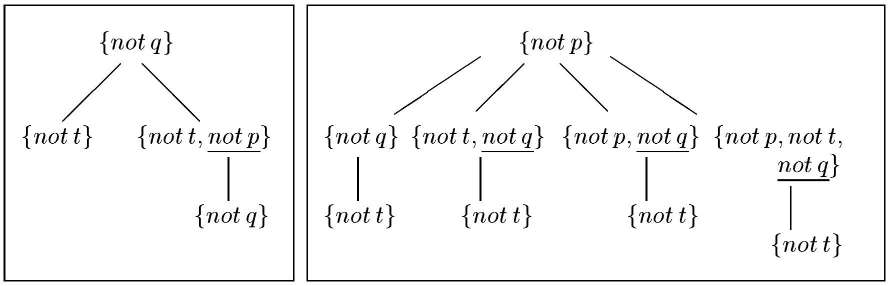

Figure 1: (a) Non-pre-tree and (b) pre-tree, for example 11 ($P = \{p \leftarrow not\, q; q \leftarrow not\, t\}$).

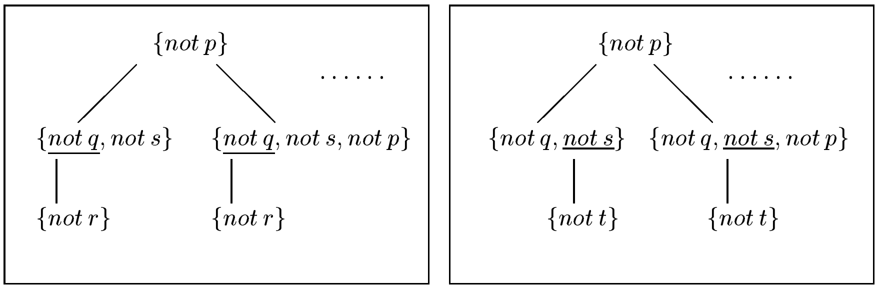

Figure 2: Two alternative pre-trees for $\Delta = \{not\, p\}$ in example 20 ($P = \{p \leftarrow not\, q, not\, s; q \leftarrow not\, r; s \leftarrow not\, t\}$).

The tree in figure 1 (b) is a pre-tree for $\{not\, p\}$, assuming, as above, that $hyp\text{-}branch(N){=}\emptyset$ for every node N. Here, note the same culprit in all four attack nodes and the same defence node child of each such attack node. In section 3.2, we will see that only the (minimal) attack node $\{not\, q\}$ needs to be considered, and therefore only the left-most branch of this tree needs to be built by the proof theory.

For any set of hypotheses Δ, there could exist no pre-trees (see $\Delta = \{not\, q\}$ in figure 1, part (a)), one pre-tree (see $\Delta = \{not\, p\}$ in figure 1 (b)) [5] or many pre-trees. The multiplicity of pre-trees arises from the non-determinism in the choice of defences, which stems from the non-determinism in the choice of culprits in attack nodes and then the non-determinism in the choice of a particular set of hypotheses attacking each of the chosen culprits.

Example 20 Consider the program

$$
\begin{aligned}
p &\leftarrow not\, q, not\, s \\
q &\leftarrow not\, r \\
s &\leftarrow not\, t
\end{aligned}
$$

and the set of hypotheses $\Delta = \{not\, p\}$. Figure 2 sketches two possible pre-trees for Δ, with respect to $hyp\text{-}branch(N){=}\emptyset$ for every node N. The trees are incomplete (shown by the dots), since not all (non-minimal) attacks against the root, $\{not\, p\}$,

[5] Note that the pre-tree in figure 1 (b) is the only possible pre-tree for $\Delta = \{not\, p\}$ in example 11.

are explicitly considered. Any such missing attack node has the same child as the minimal attack node it is a superset of.

The different argumentation semantics given in section 2 can be equivalently formulated in terms of pre-trees with respect to special choices of the function *hyp-branch*, and, in the case of admissibility and weak stability, satisfying some additional properties. In the reminder of this section we will present these different "semantic" specialisations of pre-trees.

3.1.1 Well-founded semantics

Pre-trees for wf-acceptable sets of hypotheses are special instances of general pre-trees in definition 17 in that, for every node, *hyp-branch* returns the empty set of hypotheses.

Definition 21 Given a pre-tree $\mathcal{T}$, $hyp\text{-}branch_0(\mathcal{T}, N){=}\emptyset$ for every node N in $\mathcal{T}$. A tree $\mathcal{T}$ is a *well-founded pre-tree* for a set of hypotheses Δ iff it is a pre-tree for Δ with the function *hyp-branch* given by $hyp\text{-}branch_0$.

The pre-trees in figure 1 (b) and figure 2 are well-founded pre-trees. The notion of well-founded pre-tree is further illustrated by the following examples.

Example 22 Consider the program obtained by adding to the program in example 11 the clauses

$$u \leftarrow not\, s$$
$$s \leftarrow not\, p$$

Figure 3(a) illustrates a well-founded pre-tree for the set of hypotheses $\{not\, u\}$. (Again, we omit parts of the tree corresponding to some non-minimal attacks, shown by the dots in the figure.)

Note that well-founded pre-trees can be of any (even, finite) depth. The tree in figure 3(b) is not a well-founded pre-tree because it is not finite. Note also that any subtree of a well-founded pre-tree rooted at a defence node is itself a well-founded pre-tree.

Theorem 23 A set of hypotheses Δ is wf-acceptable iff there exists a well-founded pre-tree for Δ.

The proof of this theorem can be found in appendix B.

Proof theory and procedures for computing the well-founded model semantics will be based on the construction of well-founded pre-trees. Some properties can be used to render this construction more efficient.

Theorem 24 Given a well-founded pre-tree $\mathcal{T}$, for each defence node D in $\mathcal{T}$, $D \cap culprits(\mathcal{T}){=}\emptyset$.

The proof of this theorem can be found in appendix B.

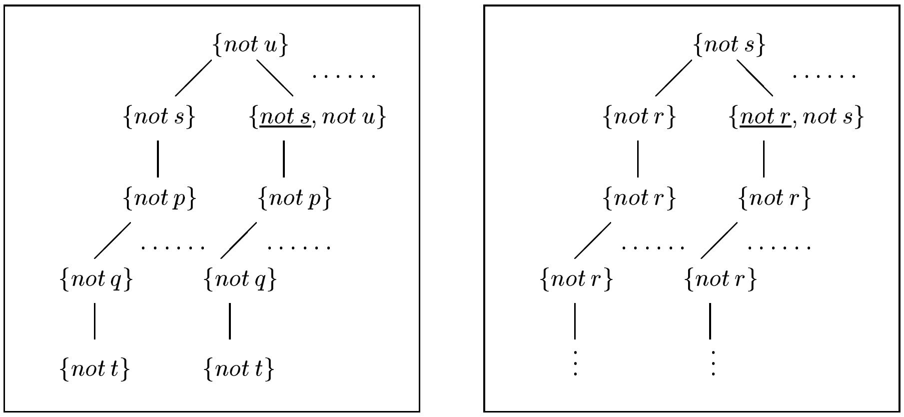

Figure 3: (a) Well-founded pre-tree for example 22 ($P = \{u \leftarrow not\, s; s \leftarrow not\, p; p \leftarrow not\, q; q \leftarrow not\, t\}$), and (b) non-well-founded pre-tree for example 7 ($P = \{s \leftarrow not\, r; r \leftarrow not\, r\}$).

Corollary 25 Given a well-founded pre-tree $\mathcal{T}$, let $\mathcal{D}$ be the union of all defence nodes in $\mathcal{T}$. Then, for every attack node A in $\mathcal{T}$, the defence node D child of A in $\mathcal{T}$ is such that D attacks $A - \mathcal{D}$.

In particular, no defence node can attack another defence node in a well-founded pre-tree.

These properties can be used to reduce the non-determinism in the choice of culprits in attacks, when constructing well-founded pre-trees (see section 4).

3.1.2 Admissibility and weak stability

Pre-trees for admissible and weakly stable semantics are special instances of the general pre-trees in definition 17, satisfying an additional condition imposed on defence nodes. Pre-trees for admissible and weakly stable semantics differ in the choice of the function *hyp-branch*, given as follows:

Definition 26 Given a pre-tree $\mathcal{T}$ for a set of hypotheses Δ, and a node N in $\mathcal{T}$ such that $N_0 = \Delta$, N_1, ..., $N_{i+1} = N$ is the branch of $\mathcal{T}$ from the root down to N:
hyp-branch$_1(\mathcal{T}, N) = \bigcup_{j=0}^{j=\lfloor \frac{i}{2} \rfloor} N^{2j}$;
hyp-branch$_2(\mathcal{T}, N) = \bigcup_{j=0}^{j=i} N^{j}$.

Namely, the function *hyp-branch*$_1$ associates to a node the set of all defence nodes in the branch down to and excluding the given node, whereas the function *hyp-branch*$_2$ associates to a node the set of all (attack and defence) nodes in the branch down to and excluding the given node. Note that *hyp-branch*$_1(\mathcal{T}, \Delta) =$ *hyp-branch*$_2(\mathcal{T}, \Delta) = \emptyset$.

Definition 27 A tree $\mathcal{T}$ is an *admissible pre-tree* (respectively *weakly stable pre-tree*) iff it is a pre-tree for a set of hypotheses Δ and

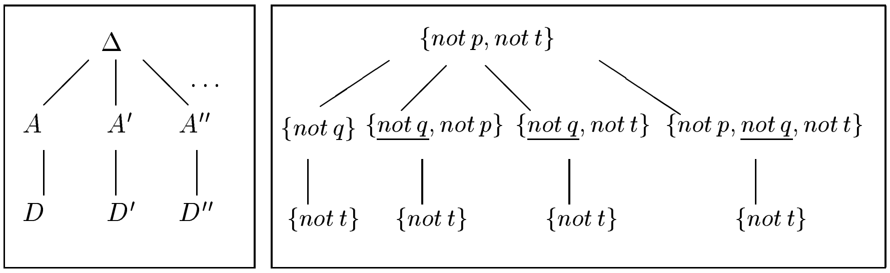

Figure 4: (a) Generic admissible/weakly stable pre-tree (of depth 2 at most), and (b) admissible/weakly stable pre-tree for $\{not\, p, not\, t\}$ in example 11 ($P = \{p \leftarrow not\, q; q \leftarrow not\, t\}$).

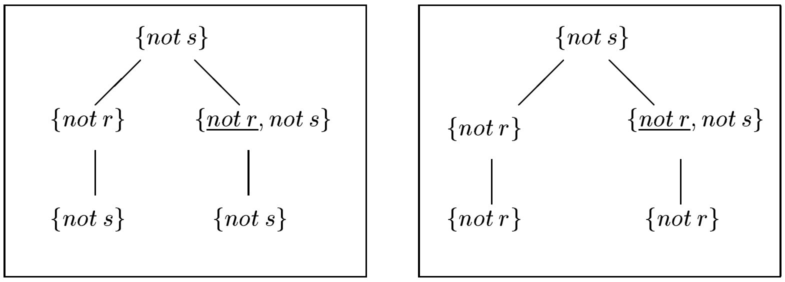

Figure 5: (a) Pre-tree for example 4 ($P = \{s \leftarrow not\, r; r \leftarrow not\, s\}$), and (b) Pre-tree for example 7 ($P = \{s \leftarrow not\, r; r \leftarrow not\, r\}$).

- the function *hyp-branch* is given by $hyp\text{-}branch_1$ (resp. $hyp\text{-}branch_2$), and
- $\mathcal{T}$ satisfies the condition that $D - hyp\text{-}branch(D) = \emptyset$ holds for every defence node D different from the root.

Note that, due to this extra condition and since there are no attacks against the empty set of hypotheses, a pre-tree is admissible or weakly stable only if it is of depth 0 or 2 (see figure 4(a)). However, not all pre-trees of depth 0 or 2 are admissible/weakly stable. For example, the pre-tree in figure 1(b) is neither admissible nor weakly stable, because, for either choices of *hyp-branch* in definition 27, such tree does not satisfy the condition $D - hyp\text{-}branch(D) = \emptyset$ for every defence node D. Note, however, that the same tree with the root extended with the hypothesis $not\, t$ is both admissible and weakly stable (see figure 4(b)). In general, the strategy of extending the root of (partially constructed) trees to obtain admissible/weakly stable pre-trees will be employed in the proof theory for computing admissible/weakly stable sets of hypotheses.

The pre-tree in figure 4(b) is also well-founded, because there are no attacks against the leaves of this tree. Figure 5(a) shows an admissible/weakly stable pre-tree, for the program in example 4, which is not well-founded, because there are attacks against its leaves. The pre-tree is admissible (respectively weakly stable) because every leaf

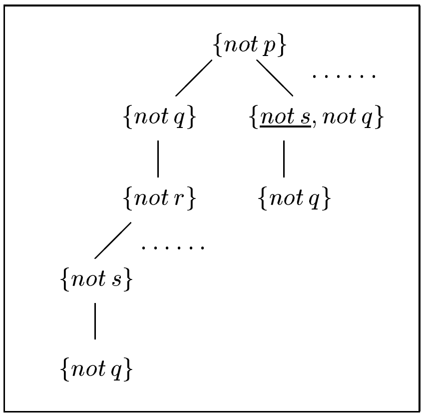

Figure 6: Acceptable pre-tree for example 12 ($P = \{p \leftarrow not\, q; q \leftarrow not\, r; r \leftarrow not\, s; s \leftarrow not\, q\}$).

D is contained in $hyp\text{-}branch_1(D)$ (respectively $hyp\text{-}branch_2(D)$). Finally, figure 5(b) shows a weakly stable pre-tree, for the program in example 7, which is not admissible (nor well-founded).

Theorem 28 A set of hypotheses Δ is admissible (respectively weakly stable) iff there exists an admissible (respectively weakly stable) pre-tree for Δ.

The proof of this theorem can be found in appendix B.

3.1.3 Acceptability semantics

Definition 29 A tree $\mathcal{T}$ is an *acceptable pre-tree* iff $\mathcal{T}$ is a pre-tree for a set of hypotheses with the function *hyp-branch* given by $hyp\text{-}branch_2$.

Therefore the definition of the function *hyp-branch* is the same as that for weakly stable pre-trees and, as expected due to corollary 10, acceptable pre-trees are a generalisation of weakly stable pre-trees, which are necessarily of depth 2 at most, to trees of any depth.

All pre-trees in the figures presented so far are acceptable. Figure 6 shows (a subtree of) an acceptable pre-tree which is neither well-founded nor admissible nor weakly stable. (The dots in the figure stand for subtrees with non-minimal attacks as roots.)

Theorem 30 A set of hypotheses Δ is acceptable iff there exists an acceptable pre-tree for Δ.

The proof of this theorem can be found in appendix B.

3.2 Restricted trees

Definition 17 of pre-trees requires that *all* attacks against (the weak part of) defence nodes are considered. Since every superset of an attack (against a given set) is itself an attack (against the set), if the Herbrand base of the given logic program is infinite then there are always infinitely many attacks to be considered, when building pre-trees. Fortunately, though, not all attacks actually need to be considered. The particular set of attacks that is sufficient to consider changes depending on the semantics. We will refer to any such set as a *restricted set of attacks*. Informally, this restricted set amounts to the set of all *minimal attacks* in the case of the admissible, weakly stable and well-founded semantics, and to the set of *minimal and extended* attacks (see definition 34) in the case of acceptability.

3.2.1 Well-founded, admissibility and weak stability semantics

Every attack against a set of hypotheses Δ is a superset of a minimal (with respect to set inclusion) attack against Δ. Theorem 32 below shows that, in order to check whether a set of hypotheses Δ is wf-acceptable, admissible or weakly stable, one only needs to construct the subtree of a well-founded, admissible or weakly stable, respectively, pre-tree for Δ where the set of all attack nodes children of each single defence node contains (or, as a special case, coincides with) the set of all minimal attacks. [6]

Definition 31 A tree $\mathcal{T}$ with root Δ is called a *well-founded* (respectively *admissible* or *weakly stable*) **restricted tree** iff $\mathcal{T}$ is a subtree of a well-founded (respectively admissible or weakly stable) pre-tree satisfying conditions (PT1), (PT2), (PT3) and (PT5) in definition 3.1 and condition

(RT4) every node D in $\mathcal{T}$, labelled as defence, has as children (at least) all *minimal* attacks against $D-hyp\text{-}branch(\mathcal{T},D)$, and each such child is labelled as attack.

Theorem 32 A set of hypotheses Δ is wf-acceptable (respectively admissible or weakly stable) iff there is a well-founded (respectively admissible or weakly stable) restricted tree for Δ.

The proof of this theorem can be found in appendix C.

Theorem 24 and corollary 25 for well-founded pre-trees continue to hold for well-founded restricted trees and, thus, well-founded restricted trees retain useful properties for the development of proof theories and procedures.

3.2.2 Acceptability semantics

In the case of acceptability the restricted set of attacks can not amount to the set of all minimal attacks only, as illustrated by following example.

[6] We will see that the proof theory, in section 4, computes, for each defence node, exactly the set of all minimal attacks, whereas the proof procedure, in section 5, computes, for each defence node, a superset of the set of all minimal attacks, namely those attacks that are computed via SLD resolution extended with abduction.

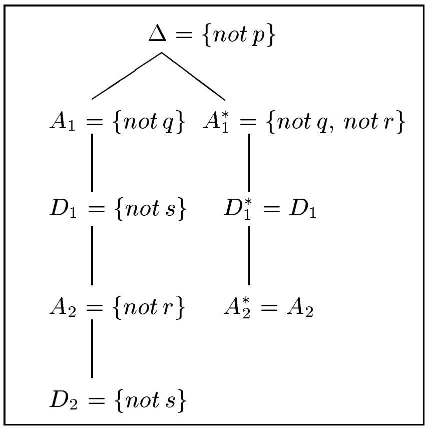

Figure 7: Minimal attacks are not sufficient for acceptability, for example 33 ($P = \{p \leftarrow not\, q; q \leftarrow not\, s; s \leftarrow not\, r; r \leftarrow not\, s\}$).

Example 33 Consider the program in example 4 extended by

$$p \leftarrow not\, q$$
$$q \leftarrow not\, s.$$

$\Delta = \{not\, p\}$ is not acceptable. Figure 7 shows a tree with Δ as root. This tree is not a pre-tree since its right branch terminates with an attack node, A_2^* (whichever culprit is chosen in A_1^*). Restricting attention to minimal attacks would give only the left branch, since A_1 is the only minimal attack against Δ. This branch terminates with a defence node, D_2, and hence, if the restricted set of attacks did coincide with the set of all minimal attacks, the set Δ would be unsoundly "accepted".

However, there is a subset of the set of all attacks that is sufficient to consider in the construction of trees in order to guarantee acceptability. Attacks in this set are built from minimal attacks and the culprits in attacks occurring in the branches below the minimal attacks in a given (partial) tree. In example 33, it is sufficient to consider only the additional, non-minimal attack A_1^*, constructed from the minimal attack A_1 and the culprit $not\, r$ in A_2, to detect that Δ is not acceptable.

Definition 34 A tree $\mathcal{T}$ with root Δ is called an *acceptable restricted tree* iff $\mathcal{T}$ is a subtree of an acceptable pre-tree satisfying conditions (PT1), (PT2), (PT3) and (PT5) and condition

(RT4) every node D in $\mathcal{T}$, labelled as defence, has as children (at least) all sets of hypotheses A satisfying the following conditions:

- A is a *minimal* attack against D–*hyp-branch*$(\mathcal{T}, D)$, or
- A is an *extended attack*, i.e. $A = A' \cup C$, where A' is a (minimal or extended) attack child of D in $\mathcal{T}$, $A \neq A'$ and C is a finite subset of *culprits*$(\mathcal{T}_{A'})$ where $\mathcal{T}_{A'}$ is the subtree of $\mathcal{T}$ with root A',

and each such child is labelled as attack.

Theorem 35 A set of hypotheses Δ is acceptable iff there exists an acceptable restricted tree for Δ.

The proof of this theorem can be found in appendix C

4 Proof theories

In this section we develop a proof theory for each of the different semantics of LP. Each proof theory can be used to decide whether a given (variable-free) goal holds with respect to the given semantics. The different proof theories are based upon a common computational framework, parametric with respect to the function *hyp-branch* and a relation *extend-root* (see below). Different instances of the parameters correspond to different semantics.

We will use the generic term *allowed* to refer to any of the terms well-founded, admissible, weakly stable or acceptable. Moreover, we will drop the qualification well-founded, admissible, weakly stable or acceptable from all kinds of trees we will consider, whenever this does not create confusion.

Definition 36 A *goal* is a set of literals.
Given a *variable-free* goal G (and a *variable-free* logic program P):
a *pre-solution* for G is a set $\Delta \subseteq \mathcal{H}$ such that $P \cup \Delta \models G$;
a *solution* for G is an allowed pre-solution for G.

The proof theories then aim to detect whether a given goal G admits solutions. In this section we will assume that a pre-solution Δ_0 for G is given (the computation of Δ_0 will be discussed in the next section 5) and we will concentrate on the questions: "Is Δ_0 allowed?" "If not, can Δ_0 be made allowed?" or "Is there an allowed superset Δ of Δ_0?".
By monotonicity of the underlying notion of derivation $\models$, any such Δ would be a solution for G. Note that, in the case of the well-founded semantics, the question reduces to "Is Δ_0 wf-acceptable?". Indeed, any set of hypotheses is wf-acceptable iff each one of its subsets is (see theorem 15) and if a set is not wf-acceptable then no superset of the set can possibly be wf-acceptable. Instead, in the case of the admissible, weakly stable or acceptable semantics, Δ might need to be a proper extension (superset) of Δ_0.

The proof theories are based upon the construction of restricted trees, via derivations of partial trees.

Definition 37 A *partial tree* $\mathcal{T}$ is defined inductively as follows:

Base case : $\mathcal{T}$ is a truncation of a restricted tree $\mathcal{T}'$ obtained by

- replacing the root of $\mathcal{T}'$ by any one of its subsets Δ, and
- removing from $\mathcal{T}'$ each subtree $\mathcal{T}_A$ of $\mathcal{T}'$ rooted at an attack node A child of the root of $\mathcal{T}'$ such that A does not attack Δ.

Inductive case : $\mathcal{T}$ is obtained by removing any number of subtrees from a partial tree.

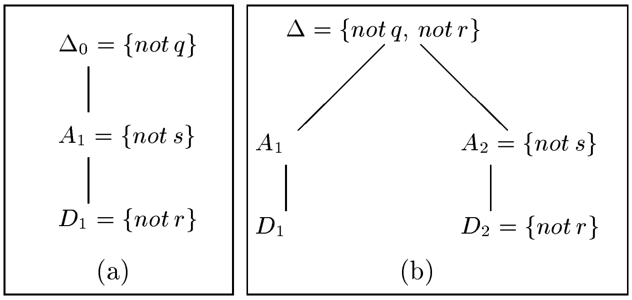

Figure 8: Δ_0 needs to be extended, for example 5 ($P = \{q \leftarrow not\, s; s \leftarrow not\, r; r \leftarrow not\, s\}$), wrt admissibility, weak stability and acceptability.

The notions of *hyp-branch* and *culprits* carry through to partial trees, in the obvious way.

Note that, as a special case, restricted trees are partial trees. Moreover, like in restricted trees, nodes of partial trees are sets of hypotheses that are labelled as attack or as defence. However, unlike in restricted trees, some of the branches of partial trees might be only partially built, i.e. leaves of partial trees might not satisfy conditions (PT3) and (RT4).

In a derivation, two main operations can be performed on a given partial tree: the partial tree can be expanded at any of its leaves or the root of the partial tree may grow, by adding hypotheses to it. This growth of the root corresponds to the extension of Δ_0 to Δ, for those semantics needing or allowing such growth. The hypotheses that are added to Δ_0 are chosen amongst the hypotheses in the weak part of defences in partial trees. In fact, by adding these hypotheses to the root Δ_0, we enlarge the set of hypotheses in the branch down to any defence node and, therefore, we potentially enlarge the set of hypotheses returned by *hyp-branch*. As a consequence, we potentially reduce the weak part of defence nodes and thus the number of attacks that needs to be considered against the defence nodes. (In the case of the well-founded semantics, *hyp-branch* to any node is always the empty set of hypotheses, and thus extending the root is not required for this semantics as this would have no effect.)

We illustrate the (possible) enlargement of the root by considering the program in example 5 and $\Delta_0 = \{not\, q\}$. The partial tree in figure 8(a) is not an admissible nor a weakly stable restricted tree. Indeed, the defence D_1 is not a subset of *hyp-branch*(D_1) (i.e. Δ_0 for admissibility, and $\Delta_0 \cup A_1$ for weak stability). However, the tree in figure 8(b) is an admissible and weakly stable restricted tree for $\Delta = \Delta_0 \cup D_1$. (Note that A_2 attacks $not\, r \in \Delta$. However, $A_2 = A_1$ and $D_2 = D_1$, therefore the right branch of the tree in figure 8(b) could actually be dropped.)

In the case of admissibility and weak stability, where trees can be of depth 0 or 2 only (due to the extra condition in definition 27), the only way to transform a partial tree into a restricted tree is to add all hypotheses in the weak part of defences to the root. In the case of acceptability, instead, the enlargement of the root is one of two options, in general. The other option is illustrated by the following example.

Example 38 Consider the program

$$p \leftarrow not\, q$$

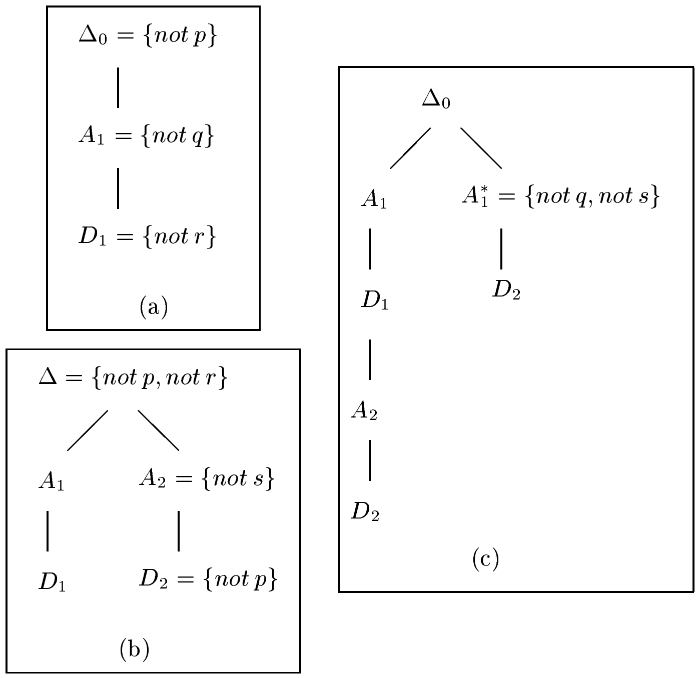

Figure 9: Δ_0 may or may not be extended, wrt acceptability, for example 38 ($P = \{p \leftarrow not\, q; q \leftarrow not\, r; r \leftarrow not\, s; s \leftarrow not\, p\}$).

$$q \leftarrow not\, r$$
$$r \leftarrow not\, s$$
$$s \leftarrow not\, p$$

and $\Delta_0 = \{not\, p\}$. The partial tree in figure 9(a) is not an acceptable tree. The trees in figure 9(b) and (c) are both acceptable. The hypothesis $not\, r$ in the (weak part of the) defence node D_1 is promoted to the root in case (b), whereas in case (c) this promotion does not take place and the branch of the tree is grown further from D_1. Note that by growing the branch the number of extended attacks to be considered grows as well (in the example we have the new extended attack A_1^*, shown in the right-most branch of the tree in figure 9(c)).

Hence, the hypothesis in the (weak part of) D_1 *may or may not* be promoted to the root.

In this example, the non-determinism given by the two options appear as "don't care". However, in general such non-determinism is "don't know". For instance, given the program in example 5, the tree in figure 8(b) is the only acceptable restricted tree derivable from the partial tree in figure 8(a), namely $not\, r \in D_1$ *should* be promoted to the root. If we consider the program in example 12, instead, and $\Delta_0 = \{not\, p\}$, neither the defence $\{not\, r\}$, nor the defence $\{not\, q\}$ (see figure 6) nor their union should be promoted to the root, since Δ_0 is the only acceptable set. Namely, in this

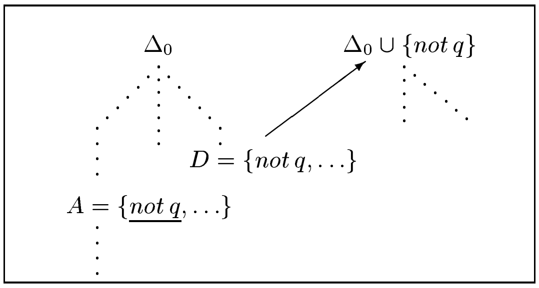

Figure 10: Hypotheses promoted to the root should not be culprits.

example, (hypotheses in the weak part of) defence nodes *should not* be promoted to the root. This (don't know) non-determinism for acceptability adds to the other "don't know" non-determinism in the choice of defence nodes, consisting of the choice of culprits in attack nodes and attacks against those culprits.

The notion of derivation (see definition 40 below) will be parametric with respect to a binary relation *extend-root*. Different semantics employ different instances of the relation *extend-root*, with each such instance reflecting the discussion above on promotion of hypotheses in the weak part of defence nodes to the root. Intuitively, for any semantics, if, for some defence node N, $extend\text{-}root(N, N_{ext})$, then N_{rext} is the subset of $N - hyp\text{-}branch(N)$ to be promoted to the root. The different instances of *extend-root* are given in the following definition.

Definition 39 For each (defence) node N in a partial tree $\mathcal{T}$, the binary relation *extend-root* holds as follows:
well-founded semantics: $extend\text{-}root(N, \emptyset)$;
admissible/weakly stable semantics: $extend\text{-}root(N, N - hyp\text{-}branch(\mathcal{T}, N))$;
acceptable semantics: $extend\text{-}root(N, N')$ iff $N' \subseteq N - hyp\text{-}branch(\mathcal{T}, N)$.

Namely, in the case of the well-founded semantics, no hypothesis is ever added to the root, in the case of admissibility and weakly stability, every hypothesis in (the weak part of) defence nodes is added to the root, and in the case of acceptability any subset of (the weak part of) defence nodes may be added to the root.

The ("don't know") non-determinism, for acceptability, in the choice of additions to the roots of partial trees can be (partially) reduced. Consider again example 12 discussed above. Here, $not\, q$ cannot be added to the root because it is the (only possible) culprit in the attack $\{not\, q\}$ against $not\, p$ in Δ_0. More generally, consider the situation sketched in figure 10. Here, the addition of $not\, q$ in D to the current root Δ_0 of the given partial tree renders any derivable tree non-acceptable, since then $not\, q$ can not be selected in A as the culprit in an acceptable restricted tree because now $not\, q$ belongs to the root and therefore to *hyp-branch* down to, and excluding, A. In general, given a partial tree that we want to expand into an acceptable tree, hypotheses already chosen as culprits should not be promoted to the root.

The stratagem of not adding culprits to the root can also be adopted when building admissible or weakly stable trees. In fact, in this case as well, if a hypothesis belongs to the root then it cannot be possibly selected as a culprit, and any addition to the

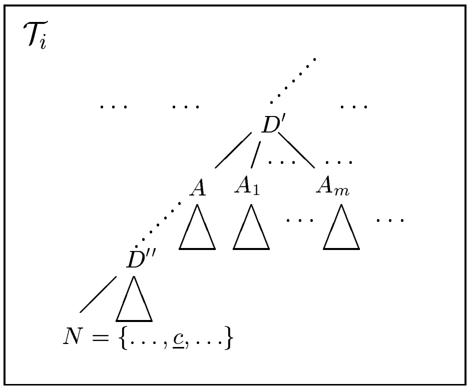

Figure 11: Construction of extended attacks in definition 40, case $\alpha.4$ (acceptability), prompted by the new culprit c chosen in N. $A \cup \{c\}, A_1 \cup \{c\}, \ldots, A_m \cup \{c\}, \ldots$ need to be added as additional children of D'.

root of already selected culprits leads, in general, to non-admissible or non-weakly stable, respectively, trees.

Let us now define the notion of derivation of partial trees formally. In this definition we assume a *selection strategy* identifying the node (in the current partial tree) to be handled next, and we *mark nodes* that should not be selected further. Moreover, we *record culprits* chosen in selected (and marked) attack nodes. This recording implicitly defines a specific culprit function on partial trees, denoted by *culprits* (as for full trees, see definition 19).

For acceptability, we impose an additional condition ($\alpha.4$ below) to handle extended attacks.

Definition 40 Given a function *hyp-branch* and a relation *extend-root*, a *derivation* for a set of hypotheses Δ_0 is a sequence of (partial) trees $\mathcal{T}_1, \ldots, \mathcal{T}_n, \ldots$ such that $\mathcal{T}_1$ consists only of the (unmarked) root Δ_0, labelled as defence, and, given $\mathcal{T}_i$ ($i \geq 1$), if N is the selected (unmarked) node in $\mathcal{T}_i$, then $\mathcal{T}_{i+1}$ is obtained as follows:

(α) if N is an attack node, choose a culprit $c \in N - hyp\text{-}branch(\mathcal{T}_i, N)$ and an attack D against c. Then, $\mathcal{T}_{i+1}$ is $\mathcal{T}_i$ where

1. N is marked,
2. c is recorded as *the* culprit of N,
3. D is added as the (unmarked) defence node child of N,
4. (only for acceptability) for each defence node D', different from the parent D'' of N in the branch of $\mathcal{T}_i$ that leads to N, let A be the child of D' in this branch and $A_1, \ldots, A_m, \ldots$ ($m \geq 0$) be all children of D' in $\mathcal{T}_i$ that are supersets of A. Then the (extended) attack nodes $A \cup \{c\}, A_1 \cup \{c\}, \ldots, A_m \cup \{c\}, \ldots$ are added as additional (unmarked) children of D' in $\mathcal{T}_{i+1}$ (illustrated in figure 11).

(δ) if N is a defence node, $extend\text{-}root(N, N_{ext})$ and $N_{ext} \cap culprits(\mathcal{T}_i) = \emptyset$, then $\mathcal{T}_{i+1}$ is $\mathcal{T}_i$ where

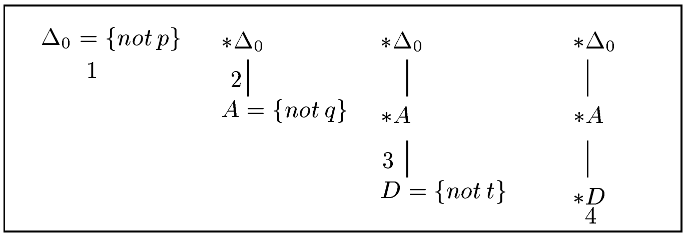

Figure 12: A successful derivation, wrt well-founded semantics, for example 11 ($P = \{p \leftarrow not\, q; q \leftarrow not\, t\}$).

1. N is marked,
2. the root is extended by N_{ext},
3. if $A_1, \ldots, A_m, \ldots$ $(m \geq 0)$ are all minimal attacks against N_{ext} then $A_1, \ldots, A_m, \ldots$ are added as additional (unmarked) attack nodes children of the root, and
4. if $B_1, \ldots, B_k, \ldots$ $(k \geq 0)$ are all minimal attacks against $(N - \textit{hyp-branch}(\mathcal{T}_i, N)) - N_{ext}$, then $B_1, \ldots, B_k, \ldots$ are added as (unmarked) attack nodes children of N.

Note that, if $N - \textit{hyp-branch}(\mathcal{T}_i, N) = \emptyset$ and N is an attack node, then no culprit c can be chosen in N, and no $\mathcal{T}_{i+1}$ can be built. Instead, if $N - \textit{hyp-branch}(\mathcal{T}_i, N) = \emptyset$ and N is a defence node, then necessarily $\textit{extend-root}(N, \emptyset)$ and $\mathcal{T}_{i+1}$ is $\mathcal{T}_i$ with N marked.

Note also that in a derivation, hypotheses already part of the root might be re-added to the root. Moreover, attack nodes might be added as additional children of defence nodes that already admit such attack nodes as children. Definition 40 could be easily modified to avoid such repetitions.

It is easy to see that the following lemma holds:

Lemma 41 Every tree in a derivation is a partial tree.

Definition 42 Given *hyp-branch* and *extend-root*, a *successful derivation* for a set of hypotheses Δ_0 is a (possibly infinite) derivation of partial trees $\mathcal{T}_1, \ldots, \mathcal{T}$ for Δ_0 such that all nodes in $\mathcal{T}$ are marked and all leaves in $\mathcal{T}$ are labelled as defence. If the root of $\mathcal{T}$ is Δ, then we say that the derivation *computes* Δ from Δ_0.

Figure 12 illustrates a (finite) successful derivation computing (in 4 steps) $\Delta = \Delta_0 = \{not\, p\}$ from Δ_0, for example 11, with respect to the choice of the parameters for the well-founded semantics. Nodes are marked with the symbol $*$. The increasing sequence of numbers, shown in the figure, indicates the incremental steps in the construction of the final tree, $\mathcal{T}_4$, for the computed $\Delta = \Delta_0$. Step 4 amounts to marking D, after having checked that there are no attacks against it.

Figure 13 illustrate a (finite) successful derivation computing $\Delta = \{not\, q, not\, r\}$ from $\Delta_0 = \{not\, q\}$, for example 5, with respect to the choice of the parameters for any of admissible, weakly stable or acceptable semantics. $\mathcal{T}_1$ is the left-most tree, $\mathcal{T}_3$ is the central tree and $\mathcal{T}_6$ is the right-most tree in the figure. Step 4 amounts to extending the root (Δ_0) of $\mathcal{T}_3$ with $not\, r$ in D_1. In the case of acceptability, it presupposes $\textit{extend-root}(D_1, D_1)$, i.e. that we have chosen to promote the whole of the (weak part of the) defence node D_1 to the root.

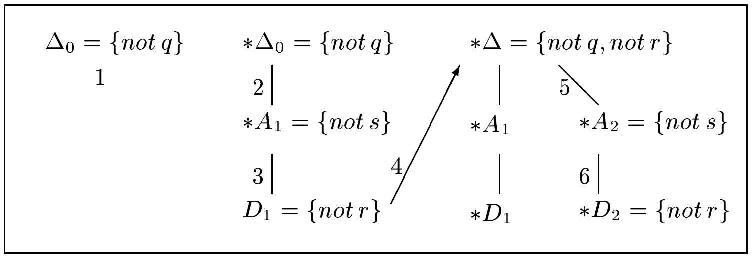

Figure 13: A successful derivation, wrt any of admissibility, weak stability and acceptability, for example 5 ($P = \{q \leftarrow not\, s; s \leftarrow not\, r; r \leftarrow not\, s\}$).

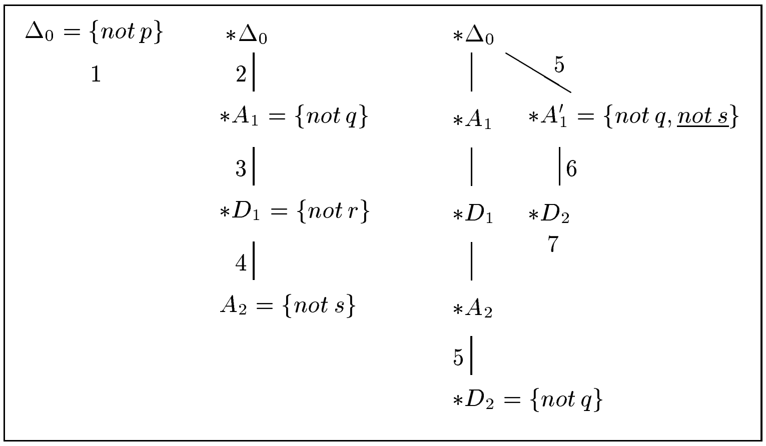

Figure 14: A successful derivation, wrt acceptability, for example 12 ($P = \{p \leftarrow not\, q; q \leftarrow not\, r; r \leftarrow not\, s; s \leftarrow not\, q\}$).

Figure 14 illustrates a (finite) successful derivation computing $\Delta = \Delta_0 = \{not\, p\}$ from Δ_0, for example 12, with respect to the choice of the parameters for acceptability. Step 5 amounts to choosing the defence node D_2 attacking A_2 and building the extended attacks from the culprit $not\, s$ in A_2.

Successful derivations are infinite if and only if they construct trees that are infinite in breadth. Such derivations are needed to guarantee completeness of the proof procedure (see theorem 43 below). For instance, given the program in example 2, the set $\Delta_0 = \{not\, p\}$ is admissible, weakly stable, wf-acceptable and acceptable, since each of the (infinitely many) minimal attacks against Δ_0:

$\{not\, r(0)\}$, $\{not\, r(succ(0))\}$, $\{not\, r(succ(succ(0)))\}, \ldots$

is counter attacked by $\emptyset$. Obviously, any successful derivation computing Δ_0 must be infinite in breadth.

Independently from the selection strategy, we have the following soundness and completeness result.

Theorem 43 Let Δ_0 be a set of hypotheses.

$\Delta_0 = \{not\, s\}$	$\Delta_0 = \{not\, s\}$	$\Delta_0 = \{not s\}$
$A = \{not\, r\}$	$A = \{not\, r\}$	$A = \{not\, r\}$
	Δ_0	Δ_0
		$\vdots$

$\Delta_0 = \{not\, s\}$	$\Delta_0 = \{not\, s\}$
$A = \{not\, r\}$	$A = \{not\, r\}$
	$D = \{not\, t\}$

Figure 15: (a) A (failed) infinite derivation ($P = \{s \leftarrow not\, r; r \leftarrow not\, s\}$) and (b) a successful derivation for Δ_0 ($P = \{s \leftarrow not\, r; r \leftarrow not\, s; r \leftarrow not\, t\}$), wrt wf-acceptability, for example 44.

If there exists a successful derivation computing Δ from Δ_0, with respect to any allowed instance of *hyp-branch* and *extend-root* (with respect to any argumentation semantics), then $\Delta_0 \subseteq \Delta$ and Δ is allowed (with respect to the corresponding argumentation semantics).

Conversely, if there exists an allowed (with respect to any argumentation semantics) set of hypotheses Δ such that $\Delta_0 \subseteq \Delta$, then there exists a successful derivation, with respect to the allowed instance of *hyp-branch* and *extend-root* (with respect to the corresponding argumentation semantics), computing Δ' from Δ_0, such that $\Delta_0 \subseteq \Delta' \subseteq \Delta$ and Δ' is allowed (with respect to the corresponding argumentation semantics).

The proof of this theorem can be found in appendix D

Note that, if we impose that successful derivations be finite, then the above proof theory can not be complete. Indeed, we need infinite derivations to derive (well-founded, admissible, weakly stable and acceptable) restricted trees for example 2. The proof procedures defined in the next section require for derivations to be finite, and are therefore incomplete.

The proof theory (and subsequent proof procedure) for the well-founded semantics can be made more "efficient" by exploiting theorem 24 and corollary 25. Indeed, hypotheses in the culprit part of attack nodes can be chosen as culprits only if they do not occur in the defences in the part of the tree that has already been built. Further improvements are also possible for the well-founded semantics. For instance, note that, by theorem 15, a well-founded pre-tree for a set of hypotheses Δ can be constructed by putting together well-founded pre-trees for each single hypothesis in Δ. In constructing a well-founded tree rooted at a singleton set $\{not\, p\}$, defence nodes below the root should be chosen in such a way that they do not contain $not\, p$. If the only possible defence nodes do contain $not\, p$, then the tree has necessarily an infinite branch, and hence can not be a well-founded restricted tree.

Example 44 Consider the program in example 4 and $\Delta_0 = \{not\, s\}$. Any derivation from Δ_0 adds $A = \{not\, r\}$ as the only attack node child of Δ_0 (see figure 15(a), where we omit marking nodes). Since A can only be counter attacked by Δ_0 itself, Δ_0 is the only possible defence node child of A, and the resulting tree for Δ_0 has an infinite branch (i.e., there exists no successful derivation for Δ_0 with respect to the well-founded semantics). Indeed, Δ_0 is not wf-acceptable. This can be detected by the finite failure to proceed the derivation when the above restrictions on the defences are adopted.

Moreover, if the program is extended by

$$r \leftarrow not\, t$$

then these restrictions will result in avoiding the infinite branch and in choosing $D = \{not\, t\}$ as defence node child of the attack node A, to obtain a successful derivation for (the now wf-acceptable) Δ_0 (see figure 15(b)).

5 Proof procedures

The proof theories developed in the previous section can be specialised to give proof procedures by incorporating a specific (resolution-based) way of computing (supersets of) minimal attacks.

The proof procedures interleave the generation of a pre-solution Δ_0 for a goal G and the extension of Δ_0 to an allowed solution Δ. The proof procedures operate directly on the non-variable-free program P. However, the goal G and the generated sets Δ_0 and Δ are variable-free. [7]

Similarly to the proof theories, the proof procedures are based upon the generation of restricted trees via derivations of intermediate, possibly non-restricted trees. Whereas nodes in the intermediate, partial trees for the proof theories are sets of (variable-free) hypotheses, nodes in the intermediate trees for the proof procedures are goals, possibly non-variable-free and containing atoms. These trees will be referred to as "goal-trees".

Definition 45 Let *goal-branch* be any function such that given a tree with goals as nodes and a node of the tree, returns a set of hypotheses. Then, a *goal-tree* is a labelled tree $\mathcal{T}$ where

(GT1) each node is a goal, labelled as *attack-goal* or *defence-goal*, but not as both;

(GT2) the root is labelled as defence-goal;

(GT3) every node labelled as attack-goal has at most one child, labelled as defence-goal;

(GT4) every node labelled as defence-goal has any number of children, labelled as attack-goal;

(GT3-4) for each node G child of a node G' there exists a hypothesis $not\, p \in G' - goal\text{-}branch(\mathcal{T}, G')$ such that $P \cup G \models p$; [8]

(GT5) every branch of $\mathcal{T}$ is finite.

While attack and defence nodes in restricted and partial trees attack their parent nodes, attack- and defence-goal nodes in goal-trees only *potentially* attack (hypotheses in) their parent nodes. The *actual* attacks result from reducing (by resolution) all atoms in the goal nodes to hypotheses. If some of the atoms in a goal node can not be reduced, then the goal node does not represent any attack against its parent node.

[7] The restriction that G be variable-free is not essential, since the proof procedures could be easily modified to return, as answers, substitutions for variables, in addition to sets of hypotheses.

[8] Here, P and G stand for the set of all their variable-free instances, and negative literals in G are treated like atoms.

Moreover, the reduction might introduce some non-minimal attacks. For example, given the program

$$p \leftarrow not\, q$$
$$p \leftarrow not\, q, not\, r$$

and the hypothesis $not\, p$, both the minimal attack $\{not\, q\}$ and the non-minimal attack $\{not\, q, not\, r\}$ are computed by reducing the atom p.

In the previous section we saw how the proof theories for the various semantics are all obtained from the general definition of derivation by appropriately instantiating the parameters *hyp-branch* and *extend-root*, and by using a function *culprits* that is implicitly (via recording of culprits) constructed during the derivations. Similarly, in this section we will see how proof procedures for the various semantic are all obtained from the general definition of "goal-derivation" below by intantiating the parameters *goal-branch* and *extend-goal-root*, and by using and constructing an appropriate function *goal-culprits*.

Definition 46 Given a goal-tree $\mathcal{T}$ and a node N in $\mathcal{T}$, such that the branch from the root to N in $\mathcal{T}$ is $N_0, N_1, \ldots, N_i = N$, for $i \geq 0$, the function *goal-branch* is defined as follows:
well-founded semantics: $goal\text{-}branch(\mathcal{T}, N)=\emptyset$;
admissibility: $goal\text{-}branch(\mathcal{T}, N)=\{h \mid h \in N_j \cap \mathcal{H},\ j = 0, \ldots, i-1$ and N_j is a defence-goal in $\mathcal{T}\}$ [9];
weak stability and acceptability: $goal\text{-}branch(\mathcal{T}, N)=\{h | h \in N_j \cap \mathcal{H}, j = 0, .., i-1\}$.

Note that, as for the proof theory, if N is the root then $goal\text{-}branch(\mathcal{T}, N)=\emptyset$, for all semantics.

Definition 47 For every goal node N in a given goal-tree $\mathcal{T}$,
$(N \cap \mathcal{H}) - goal\text{-}branch(\mathcal{T}, N)$ is the *weak part* of N.
If N is an attack-goal node, then any hypothesis in the weak part of N attacked by the (single) child of N in $\mathcal{T}$ is a *culprit* of N. A *culprit function* is any function, denoted by *goal-culprits*, from the set of goal-trees to the power set of $\mathcal{H}$, such that for any given goal-tree $\mathcal{T}$ and any attack-goal node G_A which is not a leaf of $\mathcal{T}$, $goal\text{-}culprits(\mathcal{T})$ contains exactly one culprit of G_A.

Note that *goal-culprits* returns no culprits for attack-goal nodes without children.

Definition 48 Given $h \in \mathcal{H}$, the relation *extend-goal-root* is given as follows:
well-founded semantics: $extend\text{-}goal\text{-}root(h, no)$;
admissibility and weak stability: $extend\text{-}goal\text{-}root(h, yes)$;
acceptability: $extend\text{-}goal\text{-}root(h, yes)$, $extend\text{-}goal\text{-}root(h, no)$.

Hence the relation *extend-goal-root* serves the purpose of distinguishing the different attitudes, for the different semantics, to extending the root. Hypotheses (in defence-goals) are selected one by one in "goal-derivations", and are never added to the root for the well-founded semantics, are always added to the root for admissibility and weak stability, and might or might not be added to the root for acceptability.

[9] Alternatively, $goal\text{-}branch(\mathcal{T}, N)=\{h \mid h \in N_j \cap \mathcal{H},\ j = 2k$ and $k = 0, \ldots, \lfloor \frac{i-1}{2} \rfloor\}$.

In the proof theories, defence and attack nodes are selected and marked at once. In the proof procedures, attack- and defence-goal nodes might be selected many times before they are actually marked, due to the fact that they might contain positive atoms. In effect, single steps in the proof theory are expanded to several steps in the proof procedure, needed to compute the attacking child. In order to obtain goal-trees whose branches consist of alternating attack- and defence-goal nodes, in "goal-derivations" nodes can not only be added to goal-trees, but can also be replaced by new nodes, after resolution takes place.

In the definition of "goal-derivation" below, we will denote the *selection rule* of literals in goal-nodes by $\Re$. We will assume that $\Re$ is safe, i.e. negative literals are selected by $\Re$ only if they are variable-free. Moreover, we will suppose that each goal-node G in a goal-tree is divided into two parts, with the first part, referred to as $G^{already}$, being a set of hypotheses which is transparent to the selection rule $\Re$. Hence, $\Re(G)$ will stand for $\Re(G - G^{already})$. Furthermore, $\Re(G) = \perp$ iff $(G - G^{already}) = \emptyset$.

Definition 49 Given *goal-branch* and *extend-goal-root*, a *goal-derivation* for a (variable-free) goal G is a sequence of goal-trees $\mathcal{T}_1, \ldots, \mathcal{T}_n$ such that $\mathcal{T}_1$ consists only of the (unmarked) root G, labelled as defence-goal, and given $\mathcal{T}_i$ $(i = 1, \ldots, n-1)$, if N is the selected (unmarked) node in $\mathcal{T}_i$, then $\mathcal{T}_{i+1}$ is obtained as follows:

(α) *If N is an attack-goal and $\Re(N) \neq \perp$, then*

(α_{atom}) *If $\Re(N) = q$, where q is an atom, and $G_1, \ldots, G_m$ $(m \geq 0)$ are all the resolvents of N on q (in P), then $\mathcal{T}_{i+1}$ is $\mathcal{T}_i$ where N is replaced by $G_1, \ldots, G_m$ as attack-goal nodes children of the parent of N, and $G_1^{already} = \ldots = G_m^{already} = N^{already}$.*

Note that if $m = 0$ then N is removed from $\mathcal{T}_i$ and does not appear in $\mathcal{T}_{i+1}$. This amounts to the removal of a potential attack (N) corresponding to no actual attack.

(α_{naf}) *If $\Re(N) = not\, q$, where q is an atom, then*

- *if $not\, q \in$goal-branch$(\mathcal{T}, N)$, then $\mathcal{T}_{i+1}$ is $\mathcal{T}_i$ with $N^{already}$ replaced by $N^{already} \cup \{not\, q\}$,*
- *if $not\, q \notin$goal-branch$(\mathcal{T}, N)$, then $\mathcal{T}_{i+1}$ is $\mathcal{T}_i$ where*
  1. *N is marked,*
  2. *$not\, q$ is recorded as the culprit of N,*
  3. *$\{q\}$ is added as the (unmarked) defence-goal node child of N, and*
  4. *(only for acceptability) for each defence-goal node D different from the parent of N in the branch that leads to N in $\mathcal{T}_i$ such that A is the child of D in the branch containing N and $A_1, \ldots, A_n$ are all the children of D that are supersets of A in $\mathcal{T}_i$,*
     the (extended) attack-goal nodes $A' = A \cup \{not\, q\}, A'_1 = A_1 \cup \{not\, q\}$, $\ldots$, $A'_m = A_m \cup \{not\, q\}$ are added as additional (unmarked) children of D in $\mathcal{T}_{i+1}$. $A'^{already} = A^{already}$, $A'^{already}_1 = A^{already}_1, \ldots, A'^{already}_m = A^{already}_m$.

Note that the first hypothesis selected in an attack-goal node is chosen as the culprit, and the attack-goal node is marked as soon as such culprit is selected in it. Therefore, marked attack-goal nodes might not be sets of hypotheses.

Note also that N is an attack-goal and $\Re(N) = \perp$, then there exists no $\mathcal{T}_{i+1}$ and N can not be marked. (We will see later, with definition 51, that this corresponds to a failure.) Alternatively, we could have a case $\alpha_\perp$, for $\Re(N) = \perp$, simply marking N. (This would still lead to a failure, with respect to definition 51.)

(δ) *If N is a defence-goal node, then*

($\delta_\perp$) *If $\Re(N) = \perp$, then $\mathcal{T}_{i+1}$ is $\mathcal{T}_i$ with N marked;*

Note that a defence-goal node is marked only when all the hypotheses in it have been generated (by unfolding atoms) and moved to the *already* part. Therefore, marked defence-goal nodes are sets of hypotheses.

(δ_{atom}) *If $\Re(N) = q$, where q is an atom, then choose a resolvent G' of N on q (in P), and $\mathcal{T}_{i+1}$ is $\mathcal{T}_i$ where N is replaced by the defence-goal node G' and $G'^{already} = N^{already}$;*

(δ_{naf}) *If $\Re(N) = not\,q$, where q is an atom, then*

- *if $not\,q \in$goal-branch($\mathcal{T}_i, N$), then $\mathcal{T}_{i+1}$ is $\mathcal{T}_i$ where $N^{already}$ is replaced by $N^{already} \cup \{not\,q\}$;*
- *if $not\,q \notin$goal-branch($\mathcal{T}_i, N$), then*
  - *if extend-goal-root($not\,q, yes$) and $not\,q \notin$culprits($\mathcal{T}_i$), then $\mathcal{T}_{i+1}$ is $\mathcal{T}_i$ where*
    1. *$N^{already}$ is replaced by $N^{already} \cup \{not\,q\}$,*
    2. *the root of $\mathcal{T}_{i+1}$ is $R' = R \cup \{not\,q\}$, where R is the root of $\mathcal{T}_i$, $R'^{already} = R^{already} \cup \{not\,q\}$, and*
    3. *$\{q\}$ is added as an additional attack-goal node child of R';*
  - *if extend-goal-root($not\,q, no$) and (only in the case of acceptability) $not\,q \notin$culprits($\mathcal{T}_{iN}$), with $\mathcal{T}_{iN}$ the subtree of $\mathcal{T}_i$ rooted at N, then $\mathcal{T}_{i+1}$ is $\mathcal{T}_i$ where*
    1. *$N^{already}$ is replaced by $N^{already} \cup \{not\,q\}$ and*
    2. *$\{q\}$ is added as an additional attack-goal node child of N.*

The first bullet, for *extend-goal-root*($not\,q, yes$), applies in the case of admissibility/weak stability as well as in the case of acceptability. The second bullet, for *extend-goal-root*($not\,q, no$), applies in the case of the well-founded semantics as well as in the case of acceptability. The test in the *extend-goal-root*($not\,q, no$) case, for acceptability, that the selected hypothesis does not occur as a culprit in the subtree below the defence-goal, is needed because defences are incrementally built, so that the new appearance of $not\,q$ in a defence-goal node would invalidate a previously chosen culprit $not\,q$ in the branch below the defence-goal node. This test was not present in the proof theory, but is analogous to the test needed when extending the root.

Note that cases δ_{atom} and δ_{naf}, first item, amount to performing SLD-resolution with respect to the given program extended with the hypotheses in the branch as additional facts in the program.

It is easy to see that the following lemma holds

Lemma 50 Every tree in a goal-derivation is a goal-tree.

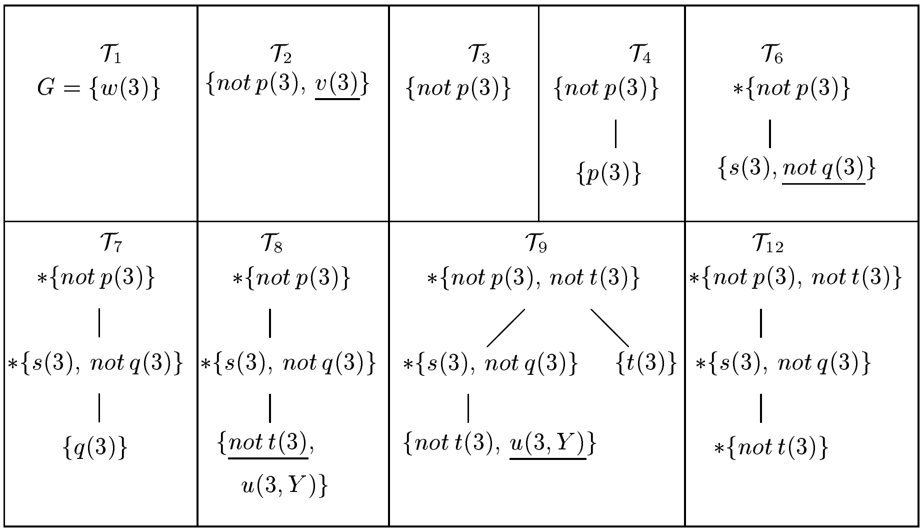

Figure 16: A successful goal-derivation for example 52 ($P = \{w(X) \leftarrow not\,p(X), v(X); p(X) \leftarrow s(X), not\,q(X); q(X) \leftarrow not\,t(X), u(X,Y); v(3); s(3); u(3,2)\}$).

Definition 51 Given *goal-branch* and *extend-goal-root*, a *successful goal-derivation* for a (variable-free) goal G is a finite goal-derivation $\mathcal{T}_1, \ldots, \mathcal{T}_n$ for G, with all nodes in $\mathcal{T}_n$ marked and all leaves in $\mathcal{T}_n$ defence-goals. If Δ is the root of $\mathcal{T}_n$, then we say that the goal-derivation *computes* Δ from G.

Note that, since every marked defence-goal node in a goal-derivation is necessarily a set of hypotheses, the Δ computed by a successful goal-derivation is a set of hypotheses.

Example 52 Consider the program

$$\begin{aligned} w(X) &\leftarrow not\,p(X),\ v(X) \\ p(X) &\leftarrow s(X),\ not\,q(X) \\ q(X) &\leftarrow not\,t(X),\ u(X,Y) \\ v(3)& \\ s(3)& \\ u(3,2)&. \end{aligned}$$

Figure 16 shows a successful derivation for $G = \{w(3)\}$, computing $\Delta = \{not\,p(3), not\,t(3)\}$, with respect to a choice of the parameters valid for admissibility, weak stability and acceptability. $\mathcal{T}_2$ and $\mathcal{T}_3$ are obtained via a δ_{atom} step (we underline the literal selected by $\Re$). $\mathcal{T}_4$ is obtained via a δ_{naf} step. $\mathcal{T}_5$ is $\mathcal{T}_4$ with the root marked, obtained via a $\delta_{\perp}$ step. $\mathcal{T}_6$ is obtained via an α_{atom} step, resolving $p(3)$ with the second clause of the program. $\mathcal{T}_7$ is obtained via an α_{naf} step. $\mathcal{T}_8$ is obtained via a δ_{atom} step. $\mathcal{T}_9$ is obtained via a δ_{naf} step, promoting $not\,t(3)$ to the root. $\mathcal{T}_{10}$ is $\mathcal{T}_9$ with the right-hand side attack-goal node $\{t(3)\}$ removed, obtained by an α_{atom} step (with no resolvents for $t(3)$). $\mathcal{T}_{11}$ is the same as $\mathcal{T}_{12}$ with the defence-goal node of level 2 not marked, and is obtained from $\mathcal{T}_{10}$ via a δ_{atom} step, on the selected $u(3,Y)$. $\mathcal{T}_{12}$ is obtained from $\mathcal{T}_{11}$ via a $\delta_{\perp}$ step.

Theorem 53 Given a (variable-free) goal G, if there exists a successful goal-derivation computing Δ from G, with respect to allowed choices of the parameters *goal-branch* and *extend-goal-root*, then Δ is a solution for G with respect to the semantics corresponding to the choice of the parameters (*soundness*).

The proof of this theorem can be found in appendix E.

The proof procedure is incomplete. One source of incompleteness is the usual possible occurrence of floundering. Another source, that applies even to the case of propositional programs, stems from the fact that successful derivations must be finite.

Example 54 Consider the program

$$p \leftarrow p$$

and the set $\{not\, p\}$. This is admissible, weakly stable, acceptable and well-founded. However, the proof procedure fails to detect it, since resolution in the α_{atom} step loops.

Moreover, consider the program in example 2 and the set of hypotheses $\{not\, p\}$. This is admissible, weakly stable, acceptable and well-founded. However, the proof procedure fails to detect it, since it cannot detect that each of the infinitely many attacks against $\{not\, p\}$ is counterattacked by the empty set.

In order to address the incompleteness of the proof procedure, we would need to integrate it with constructive negation and loop checking techniques. In the case of propositional programs, the following loop-checking technique can be easily accommodated within the proof procedure: if an attack requires the proof of an atom p, then $\{not\, p\}$ is added as a defence against the attack. Hence in an α_{atom} step with the selected atom being p we can add $\{not\, p\}$ as a (special) defence-goal in the tree. Then, if in the defence of this hypothesis we require again that p must fail in an attack-goal of a subsequent α_{atom} step, we can use the previous defence $\{not\, p\}$ and thus cut the loop. This technique together with the dual technique for the δ_{atom} step has been used in [23] to extend the related abductive proof procedure in [14] (see below in section 6). The resulting extended proof procedure can be shown to be complete with respect to the credulous semantics in [23]. A similar loop checking technique has been applied in [29] for the well-founded model semantics. Section 7 briefly discusses some other existing proof procedures adopting loop checking techniques.

6 Implementations of the proof procedures

Any implementation of the proof procedure, for a given semantics, must adopt a concrete strategy for the selection strategy for (unmarked) nodes in goal-trees and for the selection rule for literals in the selected node. Moreover, any such implementation requires a representation of goal-trees in goal-derivations. The two choices are related, since some selection strategies allow more convenient representations of goal-trees.

Indeed, some possible selection strategies require a full representation of goal-trees, e.g. the selection strategy operating breadth-first and the selection strategy allowing to arbitrarily select any node at any depth at any stage in the construction of goal-trees. Instead, for other selection strategies it may be sufficient to hold only a part of goal-trees throughout the computation, e.g. the selection strategy operating depth-first. This requires that, once a node N has been selected, the treatment of all other nodes at the same level is suspended until a complete subtree rooted at N has been constructed, and all its nodes have been marked.

As usual, this (depth-first) selection strategy renders unnecessary the representation of the full goal-trees in the goal-derivations, and allows instead to record only (the respective instance of) *goal-branch*(N), with N the currently selected node, at each time. Of course, for this to be possible, the culprits encountered in the derivations must be collected in a separate, subsidiary structure replacing the function *culprits*.

If we apply the depth-first selection strategy to the instance of the procedure to compute admissibility, we obtain a procedure which simulates the Eshghi-Kowalski proof procedure [14], in the sense that every derivation in the Eshghi-Kowalski procedure can be mapped onto a derivation, computing the same solutions to the same goals, in our procedure under the depth-first selection strategy.

Theorem 55 Given a logic program P and a (variable-free) goal G, if there exists a successful abductive derivation from $(G, \emptyset)$ to $(\perp, \Delta)$ in the proof procedure of [14] then there exists a successful goal-derivation from G computing Δ under the depth-first strategy and with respect to the instance of the parameters for the admissibility semantics.

A sketch of the proof of this theorem can be found in appendix F.

Our procedure, however, differs from the Eshghi-Kowalski procedure in the detection of failures, which is done more effectively because of the use of culprits.

If we apply the depth-first selection strategy to the instance of the procedure to compute the well-founded semantics, we obtain a procedure which simulates SLDNF, in the sense that, for a given goal G, if SLDNF succeeds then there exists a successful goal-derivation from G. The possible improvements to the proof theory we discussed at the end of section 4 can be imported into the proof procedure for the well-founded semantics and can be seen as possible improvements to SLDNF.

In the case of acceptability, the depth-first selection strategy allows to avoid the (expensive) construction of the extended attack nodes "culprit by culprit", climbing up the branch with the attack containing the selected culprit (see α_{naf}). Indeed, the depth-first selection strategy forces the full construction of the subtree below an attack-goal node G, and the marking of all nodes in such subtree, before any other attack in the tree is considered. Since attacks extending G are built from the culprits in the subtree rooted at G, the construction of all attacks extending G can be performed after the subtree has been fully constructed.

7 Related work

The argumentation-theoretic formulation of the semantics of LP adopted in this paper strongly relies upon the abductive interpretation of LP [14]. The problem of determining whether a given goal is (credulously) entailed by the program is expressed as the problem of determining an allowed set of hypotheses (negative literals). Argumentation provides a way of formulating the allowedness criteria. In conventional abduction, such criteria are expressed in terms of satisfaction of integrity constraints. However, the overall models of computation are very similar in the two cases. The computation of abduction in logic programming has been widely studied (see [21, 22] for a survey of the various proposals) and has motivated the work presented in this paper. Indeed, in the previous section 6 we have seen that the Eshghi-Kowalski abductive procedure [14] can be obtained as a special instance of the argumentative computational model we propose, although it is formulated in seemingly different, non-argumentative terms.

Other existing procedures for NAF can also be obtained as instances of our proof procedure(s). In the previous section 6 we have seen that SLDNF can be obtained as a special instance of the proof procedure for the well-founded semantics. SLS [36] can also be obtained from the same instance of our proof theory. The infinite derivations allowed in SLS correspond to the infinite derivations in our proof theory, necessary to guarantee completeness. Our proof procedures do not allow for infinite derivations, but as discussed earlier at the end of section 5 these could be integrated by means of some loop checking techniques.

A particularly effective computation of the well-founded model semantics, employing various forms of loop checking that may be usefully integrated into our procedures, is SLG resolution studied in [7, 8, 41, 38]. Here, the technique of tabling is employed to handle positive loops (like the one in example 54) and the technique of delaying is used to handle negative loops (like the ones in examples 4, 7 and 12). These techniques also allow to avoid repetitions in the resolution of goals. SLG is sound and search complete with respect to the well-founded semantics and is guaranteed to terminate for programs with the bounded-term-size property. Moreover, SLG has polynomial data complexity for programs without function symbols. The loop-checking techniques of SLG have also been adapted for the computation of stable (and preferred extensions or partial stable models) in [6].

Niemelä and Simmons [32] define proof procedures for computing well-founded and stable model semantics, whose emphasis is also on efficiency. The proof procedure for stable models can be used to compute admissibility, weak stability and acceptability whenever all these semantics coincide (see the end of section 2). The proof procedures in [32] perform a bottom-up computation. Pereira, Aparìcio and Alferes [33] define top-down derivation procedures for well-founded and partial stable model semantics, also incorporating loop checking techniques. These procedures apply to extended logic programs as well, i.e. logic programs including explicit (classical) negation [18].

Although the interest in argumentation has grown in recent years, there has been relatively little attention paid to the problem of computing argumentation, with some notable exceptions.

Prakken and Sartor [35] develop a proof theory for extended LP [18] with a priority ordering amongst rules, computing a (skeptical) argumentation semantics which extends the well-founded semantics. Their argumentative computational model closely match the semantics and is based on the notion of play dialogue, expressed as an (appropriate) sequence of interleaved arguments and counter arguments. Each such dialogue in favour of a conclusion is seen as a branch of a tree. Our approach is thus analogous to theirs: both approaches make use of a tree structure and, in order to prove that some given goal (set of arguments) is entailed within a chosen semantics of argumentation, construct trees where each branch corresponds to the investigation of an attack (or argument against) and the construction of a proper defence for (or counter argument against the argument against) the initial goal.

Dimopoulos and Kakas [10] provide a goal-oriented top-down proof procedure for computing the argumentation-theoretic semantics of LP without NAF, a framework for extended logic programs without NAF but with priorities amongst rules. This can be seen as a particular realisation of the more abstract procedures presented in this paper for the particular representation framework at hand.

Dung, Kowalski and Toni [13] develop an abstract proof procedure for the admissibility semantics, in the form of a logic program derived systematically from its specification. This proof procedure is equivalent to the instance of our proof theory for admissibility.

Thielscher [44] defines a proof procedure for computing an argumentation-based semantics for LP which is an extension of the well-founded semantics. This semantics is expressed in terms closely resembling those used for the argumentation-theoretic formulation of the LP semantics in this paper. It can be proved [45] that this semantics coincides with the sceptical version of the preferred extension semantics, namely: given a logic program P and goal G, G is entailed by the semantics in [44] for P iff G holds in every preferred extension of P. Such a semantics for LP is an instance of a more abstract semantics for non-monotonic reasoning, in the same way the semantics for LP studied in this paper are. However, the proof procedure in [44] is expressed in an imperative style language and therefore a direct comparison with ours is difficult.

8 Conclusions and future work

We have shown how different argumentation semantics for LP with NAF can be computed uniformly within the same computational model. The argumentation semantics we have considered cover many of the existing semantics for NAF in LP, i.e. partial stable model (and stable model semantics for the large class of programs where stable and partial stable model semantics coincide), stable theory, acceptability and well-founded model semantics, and other semantics equivalent to these. In fact, since the introduction of NAF in LP, there have been several proposals for its computation, seemingly very different in nature. In this paper we have proposed a single computational framework, parametrically defined, from which a family of proof theories and top-down procedures for query answering with respect to different semantics are obtained uniformly, by appropriately instantiating the parameters of this framework.

In defining the basic computational framework and the proof theories, we have abstracted away from the specific features of LP and built uniquely upon the argumentation-theoretic formulation of (the semantics of) LP, assuming that the framework satisfies some general properties, e.g. that the background logic is monotonic and that no set of hypotheses can attack the empty set of hypotheses. Indeed, our proposed computational framework is not limited to LP alone but can be directly adapted to be applied to other argumentation framework satisfying the same properties. As a result, the given proof theories can be used as a basis to define top-down, query-answering procedures for acceptability, weak stability, admissibility and well-founded semantics for any non-monotonic formalism that can be formulated in argumentation terms, whenever the given formalism satisfies the same properties. Recent work [4, 5, 12, 20] shows that most of the existing frameworks for non-monotonic reasoning fall in this category. As a consequence, the paper provides a uniform computational basis for non-monotonic reasoning in general.

We have defined proof theories and procedures in terms of derivations of appropriate trees whose nodes are sets of negative literals. The proof theories are sound and complete and the proof procedures are sound with respect to the semantics. The proof procedures can be used for query answering with respect to various semantics of LP with NAF, i.e. admissibility, weak stability, acceptability and well-founded semantics, as well as all semantics equivalent to these. Existing proof procedures can be seen as particular cases of our proof procedures by choosing different specific selection strategies for nodes of the trees.

Whereas the proof theories are defined for variable-free programs, the proof procedures are defined for logic programs in their first-order form. However, these proof procedures can not handle NAF literals with variables, and assume a safe selection rule for negative literals in goals. Recently, [9, 16, 28] have proposed proof procedures for abduction that can compute non-ground explanations where the (existential) variables of the non-ground hypotheses in such explanations are appropriately constrained. Further study is needed to examine whether techniques from these proposals might be used for extending our proof procedures to handle variables in negative literals and more generally non-ground, existentially quantified arguments.

In this paper, we have focused on understanding the main problems underlying the computation of the argumentation semantics with little attention on the complexity of the computation, pointing only to some possible directions for an effective implementation of the proof procedures. A more thorough study of such implementations is an issue for future research. Recently, Niemelä and Simmons [32] and others [19, 15, 43] have defined proof procedures for computing the well-founded and stable model semantics of range-restricted, function-free logic programs, with a strong emphasis on the computational efficiency of these procedures. These methods are mostly based on performing a bottom-up computation for programs in their propositional form. Despite these differences in approach, it would be interesting to investigate to what extent methods adopted in these works could be integrated into our proof procedures for argumentation in order to improve their efficiency. Similarly, tabling and other techniques introduced in [6, 7, 8, 41, 38] again for the efficient computation of the well-founded (and stable model) semantics might be useful in developing effective implementations of our proof procedures.

Other argumentation semantics for LP have been proposed, e.g. [1]. The use of the computational model given in our paper for developing procedures for this semantics is another issue for future research.

Acknowledgements

The authors are grateful to R.A. Kowalski for many helpful suggestions and comments on earlier drafts, and to P.M. Dung and P. Mancarella for many helpful discussions. This work was partially supported by Fujitsu Limited, Japan, and by the University of Cyprus.

References

[1] J. J. Alferes, L. M. Pereira. An argumentation-theoretic semantics based on non-refutable falsity. *Proceedings of the International Workshop on Non-monotonic Extensions of Logic Programming* (J. Dix, L. M. Pereira and T. Przymusinski, eds)

[2] K.R. Apt, H. Blair, A. Walker. Towards a theory of declarative knowledge. *Foundations of deductive databases and logic programming,* J. Minker, editor, Morgan Kaufmann, Los Altos, CA (1988)

[3] C. Aravindan, P.M. Dung. On the correctness of unfold/fold transformation of normal and extended logic programs. *Journal of Logic Programming* 24(3):201–217 (1995)

[4] A. Bondarenko, F. Toni, R. A. Kowalski. An assumption-based framework for non-monotonic reasoning. *Proceedings of the 2nd International Workshop on Logic Programming and Non-monotonic Reasoning,* Lisbon, Portugal (1993) MIT Press (L. M. Pereira and A. Nerode, eds) 171–189

[5] A. Bondarenko, P. M. Dung, R. A. Kowalski, F. Toni. An abstract, argumentation-theoretic framework for default reasoning. *Artificial Intelligence* 93(1-2) 63–101 (1997)

[6] W. Chen. Extending Prolog with non-monotonic reasoning. *Journal of Logic Programming* 27(2) 169–183 (1996)

[7] W. Chen, T. Swift, D.S. Warren. Efficient top-down computation of queries under the well-founded semantics. *Journal of Logic Programming* 24(3) 161–199 (1995)

[8] W. Chen, D.S. Warren. Tabled evaluation with delaying for general logic programs. *Journal of the ACM* 43(1) 20–74 (1996)

[9] M. Denecker, D. De Schreye. SLDNFA: An abductive proof procedure for abductive logic programming. *Journal of Logic Programming* (1997)

[10] Y. Dimopoulos, A.C. Kakas. Logic programming without negation as failure. *Proceedings of the International Logic Programming Symposium*, Portland, Oregon (1995) MIT Press (()J. Lloyd, ed.) 369–384

[11] P. M. Dung. Negation as hypothesis: an abductive foundation for logic programming. *Proceedings of the 8th International Conference on Logic Programming*, Paris, France (1991) MIT Press (K. Furukawa, ed.) 3–17

[12] P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning and logic programming. *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, Chambery, France (1993) Morgan Kaufmann (R. Bajcsy, ed.) 852–857

[13] P. M. Dung, R. A. Kowalski, F. Toni. Argumentation-theoretic proof procedures for non-monotonic reasoning. *Proceedings of the 6th Workshop on Logic Programming Synthesis and Transformation*, Stockolm, Sweden (1996) (J. Gallagher ed.)

[14] K. Eshghi, R. A. Kowalski. Abduction compared with negation as failure. *Proceedings of the 6th International Conference on Logic Programming*, Lisbon, Portugal (1989) MIT Press (G. Levi and M. Martelli, eds) 234-254

[15] J.A. Fernandez, J. Lobo, J. Minker and V. S. Subrahmaniann. Disjunctive Lp + Integrity Constraints = Stable Model Semantics. *Ann. of Mathematics and Artificial Intelligence* 8(3-4) 449-474 (1993)

[16] Tze Ho Fung, R. A. Kowalski. The IFF proof procedure for abductive logic programming. *Journal of Logic Programming* (1997)

[17] M. Gelfond, V. Lifschitz. The stable model semantics for logic programming. *Proceedings of the 5th International Conference on Logic Programming*, Washington, Seattle (1988) MIT Press (K. Bowen and R. A. Kowalski, eds) 1070-1080

[18] M. Gelfond, V. Lifschitz. Logic programs with classical negation. *Proceedings of the 7th International Conference on Logic Programming*, Jerusalem (1990) MIT Press (()D.H.D. Warren and P. Szeredi, eds.) 579-597

[19] K. Inoue, M. Koshimura and R. Hasegawa. Embedding Negation as Failure into a Model Generation Theorem Prover. *Proc. CADE-11*,Springer Verlag LNAI 607, 400–415 (1992)

[20] A. C. Kakas. Default reasoning via negation as failure. *Foundations of Knowledge Representation and Reasoning*, Berlin, Germany (1994) LNAI 810 Springer Verlag (G. Lakemeyer and B. Nebel, eds) 160-178

[21] A. C. Kakas, R. A. Kowalski, F. Toni. Abductive logic programming. *Journal of Logic and Computation* 2(6) (1992) Oxford University Press, 719-770

[22] A. C. Kakas, R. A. Kowalski, F. Toni. The role of abduction in logic programming. *Handbook of Logic in Artificial Intelligence and Logic Programming*, Volume 5, 235-324, Oxford University Press (1998)

[23] A. C. Kakas, P. Mancarella. An extended abductive framework for negation. Technical Report, Logic Programming Section, Department of Computing, Imperial College, London (1990)

[24] A. C. Kakas, P. Mancarella. Stable theories for logic programs. *Proceedings of the International Logic Programming Symposium*, San Diego, CA (1991) MIT Press (V. Saraswat and K. Ueda, eds) 85-102

[25] A. C. Kakas, P. Mancarella. Preferred extensions are partial stable models *Journal of Logic Programming* 14 (3,4) (1993), Elsevier, 341–348

[26] A. C. Kakas, P. Mancarella, P.M. Dung. The Acceptability Semantics for Logic Programs. *Proceedings of the 11th International Conference on Logic Programming*, Santa Margherita Ligure, Italy (1994) MIT Press (P. van Hentenryck, ed.) 504–519

[27] A. C. Kakas, P. Mancarella, P. M. Dung. Negation as failure revisited. Pisa University Technical Report (1992)

[28] A.C. Kakas, A. Michael. Integrating abductive and constraint logic programming. *Proceedings of the 12th International Conference on Logic Programming*, Japan (1995) MIT Press (Sterling, ed.)

[29] V. Lifschitz, N. McCain, T.C. Przymusinski, R.F. Stärk. Loop Checking and the well-founded semantics. *Proceedings of the 3rd International Workshop on Logic Programming and Non-monotonic Reasoning*, Lexington, KY (1995) LNAI 928 Springer Verlag (V. Marek, A. Nerode and M. Truszczynski, eds) 127-142

[30] D. McDermott. Nonmonotonic logic II: non-monotonic modal theories. *Journal of ACM* 29(1) (1982) 33–57

[31] R. Moore. Semantical considerations on non-monotonic logic. *Artificial Intelligence* 25 (1985), Elsevier, 75–94

[32] I. Niemelä, P. Simons. Efficient implementation of the well-founded and stable model semantics. *Proc. JICSLP'96*, MIT Press, 289–303

[33] L.M. Pereira, J.N. Aparìcio, J.J. Alferes. Derivation procedures for extended stable models. *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, Sidney, Australia (1991) Morgan Kaufmann (J. Mylopoulos, R. Reiter), 863–868

[34] D. Poole. A logical framework for default reasoning. *Artificial Intelligence* 36 (1988), Elsevier, 27–47

[35] H. Prakken, G. Sartor. A dialectical model of assessing conflicting arguments in legal reasoning. *Artificial Intelligence and Law Journal* 4(3–4) (1996), Special Issue on Logical Models of Argumentation.

[36] T. Przymusinsky. Well-founded and Stationary Models of Logic Programs, *Annals of Mathematics and Artificial Intelligence* 12(1994), 141-187.

[37] T. Przymusinsky. Semantics of disjunctive logic programs and deductive databases. *Proceedings of the 2nd International Conference on Deductive and Object-Oriented Databases*, Munich, Germany (1991) LNCS 566 Springer Verlag (C. Delobel, M. Kifer and Y. Masunaga eds.) 85–107

[38] P. Rao, K. Sagonas, T. Swift, D.S. Warren and J. Freire. XSB: A System for Efficiently Computing the Well-Founded Semantics. *Proceedings of the 4th International Workshop on Logic Programming and Non-monotonic Reasoning* (1997) 430–440

[39] R. Reiter. A logic for default reasoning. *Artificial Intelligence* 13 (1980), Elsevier, 81–132

[40] D. Saccà, C. Zaniolo. Stable model semantics and non-determinism for logic programs with negation. *Proceedings of the 9th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, Nashville, Tennessee (1990) ACM Press, 205–217

[41] K. Sagonas, T. Swift and D.S. Warren. An abstract machine for computing the well-founded semantics. *JICSLP'96*, Bonn, Germany (1996), MIT Press (M. Maher ed.) 274–289

[42] T. Sato. Completed logic programs and their consistency. *Journal of Logic Programming* 9, 33–44 (1990)

[43] Y. Shimajiri, H. Seki and H. Itoh. Incorporating a Pruning Strategy into the Computation of Stable Models based on MGTP. *Proceedings of the 14th International Conference on Logic Programming*, Leuven, Belgium (1997) MIT Press (L. Naish, ed.) 376–390

[44] M. Thielscher. A nonmonotonic disputation-based semantics and proof procedure for logic programs. *JICSLP'96*, Bonn, Germany (1996), MIT Press (M. Maher ed.) 483–497

[45] F. Toni. Argumentation-based proof procedures for logic programming. Technical Report, Logic Programming Section, Department of Computing, Imperial College, London, UK (1997).

[46] F. Toni, A. C. Kakas. Computing the acceptability semantics. *Proceedings of the 3rd International Workshop on Logic Programming and Non-monotonic Reasoning*, Lexington, KY (1995) LNAI 928 Springer Verlag (V. Marek, A. Nerode and M. Truszczynski, eds) 401–415

[47] A. Torres. Negation as failure to support. *Proc. 2nd International Workshop on Logic Programming and Nonmonotonic Reasoning* MIT press (Pereira and Nerode eds.), Lisbon (1993) 223–243

[48] A. Van Gelder, K. A. Ross, J. S. Schlipf. Unfounded sets and the well-founded semantics for general logic programs. *Proceedings of the 7th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, Austin, Texas (1988) ACM Press, 221–230

[49] J.- H. You, L. Y. Yuan. On the equivalence of semantics for normal logic programs. *Journal of Logic Programming* 22(3):211-221 (1995)

A Proofs for section 2

Proof of corollary 10:
Let us define two approximations, acceptable* and acceptable**, of the definition of acceptability in definition 9 (at the second iteration) starting from the base case BC2:

- A set of hypotheses Δ is **acceptable*** (**acceptable****, respectively) to another set Δ_0 iff for all sets of hypotheses A, if A attacks $\Delta - \Delta_0$, then there exists a set of hypotheses D such that
  - D attacks $A - (\Delta_0 \cup \Delta)$ and
  - $D \subseteq A \cup \Delta \cup \Delta_0$ $\quad$ ($D \subseteq \Delta \cup \Delta_0$, respectively).

It is easy to prove that:

Lemma 56
If Δ is acceptable** to Δ_0 then Δ is acceptable* to Δ_0.
If Δ is acceptable* or acceptable** to Δ_0 then Δ is acceptable to Δ_0.
Δ is acceptable* to $\emptyset$ iff Δ is weakly stable.
Δ is acceptable** to $\emptyset$ iff Δ is admissible.

Therefore, the corollary trivially holds.

Well-founded model semantics via argumentation
The following definition is adapted from the (unpublished) paper [27].

Definition 57 Let P be a logic program, $\mathcal{H}$ the corresponding set of (negative) hypotheses and R the set of unary relations on $2^{\mathcal{H}}$. Then $\mathcal{F} : R \rightarrow R$ is defined as follows. For any $acc_{wf} \in R$ and $\Delta \in 2^{\mathcal{H}}$:

$\mathcal{F}(acc_{wf})(\Delta)$ iff for any attack A against Δ, there exists an attack D against A such that $acc_{wf}(D)$.

It is easy to show that this operator is monotonic with respect to $\subseteq$. Its least fixed point starting from the empty relation defines the **wf-acceptable** subsets of $\mathcal{H}$.

It is also easy to see that the union of two wf-acceptable sets is wf-acceptable and therefore there exists a maximal wf-acceptable set, which we will denote by Δ^{wf}.

Proof of theorem 14:
Without loss of generality we can assume that the program P contains only rules of the form $A \leftarrow notB_1, \ldots, notB_n$, $n \geq 0$, (i.e. rules with no positive conditions). Indeed, [3] proves that, for any program P, the attacking relation and therefore the well-founded semantics of P and of the *semantic kernel* of P, $sk(P)$, coincide, where the rules of $sk(P)$ are obtained by unfolding all atoms in the rules of P [3], and have therefore the form of the rules above.

We prove the theorem by exploiting the least-fixed point construction of the well-founded model and of the wf-acceptable sets. Indeed, the theorem directly follows from the following

Lemma 58 Let $M_i = (I_i^+, I_i^-)$ be the part of the well-founded model constructed at the ith iteration, and Δ_i^{wf} the maximal wf-acceptable set in the ith iteration $\mathcal{F}^i$ of $\mathcal{F}$, starting from the empty relation (i a countable ordinal). Then, $\Delta_{M_i} = \{not\, p \,|\, p \in I_i^-\} = \Delta_i^{wf}$.

Proof Base case ($i = 1$): By definition of well-founded model, $\Delta_{M_1} = \{not\, p \,|$ there are no rules for p in $P\}$. Hence, by definition of attack, $\Delta_{M_1} = \{not\, p \,|$ there are no attacks against $not\, p\}$. Therefore, by definition of $\mathcal{F}$, Δ_{M_1} is the maximal set belonging to $\mathcal{F}^1$. Thus, by definition of Δ_i^{wf}, $\Delta_{M_1} = \Delta_1^{wf}$.

Inductive step: Suppose that $\Delta_{M_i} = \Delta_i^{wf}$, for some $i > 1$.. We show that $\Delta_{M_{i+1}} = \Delta_{i+1}^{wf}$.

Let $not\, p$ belong to $\Delta_{M_{i+1}}$ and assume that A is an attack against $not\, p$. This attack can be identified with the body of some rule for p in P. Since $not\, p \in \Delta_{M_{i+1}}$, by construction of the well-founded model p belongs to the (greatest) unfounded set of M_i. The body of each rule for p is then inconsistent with M_i. This means that there exists a literal $not\, q$ in A such that $q \in I_i^+$ and therefore $P \cup \Delta_{M_i} \models q$. Hence Δ_{M_i} attacks A and therefore forms a defence for $\Delta_{M_{i+1}}$ which, by inductive hypothesis, is wf-acceptable at the ith iteration. Thus, $\Delta_{M_{i+1}}$ belongs to $\mathcal{F}^{i+1}$ and so $\Delta_{M_{i+1}} \subseteq \Delta_{i+1}^{wf}$.

Conversely, assume $not\, p \in \Delta_{i+1}^{wf}$ and consider any rule for p in P. The body A of this rule forms an attack against $not\, p$ and so there must be a defence D attacking A and belonging to $\mathcal{F}^i$. In particular, since Δ_i^{wf} is the maximal element of $\mathcal{F}^i$, $D \subseteq \Delta_i^{wf}$. This means that $P \cup \Delta_i^{wf} \models q$ for some $not\, q$ in A. By inductive hypothesis, $P \cup \Delta_{M_i} \models q$ and so $q \in I_i^+$. Thus, A is inconsistent with M_i. Hence $not\, p \in \Delta_{M_{i+1}}$ and so $\Delta_{M_{i+1}} \supseteq \Delta_{i+1}^{wf}$, as required.

B Proofs for section 3.1

Proof of theorem 23:

If-half: Let $\mathcal{T}$ be a well-founded pre-tree for Δ. By definition of pre-tree, $\mathcal{T}$ has even depth $2n$, $n \geq 0$. We prove that Δ is wf-acceptable by induction on n.

For $n = 0$, Δ is trivially wf-acceptable, as there are no attacks against Δ.

Let $n > 0$, and assume that the root of any well-founded pre-tree of depth at most $2(n-1)$ is wf-acceptable. All subtrees of $\mathcal{T}$ with defence nodes of level 2 as roots are well-founded and have depth at most $2(n-1)$. Then, by inductive hypothesis, every defence node of level 2 is wf-acceptable. Thus, for every attack against Δ (given by a node of level 1), there exists a wf-acceptable defence, i.e. Δ is wf-acceptable.

Only-if-half: Let Δ be wf-acceptable. It is easy to construct, inductively, at each level of iteration of the operator $\mathcal{F}$ whose least fixed point defines the wf-acceptable sets, a tree with root Δ satisfying conditions (PT1)-(PT4) in definition 21. By (transfinite) induction on the number of iterations of $\mathcal{F}$ we show that such a tree satisfies condition (PT5) as well:

Base case: $\Delta \in \mathcal{F}^1(\emptyset)$ iff there is no attack against Δ. Hence the tree has depth 0 and condition (PT5) is satisfied.

Inductive step: Assume that trees for wf-acceptable sets reached at level α have finite depth and let $\Delta \in \mathcal{F}^{\alpha+1}(\emptyset)$. Then, since Δ is wf-acceptable, for each attack A against Δ, and hence for each branch rooted at Δ, there exists a defence D which is wf-acceptable and $D \in \mathcal{F}^{\alpha}(\emptyset)$. By inductive hypothesis, for each such defence D we can build a well-founded pre-tree of finite depth. This then gives a well-founded pre-tree for Δ of finite depth.

Proof of theorem 28: Trivial, by lemma 56.

Proof of theorem 24: Suppose that there exists a hypothesis h in a defence node D which is the culprit in some attack node A. Then, it is easy to see that there exists a branch of the subtree of $\mathcal{T}$ rooted at A which is the same sequence of sets of hypotheses as that of a branch of the subtree of $\mathcal{T}$ rooted at D. Therefore, necessarily $\mathcal{T}$ has an attack node as leaf, and $\mathcal{T}$ is not a pre-tree (due to violation of condition

(PT3)). This is a contradiction.

Proof of theorem 30: Analogous to the proof of theorem 23. By (transfinite) induction on the least fixed point construction of the acceptability relation we can show that acceptable trees have finite branches.

C Proofs for section 3.2

Proof of theorem 32: By theorems 23 and 28, it is sufficient to show that there exists a (well-founded, admissible or weakly stable) restricted tree for Δ iff there exists a (well-founded, admissible or weakly stable) pre-tree for Δ. The if-half of this equivalence holds trivially, since restricted trees are subtrees of pre-trees. We prove the only-if-half as follows. Given a (well-founded, admissible or weakly stable) restricted tree $\mathcal{T}$ for Δ, we construct (a well-founded, admissible or weakly stable pre-tree) $\mathcal{T}'$ recursively as follows:

- Δ is the root of $\mathcal{T}'$, labelled as defence.
  Δ in $\mathcal{T}$ and Δ in $\mathcal{T}'$ iare said to "correspond" to each other.
- For every defence node D in $\mathcal{T}'$, the children of D in $\mathcal{T}'$ are all supersets (with respect to the Herbrand base of the given program P) of the children of the node "corresponding" to D in $\mathcal{T}$, and all such children are labelled as attack.
  Each such new attack node of $\mathcal{T}'$ is said to "correspond" to the attack node of $\mathcal{T}$ of which it is a superset.
- For every attack node A in $\mathcal{T}'$, if D is the child of the node "corresponding" to A in $\mathcal{T}$, then D is the child of A in $\mathcal{T}'$, labelled as defence.
  The defence node D in $\mathcal{T}$ and the new defence node D in $\mathcal{T}'$ are said to "correspond".

Trivially, $\mathcal{T}'$ has the same depth as $\mathcal{T}$, and $\mathcal{T}'$ satisfies condition (PT5). Also, it is easy to see that $\mathcal{T}'$ satisfies conditions (PT1) and (PT2), whatever the chosen instance of the function *hyp-branch*. Moreover, by construction and by monotonicity of attack, each node in $\mathcal{T}'$, except the root, attacks its parent node. We now prove that, for every node N child of a node N' in $\mathcal{T}'$:
($*$) N does not attack *hyp-branch*($\mathcal{T}'$,N'),
for the respective instances of *hyp-branch*.

Well-founded semantics: *hyp-branch*$_0$($\mathcal{T}'$,N')=$\emptyset$ for every N' and ($*$) trivially holds.

Admissible and weakly stable semantics: we have seen that $\mathcal{T}'$ has depth 2 at most and therefore *hyp-branch*($\mathcal{T}'$,N')=$\emptyset$=*hyp-branch*($\mathcal{T}$,N') for N' of level 0 and *hyp-branch*($\mathcal{T}'$,N')=Δ=*hyp-branch*($\mathcal{T}$,N') for N' of level 1. Thus, since $\mathcal{T}$ satisfies ($*$), $\mathcal{T}'$ does.

Thus, since, by construction, each attack node has exactly one child, $\mathcal{T}'$ satisfies condition (PT3), whatever the instance of *hyp-branch*. Furthermore, since every attack against a set of hypotheses is a superset of some minimal attack against the set, $\mathcal{T}'$ satisfies condition (PT4) (again, whatever the instance of *hyp-branch*).

Therefore, in the well-founded case, $\mathcal{T}'$ is a well-founded pre-tree and the theorem holds.

In the admissible and weakly stable semantics case, we need to prove additionally that, for every defence node D of level 2 in $\mathcal{T}'$, $D \subseteq$ *hyp-branch*$(\mathcal{T}', D)$. If $\mathcal{T}$ is admissible, then the defence node D^* of level 2 in $\mathcal{T}$, corresponding to D in $\mathcal{T}'$, is a subset of Δ and, since $D^* = D$, D is a subset of Δ in $\mathcal{T}'$. Therefore, $\mathcal{T}'$ is an admissible pre-tree. Finally, if $\mathcal{T}$ is weakly stable, then the defence node D^* of level 2 in $\mathcal{T}$, corresponding to D in $\mathcal{T}'$, is a subset of $\Delta \cup A^*$, with A^* the parent of D^* in $\mathcal{T}$, corresponding to the parent A of D in $\mathcal{T}'$. By transitivity of $\subseteq$ and since $D^* = D$, D is a subset of $\Delta \cup A$ in $\mathcal{T}'$. Therefore, $\mathcal{T}'$ is a weakly stable pre-tree.

Proof of theorem 35: (Analogous to that of theorem 32.) By theorem 30, we need to prove only that there exists an acceptable restricted tree for Δ iff there exists an acceptable pre-tree for Δ. The if-half of this equivalence holds trivially, since an acceptable restricted tree (for Δ) is a subtree of an acceptable pre-tree (for Δ). To prove the only-if-half, we construct an acceptable pre-tree $\mathcal{T}'$ for Δ from a given acceptable restricted tree $\mathcal{T}$ for Δ, by first expanding $\mathcal{T}$ into an intermediate (not necessarily acceptable) tree $\mathcal{T}_{int}$, and then eliminating some (unwanted) subtrees from $\mathcal{T}_{int}$. $\mathcal{T}_{int}$ is recursively constructed from $\mathcal{T}$ as follows:

- Δ is the root of $\mathcal{T}_{int}$, labelled as defence.
  Δ in $\mathcal{T}$ and Δ in $\mathcal{T}_{int}$ are said to "correspond" to each other.
- for every defence node D in $\mathcal{T}_{int}$, the children of D in $\mathcal{T}_{int}$ are all supersets (with respect to the Herbrand base of the given program P) of the children of the node "corresponding" to D in $\mathcal{T}$, and all such children are labelled as attack.
  Each such new attack node of $\mathcal{T}_{int}$ is said to "correspond" to the node of $\mathcal{T}$ of which it is a superset.
- for every attack node A in $\mathcal{T}_{int}$, "corresponding" to a node A' in $\mathcal{T}$:
  - If A' is a minimal or extended attack, then if the child of A' in $\mathcal{T}$ is D then the child of A in $\mathcal{T}_{int}$ is D, labelled as defence.
    D in $\mathcal{T}_{int}$ and D in $\mathcal{T}$ are said to "correspond".
  - If A' is a strict superset of every minimal or extended attack, then let D be the parent of A' in $\mathcal{T}$ and $A'' \subset A'$ be any extended attack child of D in $\mathcal{T}$ such that A'' is not strictly contained in any extended attack, i.e., A'' is a largest extended attack contained in A' (such a "maximal" attack exists since attacks are finite sets). If the child of A'' in $\mathcal{T}$ is D' then the child of A in $\mathcal{T}_{int}$ is D', labelled as defence.
    D' in $\mathcal{T}$ and D' in $\mathcal{T}_{int}$ are said to "correspond" to each other.

Note that, by construction, branches of $\mathcal{T}_{int}$ mirror exactly branches of $\mathcal{T}$, in the sense that every defence node of level i in a given branch of $\mathcal{T}_{int}$ is a defence node of level i in the corresponding branch of $\mathcal{T}$, and every attack node of level i in a given branch of $\mathcal{T}_{int}$ is a (non-necessarily strict) superset of an attack node of level i in the corresponding branch of $\mathcal{T}$. Note also that every branch in $\mathcal{T}$ may be mirrored by many branches in $\mathcal{T}_{int}$.

Trivially, $\mathcal{T}_{int}$ has the same depth as $\mathcal{T}$, and $\mathcal{T}_{int}$ satisfies condition (PT5). Also, it is easy to see that $\mathcal{T}_{int}$ satisfies conditions (PT1) and (PT2). Moreover, it is easy to see (by construction and by the monotonicity property of the relation attack) that each node N in $\mathcal{T}_{int}$, except the root, attacks its parent node N'. We will prove that $\mathcal{T}_{int}$ satisfies condition (PT3) but not (PT4), and therefore is not an acceptable pre-tree. We will modify $\mathcal{T}_{int}$ to obtain an acceptable pre-tree $\mathcal{T}'$

$\mathcal{T}_{int}$ satisfies condition (PT3).

By contradiction, suppose there exists a defence node D child of an attack node A in $\mathcal{T}_{int}$, such that D attacks A but D does not attack $A - \textit{hyp-branch}(\mathcal{T}_{int}, A)$. By construction, A corresponds to an attack node $A' \subseteq A$ in $\mathcal{T}$, and D is the child of A' in $\mathcal{T}$. Since $\mathcal{T}$ is an acceptable restricted tree, D attacks $A' - \textit{hyp-branch}(\mathcal{T}, A')$. Let $c \in A' - \textit{hyp-branch}(\mathcal{T}, A')$ be the culprit in A' in $\mathcal{T}$. Then, since we are assuming that D does not attack $A - \textit{hyp-branch}(\mathcal{T}_{int}, A)$, $c \notin A - \textit{hyp-branch}(\mathcal{T}_{int}, A)$, and thus there exists an attack node A^* which is an ancestor of A in $\mathcal{T}_{int}$, such that $c \in A^*$. By construction, A^* corresponds to an attack node $A^{*\prime}$ in $\mathcal{T}$, and $A^{*\prime}$ is a maximal attack node contained in A^* in $\mathcal{T}$. Since each branch in $\mathcal{T}_{int}$ mirrors some branch(es) in $\mathcal{T}$, there exists a branch in $\mathcal{T}$ with $A^{*\prime}$ ancestor of A'. Either $c \in A^{*\prime}$ or $c \notin A^{*\prime}$. In the first case c could not be chosen as culprit in A' in $\mathcal{T}$, and hence we have a contradiction. In the second case, the extended attack $A^{*\prime} \cup \{c\}$ belongs to $\mathcal{T}$, and is contained in A^*. Therefore, $A^{*\prime}$ is not the largest extended attack contained in A^*, and this contradicts the construction of $\mathcal{T}_{int}$.

Finally, trivially every defence node in $\mathcal{T}_{int}$ has exactly one child, labelled as attack, and so condition (PT3) is satisfied.

$\mathcal{T}_{int}$ violates condition (PT4).

Indeed, some attack node A child of a defence node D in $\mathcal{T}_{int}$, attacking some hypothesis $h \in D$, might not attack $D - \textit{hyp-branch}(\mathcal{T}_{int}, D)$, because $d \in A^*$, for some attack node A^* ancestor of A in $\mathcal{T}_{int}$.

Let us define $\mathcal{T}'$ to be $\mathcal{T}_{int}$ after having eliminated the subtrees with roots the attack nodes A, children of defence node D, such that A does not attack $D - \textit{hyp-branch}(\mathcal{T}_{int}, D)$. Trivially, by this deletion, $\mathcal{T}'$ satisfies condition (PT4). Moreover, $\mathcal{T}'$ satisfies all other properties, since $\mathcal{T}_{int}$ does, and is therefore an acceptable pre-tree.

D Proofs for section 4

Proof of theorem 43:
Soundness: Suppose there is a successful derivation $\mathcal{T}_1, \ldots, \mathcal{T}$ computing Δ from Δ_0 with respect to an allowed instance of the parameters. Trivially, by definition of derivation, $\Delta_0 \subseteq \Delta$. (In particular, for the well-founded semantics, by definition of *extend-root*, $\Delta_0 = \Delta$.) We prove that $\mathcal{T}$ can be mapped onto a restricted tree $\mathcal{T}'$, with the same root Δ. Then, by theorems 32 and 35, Δ is allowed with respect to the corresponding argumentation semantics. ($\mathcal{T}' = \mathcal{T}$ for well-founded, admissible and weakly stable semantics.) We use the following:

Lemma 59 For any *hyp-branch* and *extend-root*, $\mathcal{T}$ satisfies conditions (PT1), (PT2) and (PT5) of definition 3.1.

Proof of lemma 59: trivial, by definition of derivation.
Let $hyp\text{-}branch^{old}$ be defined as follows: for every node N, $hyp\text{-}branch^{old}(N) = hyp\text{-}branch(\mathcal{T}_i, N)$ for the (unique) i such that N is unmarked in $\mathcal{T}_i$ and marked in $\mathcal{T}_{i+1}$.

It is easy to see that, by definition of *hyp-branch* in all (allowed) special cases,
$hyp\text{-}branch^{old}(\Delta){=}hyp\text{-}branch(\mathcal{T}_i, \Delta){=}hyp\text{-}branch(\mathcal{T}, \Delta){=}\emptyset$,
for any $i \geq 1$. Moreover, by definition of *extend-root* in all (allowed) special cases, for any node N different from the root, $hyp\text{-}branch(\mathcal{T}, N){=}hyp\text{-}branch^{old}(N) \cup (\Delta - \Delta_0)$. Therefore, for any node N, $hyp\text{-}branch(\mathcal{T}, N){\supseteq}hyp\text{-}branch^{old}(N)$. This will help us to prove the following lemma:

Lemma 60 For any *hyp-branch* and *extend-root*, $\mathcal{T}$ satisfies condition (PT3) and condition

(RT4*) for every defence node D in $\mathcal{T}$, all *minimal* attacks against $D{-}hyp\text{-}branch(\mathcal{T}, D)$ are children of D in $\mathcal{T}$, and are labelled as attack.
(c.f. with condition (RT4) in definition 31)

Proof of lemma 60:

Condition (PT3) is satisfied: By definition of derivation, every defence node D, child of an attack node A, attacks $c \in A{-}hyp\text{-}branch^{old}(A)$. Since no culprit belongs to Δ (see part δ in the definition of derivation), $c \notin hyp\text{-}branch^{old}(A) {\cup} \Delta$ and so (from above) $c \notin hyp\text{-}branch(\mathcal{T}, A$. Thus, D attacks $A{-}hyp\text{-}branch(\mathcal{T}, A)$ and condition (PT3) is satisfied.

Condition (RT4*) is satisfied: By definition of derivation, for every defence node D in $\mathcal{T}$, all minimal attacks against $D{-}hyp\text{-}branch^{old}(D)$ are children of D in $\mathcal{T}$, labelled as attack. Any such attack is also an attack against $D{-}hyp\text{-}branch(\mathcal{T}, D$, due to the property above that, for any node N, $hyp\text{-}branch(\mathcal{T}, N){\supseteq}hyp\text{-}branch^{old}(N)$. Thus, condition (RT4*) is satisfied.

Well-founded, admissible and weakly stable semantics. Condition (RT4*) coincides with condition (RT4). Therefore, for the instances of *hyp-branch* and *extend-root* for the well-founded semantics, $\mathcal{T}$ is a well-founded restricted tree and soundness is proved. In the case of admissible and weakly stable semantics, we still need to prove that $\mathcal{T}$ satisfies the extra condition in definition 27. But, by definition of *extend-root*, for every defence node D of level 2, $extend\text{-}root(D, D{-}hyp\text{-}branch^{old}(D))$ holds, and thus $D{-}hyp\text{-}branch^{old}(D){\subseteq}\ \Delta - \Delta_0$ and, since $\Delta - \Delta_0\ {\subseteq}hyp\text{-}branch(\mathcal{T}, D)$, $D{-}hyp\text{-}branch(\mathcal{T}, D){=}\emptyset$. Therefore, $\mathcal{T}$ is an admissible and weakly stable restricted tree, for the respective instances of *hyp-branch*, and soundness is proved.

Acceptability semantics. Let us consider the tree $\mathcal{T}'$ obtained from $\mathcal{T}$ as follows: for every defence node D in $\mathcal{T}$ different from the root Δ and for every attack node A child of D in $\mathcal{T}$ such that A attacks some hypothesis $h \in \Delta$ but A does not attack $D - \{h\}$, delete from $\mathcal{T}$ the subtree rooted at A. By lemmas 59, 60 and by construction, $\mathcal{T}'$ satisfies conditions (PT1), (PT2), (PT3), (PT5) and (RT4*).

By contradiction, let us assume there exists a defence node D in $\mathcal{T}'$ and an extended attack A against D such that A is not a child of D in $\mathcal{T}'$. By definition of extended attack, $A = A' \cup \{c\}$, with $A \neq A'$, for some (minimal or extended) attack A' against D in $\mathcal{T}'$, and for some $c \in culprits(\mathcal{T}'_{A'})$, with $\mathcal{T}'_{A'}$ the subtree of $\mathcal{T}'$ rooted at A'. If A' is a child of D in $\mathcal{T}$, then, by definition of derivation of partial trees (case $\alpha.4$) A must be a child of D in $\mathcal{T}$, and, by construction, A must be a child of D in $\mathcal{T}'$ as well (contradiction). If A' is not a child of D in $\mathcal{T}$ and A' is a minimal attack against D, then contradiction (by (RT4*), see above). If A' is not a child of D in $\mathcal{T}$ and A' is an extended attack, then we repeat this argument recursively with A' in place of A. Since attacks are finite sets this recursive process will necessarily lead to one of the previous two contradicting cases. Therefore, $\mathcal{T}'$ satisfies condition (RT4) and is an acceptable restricted tree.

Completeness: Suppose $\Delta \supseteq \Delta_0$ is allowed with respect to any semantics. We show that there is a successful derivation $\mathcal{T}_1, \ldots, \mathcal{T}'$ for Δ_0, for any allowed, corresponding instance of *hyp-branch* and *extend-root*. In this derivation, the choice of defence nodes in step α (i.e. culprits in attack nodes and sets of hypotheses attacking the culprits) and, for acceptability, the choice performed by *extend-root*, are obtained from the corresponding choices in a given restricted tree $\mathcal{T}$ for Δ, whose existence is guaranteed by theorems 32 and 35.

- $\mathcal{T}_1$ consists of Δ_0, labelled as defence.
- $\mathcal{T}_2$ is obtained from $\mathcal{T}_1$ by selecting Δ_0, marking it, and adding as children of Δ_0 all *minimal* attacks against Δ_0, with each such child labelled as attack.

We say that the root Δ_0 of $\mathcal{T}_2$ and the root Δ of $\mathcal{T}$ "correspond". Trivially, $\mathcal{T}_2$ is obtained by a derivation step of kind δ. Moreover, since $\mathcal{T}$ is allowed, each child of Δ_0 in $\mathcal{T}_2$ "corresponds" to an attack node, child of Δ, in $\mathcal{T}$. Then, the derivation continues as follows.

Given $\mathcal{T}_i$ ($i \geq 2$), let N be any unmarked node in $\mathcal{T}_i$ and let N' be the "corresponding" node in $\mathcal{T}$. Then, $\mathcal{T}_{i+1}$ is obtained as follows:

- if N is an attack node and D' is the child of N' in $\mathcal{T}$, then D' is added as the (unmarked) child of N in $\mathcal{T}_{i+1}$, labelled as defence, and N is marked. Moreover, if c is the culprit in N', given by the function $culprits(\mathcal{T})$, then c is recorded as the culprit of N in $\mathcal{T}_{i+1}$.

  In addition, for acceptability, a step of kind $\alpha.4$ is applied, with respect to c.

The above definition of "correspondence" is extended, so that the defence node D' in $\mathcal{T}$ and the newly added defence node in $\mathcal{T}_{i+1}$ "correspond". In the acceptability case, since $\mathcal{T}$ is restricted acceptable, the set of hypotheses of each (extended) attack node A added as child of some defence node D in $\mathcal{T}_{i+1}$ via $\alpha.4$ necessarily coincides with the set of hypotheses in an attack node A' child of the defence node corresponding to D in $\mathcal{T}$. We say that A and A' "correspond".

- if N is a defence node, then a step of kind δ is applied, for the respective, given instances of *extend-root* for well-founded, admissible and weakly stable semantics, and, in the case of acceptability, for the following choice of *extend-root*:

  $extend\text{-}root(N, [N - hyp\text{-}branch(\mathcal{T}_i, N)] \cap \Delta)$.

Obviously, the given *extend-root* for acceptability satisfies definition 39.

Since $\mathcal{T}$ is a restricted tree, each (minimal) attack node added via a step of kind δ to $\mathcal{T}_i$ "corresponds" to a (minimal) attack node in $\mathcal{T}$.

Let $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}'$ be the (possibly infinite) sequence of trees, obtained as above, and such that all nodes of $\mathcal{T}'$ are marked. Since the restricted tree $\mathcal{T}$ is finite in depth, every tree in the sequence is finite in depth and therefore is a partial tree. Therefore, $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}'$ is a derivation for Δ_0. Moreover, for every attack node encountered in any of the partial trees in the sequence above there exists a defence for it, taken from $\mathcal{T}$. Therefore, all leaves of $\mathcal{T}'$ are defences, and $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}'$ is a successful derivation for Δ_0.

Let Δ' be the root of $\mathcal{T}'$. By soundness, Δ' is allowed and $\Delta_0 \subseteq \Delta'$. Therefore, we only need to prove that $\Delta' \subseteq \Delta$. By induction on the length of the given derivation:

Base case : $\Delta_0 \subseteq \Delta$.

Inductive case : let Δ_i be the root of $\mathcal{T}_i$ and Δ_{i+1} be the root of $\mathcal{T}_{i+1}$, and assume that $\Delta_i \subseteq \Delta$. Let N be the node selected in $\mathcal{T}_i$:

- if N is an attack node, then $\Delta_{i+1} = \Delta_i$ and thus $\Delta_{i+1} \subseteq \Delta$;
- if N is a defence node, then $\Delta_{i+1} = \Delta_i \cup N_{ext}$, with *extend-root*$(N, N_{ext})$. We prove, for each instance of *extend-root*, that $N_{ext} \subseteq \Delta$, and therefore $\Delta_{i+1} \subseteq \Delta$.

  Well-founded semantics : Trivial, since $N_{ext} = \emptyset$.

  Acceptable semantics : Trivial, by construction of *extend-root*.

  Admissible and weakly stable semantics :
  Since $\mathcal{T}$ is admissible/weakly stable, $N \subseteq$*hyp-branch*$(\mathcal{T}, N)$.
  Therefore, $N -$*hyp-branch*$(\mathcal{T}_i, N) \subseteq$*hyp-branch*$(\mathcal{T}, N) -$*hyp-branch*$(\mathcal{T}_i, N)$.
  It is easy to see that, by construction of $\mathcal{T}_i$, *hyp-branch*$(\mathcal{T}, N) =$ *hyp-branch*$(\mathcal{T}_i, N) \cup \Delta$.
  Thus, $N -$*hyp-branch*$(\mathcal{T}_i, N) \subseteq$*hyp-branch*$(\mathcal{T}_i, N) \cup \Delta -$*hyp-branch*$(\mathcal{T}_i, N)$.
  Therefore, since by definition 39 $N_{ext} = N -$*hyp-branch*$(\mathcal{T}_i, N)$, directly $N_{ext} \subseteq \Delta$.

E Proofs for section 5

Proof of theorem 53 For every allowed choice of the parameters *goal-branch* and *extend-goal-root*, by soundness of SLD-resolution and by monotonicity of the background logic, Δ is a pre-solution for G. Therefore, we need only to show that Δ is allowed. By theorems 32 and 35, it suffices to prove that there exists an allowed restricted tree $\mathcal{T}$ for Δ. $\mathcal{T}_n$ is not a restricted tree, in general, since its (attack-goal) nodes might not be sets of hypotheses. However, $\mathcal{T}_n$ can be mapped onto an (allowed) restricted tree. To show this, we first prove the following

Lemma 61

1. *$\mathcal{T}_n$ is finite in both depth and width.*
   Trivial, since the given goal-derivation is finite.

2. *Every defence-goal node in $\mathcal{T}_n$ is a set of hypotheses.* It holds since all (defence-goal) nodes in $\mathcal{T}_n$ are marked and since, by construction, defence-goal nodes are marked only when they do not contain positive atoms (case $\delta_\perp$).

3. *Every attack-goal node in $\mathcal{T}_n$ has exactly one child, labelled as defence-goal.* Trivial, since no attack-goal in $\mathcal{T}_n$ is a leaf, due to the definition of successful goal-derivation, and since, by construction, case α_{naf}, attack-goal nodes have only one child.

4. *Every node in $\mathcal{T}_n$ is labelled either as defence- or as attack-goal, but not as both.* Trivial, since, by definition of goal-derivation, each node is labelled when it is created, and every time the node is replaced by other nodes the labelling does not change.

5. *The root, Δ, of $\mathcal{T}_n$ is labelled as defence-goal.* Trivial, since the root of $\mathcal{T}_1$ is labelled as defence-goal, and by part 4 of this lemma.

6. $\Delta \cap culprits(\mathcal{T}_n) = \emptyset$. It follows directly by induction, due to the check on culprits in case δ_{naf}.

7. *For every node N child of a node N' in $\mathcal{T}_n$, $P \cup N \models p$, for some hypothesis $not\, p \in N'$.* It follows from cases α_{naf} and δ_{naf}, and from the soundness of SLD-resolution, used in α_{atom} and δ_{atom}.

8. *For every defence-goal node D in $\mathcal{T}_n$, for every hypothesis $not\, p$ in D and for every minimal set of hypotheses A such that $P \cup A \models p$, there exists an attack-goal G_A child of D in $\mathcal{T}_n$ such that $P \cup A \models G_A$.* It follows from α_{naf} and from the completeness of SLD-resolution used in α_{atom}.

We build $\mathcal{T}$ recursively from $\mathcal{T}_n$, independently from the choice of the parameters. In all cases, defence nodes of $\mathcal{T}$ are defence-goal nodes of $\mathcal{T}_n$, and attack nodes of $\mathcal{T}$ are obtained by unfolding atoms in attack-goal nodes of $\mathcal{T}_n$. Some attack-goal nodes in $\mathcal{T}_n$ might not give rise to any attack nodes in $\mathcal{T}$. Moreover, some attack-goal nodes in $\mathcal{T}_n$ might give rise to many attack nodes in $\mathcal{T}$. The recursive construction of $\mathcal{T}$, starting from the root, is given as follows:

(R) Δ is the root of $\mathcal{T}$, labelled as defence. (We say that the root of $\mathcal{T}_n$ and the root of $\mathcal{T}$ "correspond".)

(DA) If D is a defence node of $\mathcal{T}$, for each attack-goal node G_A which is a child of the defence-goal node "corresponding" to D in $\mathcal{T}_n$, consider all minimal sets of hypotheses A such that $P \cup A \models G_A$. For every such A, if

1. A attacks $not\, q$ in D and $not\, q \notin$ *hyp-branch*($\mathcal{T}$,D) [10], and
2. (for acceptability only) $culprits(\mathcal{T}_{n\,G_A}) \cap A = \emptyset$, where $\mathcal{T}_{n\,G_A}$ is the subtree of $\mathcal{T}_n$ rooted at G_A,

[10] The recursive construction is well-defined as *hyp-branch*($\mathcal{T}$,D) depends only on the part of $\mathcal{T}$ that has already been built.

then, A is a child of D in $\mathcal{T}$, labelled as attack. (We say that G_A and each such A "correspond".)

(AD) If A is an attack node in $\mathcal{T}$, let G_A be the attack-goal node in $\mathcal{T}_n$ such that A and G_A "correspond", and D be the defence-goal (set of hypotheses) node child of G_A in $\mathcal{T}_n$. Then, D is the child of A in $\mathcal{T}$, labelled as defence. (We say that D in $\mathcal{T}_n$ and D in $\mathcal{T}$ "correspond".)

It is easy to see that, for each node N in $\mathcal{T}$ and corresponding goal-node N' in $\mathcal{T}_n$, $\textit{hyp-branch}(\mathcal{T},N) \supseteq \textit{goal-branch}(\mathcal{T}_n,N')$.

Independently from the instances of the parameters, by construction and lemma 61 (part 3), $\mathcal{T}$ trivially satisfies conditions (PT1), (PT2) and (PT5) of definition 17. Furthermore, again by construction and lemma 61, each attack node in $\mathcal{T}$ has exactly one child, labelled as defence, and, by lemma 61 (part 7), by definition of attack, every node in $\mathcal{T}$, different from the root, attacks its parent node. Therefore, to prove that $\mathcal{T}$ satisfies condition (PT3), we only need to prove that

(i) for every attack node N in $\mathcal{T}$, the culprit of N does not belong to $\textit{hyp-branch}(\mathcal{T}, N)$.

Furthermore, to prove that $\mathcal{T}$ satisfies condition (RT4) in definition 31 (for well-founded/admissible and weakly stable semantics) and 34 (for acceptability) we only need to prove that

(ii) for every defence node D in $\mathcal{T}$ and minimal attack A against $D - \textit{hyp-branch}(\mathcal{T}, D)$, A is a child of D in $\mathcal{T}$,

and in addition, for acceptability only,

(iii) for every defence node D in $\mathcal{T}$ and extended attack A, with respect to $\mathcal{T}$, against $D - \textit{hyp-branch}(\mathcal{T}, D)$, A is a child of D in $\mathcal{T}$.

Finally, for admissibility and weak stability only, we need to prove that

(iv) for every defence node D of level 2 in $\mathcal{T}$, $D - \textit{hyp-branch}(\mathcal{T},D) = \emptyset$.

Well-founded semantics.

(i) is trivially satisfied since $\textit{goal-branch}(N) = \textit{hyp-branch}(N) = \emptyset$.

(ii) holds by lemma 61 (part 8), by construction of $\mathcal{T}$, and by definition of attack and of *goal-branch*.

Admissible and weakly stable semantics.

(iv) holds trivially by definition of derivation of goal-trees, case δ_{naf}, and by definition of *extend-goal-root* for these semantics.

Therefore, $\mathcal{T}$'s depth is 0 or 2 (the same as $\mathcal{T}_n$'s depth).

(i): By contradiction, given an attack node N, let the culprit of N belong to $\textit{hyp-branch}(\mathcal{T}, N)$. Let N' be the child of N in $\mathcal{T}$. Then N' is a defence node of level 2, and, since $\textit{hyp-branch}(\mathcal{T}, N = \Delta$, N' attacks Δ. By completeness of SLD, there exists an attack-goal node G_A child of Δ in $\mathcal{T}_n$ such that G_A is a leaf of $\mathcal{T}_n$ (contradiction).

(ii) follows directly from lemma 61 (part 8), by construction of $\mathcal{T}$ and by definition of attack.

Acceptability semantics.

(i) For each defence node N' child of an attack node N in $\mathcal{T}$, the culprit in N (i.e. the culprit in G_N) does not belong to *hyp-branch*($\mathcal{T}$,N), for the following reasons. The tests in case δ_{naf}, while building $\mathcal{T}_n$, ensures that the culprit in G_N does not belong to the root or to a defence node contained in *hyp-branch*($\mathcal{T}_n$,N), and hence in *hyp-branch*($\mathcal{T}$,N). Moreover, case (DA), part 2, in the construction of $\mathcal{T}$ ensures that the culprit in N does not belong to attack nodes in *hyp-branch*($\mathcal{T}$,N).

To prove (ii) and (iii), note that, as before, all minimal attacks are potentially represented in $\mathcal{T}_n$ by attack-goals. Moreover, by α_{naf}, all extended attacks are potentially represented in $\mathcal{T}_n$ by attack-goals. Actual attacks are obtained in $\mathcal{T}$ by construction from $\mathcal{T}_n$. However, some of the attacks are removed, namely those that would cause the choice of some culprit to be invalid (see case (DA), part 2). We can show that these attacks do in fact belong to $\mathcal{T}$ anyway.

Suppose for instance that A is an attack that obtained from an attack-goal G_A in $\mathcal{T}_n$ but removed by test (DA), part 2. Then, A contains the culprit, $not\, q$, of G_A and at least another hypothesis, $not\, p$, which is not explicitly present in the goal G_A and such that $not\, p$ is the culprit in an attack-goal G' in the subtree with root G_A in $\mathcal{T}_n$. Then, by the definition of α_{naf}, an extended attack-goal $G_A \cup \{not\, p\}$ must be in $\mathcal{T}_n$. If the attack A' corresponding to $G_A \cup \{not\, p\}$ in $\mathcal{T}$ is not removed during the construction of $\mathcal{T}$, then A is in $\mathcal{T}$, since A' is exactly A. If instead such A' is removed, we can repeat the same argument and be sure that, eventually, there is an extended attack-goal in $\mathcal{T}_n$ that gives rise to A, otherwise we would contradict the finiteness of the number of (minimal and extended) attacks generated in the finite successful derivation that leads to $\mathcal{T}_n$.

F Proofs for section 6

Proof of theorem 55 (sketch): The theorem is proved by defining a partial order on the set of Eshghi-Kowalski derivations, as in [11], and then using induction on this partial order, exploiting the direct correspondence between the following pairs of derivation steps in our proof procedure (left) and in the Eshghi-Kowalski procedure (right):

$\alpha_{atom} \Leftrightarrow C_1$ in the consistency phase

α_{naf}, first item $\Leftrightarrow C_2$ in the consistency phase

$\delta_{atom} + \delta_{\perp} \Leftrightarrow A_1$ in the abductive phase

δ_{naf}, first item $\Leftrightarrow A_2$ in the abductive phase.

Therefore, cases α and δ in our proof procedure play the role of consistency and abductive phases, respectively, in the Eshghi-Kowalski procedure. The interleaving of consistency and abductive phases in the Eshghi-Kowalski procedure corresponds to the interleaving between α and δ steps in our proof procedure. However, whereas consistency and abductive phases are defined recursiveley in terms of each other, α and δ steps are not (the interleaving is guaranteed by the selection strategy).