Argumentation-based Proof Procedures for Credulous and Sceptical Non-monotonic Reasoning

Phan Minh Dung[1], Paolo Mancarella[2], and Francesca Toni[3]

[1] Division of Computer Science, Asian Institute of Technology, GPO Box 2754, Bangkok 10501, Thailand
dung@cs.ait.ac.th
[2] Dipartimento di Informatica, Università di Pisa, Corso Italia 40, I-56125 Pisa, Italy
p.mancarella@di.unipi.it
[3] Department of Computing, Imperial College of Science, Technology and Medicine, 180 Queen's Gate, London SW7 2BZ, U.K.
ft@doc.ic.ac.uk

Abstract. We define abstract proof procedures for performing credulous and sceptical non-monotonic reasoning, with respect to the argumentation-theoretic formulation of non-monotonic reasoning proposed in [1]. Appropriate instances of the proposed proof procedures provide concrete proof procedures for concrete formalisms for non-monotonic reasoning, for example logic programming with negation as failure and default logic. We propose (credulous and sceptical) proof procedures under different argumentation-theoretic semantics, namely the conventional stable model semantics and the more liberal partial stable model or preferred extension semantics. We study the relationships between proof procedures for different semantics, and argue that, in many meaningful cases, the (simpler) proof procedures for reasoning under the preferred extension semantics can be used as sound and complete procedures for reasoning under the stable model semantics. In many meaningful cases still, proof procedures for credulous reasoning under the preferred extension semantics can be used as (much simpler) sound and complete procedures for sceptical reasoning under the preferred extension semantics. We compare the proposed proof procedures with existing proof procedures in the literature.

1 Introduction

In recent years argumentation [1, 3, 4, 6, 12, 15, 21, 23, 24, 29, 30, 32] has played an important role in understanding many non-monotonic formalisms and their semantics, such as logic programming with negation as failure, default logic and autoepistemic logic. In particular, Eshghi and Kowalski [9] have given an interpretation of negation as failure in Logic Programming as a form of assumption based reasoning (abduction). Continuing this line of work, Dung [5] has given a declarative understanding of this assumption based view, by formalizing the concept that an assumption can be safely accepted if "there is no evidence to the contrary". It has also been shown in [5] that the assumption based view provides a unifying framework for different semantics of logic programming. Later, this view has been further put forward [1, 6, 12] by the introduction the notions of attack and counterattacks between sets of assumptions, finally leading to an argumentation-theoretic understanding of the semantics of logic programming and nonmonotonic reasoning. In particular, Dung [6] has introduced an abstract framework of argumentation, that consists of a set of arguments and an attack relation between them. However, this abstract framework leaves open the question of how the arguments and their attack relationship are defined. Addressing this issue, Bondarenko et al. [1] has defined an abstract, argumentation-theoretic assumption-based framework to non-monotonic reasoning that can be instantiated to capture many of the existing approaches to non-monotonic reasoning, namely logic programming with negation as failure, default logic [25], (many cases of) circumscription [16], theorist [22], autoepistemic logic [18] and non-monotonic modal logics [17]. The semantics of argumentation can be used to characterize a number of alternative semantics for non-monotonic reasoning, each of which can be the basis for credulous and sceptical reasoning. In particular, three semantics have been proposed in [1, 6] generalizing, respectively, the semantics of admissible scenaria for logic programming [5], the semantics of preferred extensions [5] or partial stable models [26] for logic programming, and the conventional semantics of stable models [10] for logic programming as well as the standard semantics of theorist [22], circumscription [16], default logic [25], autoepistemic logic [18] and non-monotonic modal logic [17].

More in detail, Bondarenko et al. understand non-monotonic reasoning as extending theories in some monotonic language by means of sets of assumptions, provided they are "appropriate" with respect to some requirements. These are expressed in argumentation-theoretic terms, as follows. According to the semantics of admissible extensions, a set of assumptions is deemed "appropriate" iff it does not attack itself and it attacks all sets of assumptions which attack it. According to the semantics of preferred extensions, a set of assumptions is deemed "appropriate" iff it is maximally admissible, with respect to set inclusion. According to the semantics of stable extensions, a set of assumptions is deemed "appropriate" iff it does not attack itself and it attacks every assumption which it does not belong.

Given any such semantics of extensions, credulous and sceptical non-monotonic reasoning can be defined, as follows. A given sentence in the underlying monotonic language is a credulous non-monotonic consequence of a theory iff it holds in some extension of the theory that is deemed "appropriate" by the chosen semantics. It is a sceptical non-monotonic consequence iff it holds in all extensions of the theory that are deemed "appropriate" by the chosen semantics.

In this paper we propose abstract proof procedures for performing credulous and sceptical reasoning under the three semantics of admissible, preferred and stable extensions, concentrating on the special class of *flat* frameworks. This class includes logic programming with negation as failure and default logic.

We define all proof procedures parametrically with respect to a proof procedure computing the semantics of admissible extensions. A number of such procedures have been proposed in the literature, e.g. [9, 5, 7, 8, 15].

We argue that the proof procedures for reasoning under the preferred extension semantics are "simpler" than those for reasoning under the stable extension semantics. This is an interesting argument in that, in many meaningful cases (e.g. when the frameworks are order-consistent [1]), the proof procedures for reasoning under the preferred extension semantics can be used as sound and complete procedures for reasoning under the stable model semantics.

The paper is organized as follows. Section 2 summarises the main features of the approach in [1]. Section 3 gives some preliminary definitions, used later on in the paper to define the proof procedures. Sections 4 and 5 describe the proof procedures for performing credulous reasoning under the preferred and stable extension semantics, respectively. Sections 6 and 7 describe the proof procedures for performing sceptical reasoning under the stable and preferred extension semantics, respectively. Section 8 compares the proposed proof procedures with existing proof procedures proposed in the literature. Section 9 concludes.

2 Argumentation-based Semantics

In this section we briefly review the notion of assumption-based framework [1], showing how it can be used to extend any deductive system for a monotonic logic to a non-monotonic logic.

A *deductive system* is a pair ($\mathcal{L}$, $\mathcal{R}$) where

- $\mathcal{L}$ is a formal language consisting of countably many sentences, and
- $\mathcal{R}$ is a set of inference rules of the form

$$\frac{\alpha_1, \ldots, \alpha_n}{\alpha}$$

  where $\alpha, \alpha_1, \ldots, \alpha_n \in \mathcal{L}$ and $n \geq 0$. If $n = 0$, then the inference rule is an axiom.

A set of sentences $T \subseteq \mathcal{L}$ is called a *theory*.

A *deduction* from a theory T is a sequence $\beta_1, \ldots, \beta_m$, where $m > 0$, such that, for all $i = 1, \ldots, m$,

- $\beta_i \in T$, or
- there exists $\dfrac{\alpha_1, \ldots, \alpha_n}{\beta_i}$ in $\mathcal{R}$ such that $\alpha_1, \ldots, \alpha_n \in \{\beta_1, \ldots, \beta_{i-1}\}$.

$T \vdash \alpha$ means that there is a deduction (of α) from T whose last element is α. $Th(T)$ is the set $\{\alpha \in \mathcal{L} \mid T \vdash \alpha\}$. Deductive systems are *monotonic*, in the sense that $T \subseteq T'$ implies $Th(T) \subseteq Th(T')$. They are also *compact*, in the sense that $T \vdash \alpha$ implies $T' \vdash \alpha$ for some finite subset T' of T.

Given a deductive system ($\mathcal{L}$, $\mathcal{R}$), an *argumentation-theoretic framework* with respect to ($\mathcal{L}$, $\mathcal{R}$) is a tuple $\langle T, Ab, \overline{} \rangle$ where

– T, $Ab \subseteq \mathcal{L}$, $Ab \neq \{\}$
– $\overline{}$ is a mapping from Ab into $\mathcal{L}$. $\overline{\alpha}$ is called the *contrary* of α.

The theory T can be viewed as a given set of beliefs, and Ab as a set of candidate *assumptions* that can be used to extend T. An *extension* of a theory T is a theory $Th(T \cup \Delta)$, for some $\Delta \subseteq Ab$. Sometimes, informally, we refer to the extension simply as $T \cup \Delta$ or Δ.

Given a deductive system $(\mathcal{L}, \mathcal{R})$ and an argumentation-theoretic framework $\langle T, Ab, \overline{}\rangle$ with respect to $(\mathcal{L}, \mathcal{R})$, the problem of determining whether a given sentence σ in $\mathcal{L}$ is a *non-monotonic consequence* of the framework is understood as the problem of determining whether there exist "appropriate" extensions $\Delta \subseteq Ab$ of T such that $T \cup \Delta \vdash \sigma$. In particular, σ is a *credulous* non-monotonic consequence of $\langle T, Ab, \overline{}\rangle$ if there exists *some* "appropriate" extension of T. Many logics for default reasoning are credulous in this same sense, differing however in the way they understand what it means for an extension to be "appropriate". Some logics, in contrast, are *sceptical*, in the sense they they require that σ belong to *all* "appropriate" extensions. However, the semantics of any of these logics can be made sceptical or credulous, simply by varying whether a sentence is deemed to be a non-monotonic consequence of a theory if it belongs to all "appropriate" extensions or if it belongs to some "appropriate" extension.

A number of notions of "appropriate" extensions are given in [1], for any argumentation-theoretic framework $\langle T, Ab, \overline{}\rangle$ with respect to $(\mathcal{L}, \mathcal{R})$. All these notions are formulated in argumentation-theoretic terms, with respect to a notion of "attack" defined as follows. Given a set of assumptions $\Delta \subseteq Ab$:

– Δ *attacks* an assumption $\alpha \in Ab$ iff $T \cup \Delta \vdash \overline{\alpha}$
– Δ *attacks* a set of assumptions $\Delta' \subseteq Ab$ iff Δ attacks an assumption α, for some $\alpha \in \Delta'$.

In this paper we will consider the notions of "stable", "admissible" and "preferred" extensions, defined below.

Let a set of assumptions $\Delta \subseteq Ab$ be *closed* iff $\Delta = \{\alpha \in Ab \,|\, T \cup \Delta \vdash \alpha\}$. Then, $\Delta \subseteq Ab$ is *stable* if and only if

1. Δ is closed,
2. Δ does not attack itself, and
3. Δ attacks α, for every assumption $\alpha \notin \Delta$.

Furthermore, $\Delta \subseteq Ab$ is *admissible* if and only if

1. Δ is closed,
2. Δ does not attack itself, and
3. for each closed set of assumptions $\Delta' \subseteq Ab$,
   if Δ' attacks Δ then Δ attacks Δ'.

Finally, $\Delta \subseteq Ab$ is *preferred* if and only if Δ is maximally admissible, with respect to set inclusion.

In general, every admissible extension is contained in some preferred extension. Moreover, every stable extension is preferred (and thus admissible) [1] but not vice versa. However, in many cases, e.g. for *stratified* and *order-consistent* argumentation-theoretic frameworks (see [1]), preferred extensions are always stable[1].

In this paper we concentrate on *flat* frameworks [1], namely frameworks in which every set of assumptions $\Delta \subseteq Ab$ is closed. For this kind of frameworks, the definitions of admissible and stable extensions can be simplified by dropping condition 1 and by dropping the requirement that Δ' be closed in condition 3 of the definition of admissible extension. In general, if the framework is flat, both admissible and preferred extensions are guaranteed to exist. Instead, even for flat frameworks, stable extensions are not guaranteed to exist. However, in many cases, e.g. for *stratified* argumentation-theoretic frameworks [1], stable extensions are always guaranteed to exist.

Different logics for default reasoning differ, not only in whether they are credulous or sceptical and how they interpret the notion of what it means to be an "appropriate" extension, but also in their underlying framework.

Bondarenko et al. [1] show how the framework can be instantiated to obtain theorist [22], (some cases of) circumscription [16], autoepistemic logic [18], non-monotonic modal logics [17], default logic [25], and logic programming, with respect to, e.g., the semantics of stable models [10] and partial stable models [26], the latter being equivalent [13] to the semantics of preferred extensions [5]. They also prove that the instances of the framework for default logic and logic programming are flat.

Default logic is the instance of the abstract framework $\langle T, Ab, \overline{} \rangle$ where the $\vdash$ is first-order logic augmented with domain-specific inference rules of the form

$$\frac{\alpha_1, \ldots, \alpha_m, M\beta_1, \ldots, M\beta_n}{\gamma}$$

where α_i, β_j, γ are sentences in classical logic. T is a classical theory and Ab consists of all expressions of the form $M\beta$ where β is a sentence of classical logic. The contrary $\overline{M\beta}$ of an assumption $M\beta$ is $\neg\beta$. The conventional semantics of extensions of default logic [25] corresponds to the semantics of stable extensions of the instance of the abstract framework for default logic [1]. Moreover, default logic inherits the semantics of admissible and preferred extensions, simply by being an instance of the framework.

Logic programming is the instance of the abstract framework $\langle T, Ab, \overline{} \rangle$ where T is a logic program, the assumptions in Ab are all negations $not\, p$ of atomic sentences p, and the contrary $\overline{not\, p}$ of an assumption is p. $\vdash$ is Horn logic provability, with assumptions, $not\, p$, understood as new atoms, p^* (see [9]). The logic programming semantics of stable models [10], admissible scenaria [5], and partial stable models [26]/preferred extensions [5] correspond to the semantics of stable, admissible and preferred extensions, respectively, of the instance of the abstract framework for logic programming [1].

1 See the Appendix for the definition of stratified and order-consistent frameworks.

In the remainder of the paper we will concentrate on computing credulous and sceptical consequences under the semantics of preferred and stable extensions. We will rely upon a proof procedure for computing credulous consequences under the semantics of admissible extensions (see Sect. 8 for a review of such procedures). Note that we ignore the problem of computing sceptical consequences under the semantics of admissible extensions as, for flat frameworks, this problem reduces to that of computing monotonic consequences in the underlying deductive system. Indeed, in flat frameworks, the empty set of assumptions is always admissible.

We will propose abstract proof procedures, but, for simplicity, we will illustrate their behaviour within the concrete instance of the abstract framework for logic programming.

3 Preliminaries

In the sequel we assume that a framework is given and we omit mentioning it explicitly if clear by the context.

Let S be a set of sets. A subset B of S is called a **base** of S if for each element s in S there is an element b in B such that $b \subseteq s$.

We assume that the following procedures are defined, where α is a sentence in $\mathcal{L}$ and $\Delta \subseteq Ab$ is a set of assumptions:

- $support(\alpha, \Delta)$ computes a set of sets $\Delta' \subseteq Ab$ such that $\alpha \in Th(T \cup \Delta')$ and $\Delta' \supseteq \Delta$.
  $support(\alpha, \Delta)$ is said to be **complete** if it is a base of the set $\{\Delta' \subseteq Ab | \alpha \in Th(T \cup \Delta') \text{ and } \Delta' \supseteq \Delta\}$.
- $adm\_expand(\Delta)$ computes a set of sets $\Delta' \subseteq Ab$ such that $\Delta' \supseteq \Delta$ and Δ' is admissible.
  $adm\_expand(\Delta)$ is said to be **complete** if it is a base of the set of all admissible supersets of Δ.

We will assume that the above procedures are nondeterministic. We will write, e.g.

$$A := support(\alpha, \Delta)$$

meaning that the variable A is assigned, if any, a result of the procedure *support*. Such a statement represents a backtracking point, which may eventually fail if no further result can be produced by *support*.

The following example illustrates the above procedures.

Example 1. Consider the following logic program

$$\begin{array}{l} p \leftarrow q, not\ r \\ q \leftarrow not\ s \\ t \leftarrow not\ h \\ f \end{array}$$

and the sentence p. Possible outcomes of the procedure $\mathit{support}(p, \{\})$ are $\Delta_1 = \{\mathit{not}\, s, \mathit{not}\, r\}$ and $\Delta_2 = \{\mathit{not}\, s, \mathit{not}\, r, \mathit{not}\, f\}$. Possible outcomes of the procedure $\mathit{adm\_expand}(\Delta_1)$ are Δ_1 and $\Delta_1 \cup \{\mathit{not}\, h\}$. No possible outcomes exist for $\mathit{adm\_expand}(\Delta_2)$.

Note that different implementations for the above procedures are possible. In all examples in the remainder of the paper we will assume that *support* and *adm_expand* return minimal sets. In the above example, Δ_1 is a minimal support whereas Δ_2 is not, and Δ_1 is a minimal admissible expansion of Δ_1 whereas $\Delta_1 \cup \{\mathit{not}\, h\}$ is not.

4 Computing Credulous Consequences under Preferred Extensions

To show that a sentence is a credulous consequence under the preferred extension semantics, we simply need to check the existence of an admissible set of assumptions which entails the desired sentence. This can be done by:

- finding a support set for the sentence
- showing that the support set can be extended into an admissible extension.

Proof procedure 4.1 (Credulous Preferred Extensions).

> $CPE(\alpha)$:
>> $S := \mathit{support}(\alpha, \{\})$;
>> $\Delta := \mathit{adm\_expand}(S)$;
>> **return** Δ

Notice that the two assignments in the procedure are backtracking points, due to the nondeterministic nature of both *support* and *adm_expand*.

Example 2. Consider the following logic program

$$\begin{array}{l} p \leftarrow \mathit{not}\ s \\ s \leftarrow q \\ q \leftarrow \mathit{not}\ r \\ r \leftarrow \mathit{not}\ q \end{array}$$

and the sentence p. The procedure $CPE(p)$ will perform the following steps:

- first the set $S = \{\mathit{not}\ s\}$ is generated by $\mathit{support}(p, \{\})$
- then the set $\Delta = \{\mathit{not}\ s, \mathit{not}\ q\}$ is generated by $\mathit{adm\_expand}(S)$
- finally, Δ is the set returned by the procedure

Consider now the conjunction p, q. The procedure $CPE((p, q))$[2] would fail, since

[2] Note that, in the instance of the framework of [1] for logic programming, conjunction of atoms are not part of the underlying deductive system. However, conjunctions can be accommodated by additional program clauses. E.g., in the given example, the logic program can be extended by $t \leftarrow p, q$, and CPE can be called for t.

– $S = \{not\ s, not\ r\}$ is generated by $support((p, q), \{\})$
– there exists no admissible set $\Delta \supseteq S$.

Theorem 1 (Soundness and Completeness of CPE).

1. *If $CPE(\alpha)$ succeeds then there exists a preferred extension Δ such that $\alpha \in Th(T \cup \Delta)$.*
2. *If both support and adm_expand are complete then for each preferred extension E there exist appropriate selections such that $CPE(\alpha)$ returns $\Delta \subseteq E$.*

Proof.

1. It follows immediately from the fact that each admissible set of assumptions could be extended into a preferred extension.
2. Let E be a preferred extension such that $\alpha \in Th(T \cup E)$. Since $support(\alpha, \{\})$ is complete, there is a set $S \subset E$ such that S could be computed by $support(\alpha, \{\})$. From the completeness of *adm_expand*, it follows that there is $\Delta \subseteq E$ such that Δ is computed by $adm\_expand(S)$. □

5 Computing Credulous Consequences under Stable Extensions

A stable model is nothing but a preferred extension which entails either α or its contrary, for each assumption α [1]. Hence, to show that a sentence is a credulous consequence under the stable model semantics, we simply need to find an admissible extension which entails the sentence and which can be extended into a stable model.

We assume that the following procedures are defined:

– $full\_cover(\Gamma)$ returns *true* iff the set of sentences Γ entails any assumption or its contrary, $false$ otherwise;
– $uncovered(\Gamma)$ nondeterministically returns, if any, an assumption which is undefined, given Γ, i.e. neither the assumption nor its contrary is entailed by Γ.

In the following procedure CSM, both $full\_cover$ and *uncovered* will be applied to sets of assumptions only.

Proof procedure 5.1 (Credulous Stable Models).

$CSM(\alpha)$:
    $\Delta := CPE(\alpha)$;
    loop
        if $full\_cover(\Delta)$
            then **return** Δ
            else $\beta := uncovered(\Delta)$
                $\Delta := adm\_expand(\Delta \cup \{\beta\})$;
        end if
    end loop

Note that CSM is a non-trivial extension of CPE: once an admissible extension is selected, as in CPE, CSM needs to further expand the selected admissible extension, if possible, to render it stable. This is achieved by the main loop in the procedure.

Clearly, the above procedure may not terminate if the underlying framework $\langle T, Ab, \overline{} \rangle$ contains infinitely many assumptions, since in this case the main loop may go on forever. In the following theorem we assume that the set of assumptions Ab is finite.

Theorem 2 (Soundness and Completeness of CSM).
Let $\langle T, Ab, \overline{} \rangle$ be a framework such that Ab is finite.

1. *If $CSM(\alpha)$ succeeds then there exists a stable extension Δ such that $\alpha \in Th(T \cup \Delta)$.*
2. *If both support and adm_expand are complete then for each stable extension E such that $\alpha \in Th(T \cup E)$ there exist appropriate selections such that $CSM(\alpha)$ returns E.*

Proof. The theorem follows directly from theorem 3. □

The CSM procedure is based on backward-chaining in contrast to the procedure of Niemelä et al. [19, 20] that is based on forward-chaining. We explain the difference between the two procedures in the following example.

Example 3.

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

Assume that the given query is p. The CSM procedure would compute $\{not\ q\}$ as a support for p. The procedure *adm_expand*($\{not\ q\}$) will produce $\Delta = \{not\ q\}$ as its result. Since Δ covers all assumptions, Δ is the result produced by the procedure. Niemelä et. al procedure would start by picking an arbitrary element from $\{not\ p, not\ q, not\ r\}$ and start to apply the Fitting operator to it to get a fixpoint. For example, *not* r may be selected. Then the set $B = \{q, not\ r\}$ is obtained. Since there is no conflict in B and B does not cover all the assumptions, *not* p will be selected. Since $\{not\ p, q, not\ r\}$ covers all assumptions, a test to check whether p is implied from it is performed with $false$ as the result. Therefore backtracking will be made and *not* q will be selected leading to the expected result.

A drawback of Niemelä et. al procedure is that it may have to make too many unnecessary choices as the above example shows. However forward chaining may help in getting closer to the solution more efficiently. The previous observations suggest a modification of the procedure which tries to combine both backward and forward chaining. This can be seen as an integration of ours and Niemelä et. al procedures. In the new procedure, $CSM2$, we make use of some additional procedures and notations:

- Given a set of sentences Γ, Γ^- denotes the set of assumptions contained in Γ.
- A set of sentences Γ is said to be **coherent** if Γ^- is admissible and $\Gamma \subseteq Th(T \cup \Gamma)$,
- Given a set of sentences Γ, $expand(\Gamma)$ defines a forward expansion of Γ satisfying the following conditions:
  1. $\Gamma \subseteq expand(\Gamma)$
  2. If Γ is coherent then
     (a) $expand(\Gamma)$ is also coherent, and
     (b) for each stable extension E, if $\Gamma^- \subseteq E$ then $expand(\Gamma)^- \subseteq E$.

Proof procedure 5.2 (Credulous Stable Models).

$CSM2(\alpha)$:
    $\Delta := CPE(\alpha)$;
    $\Gamma := expand(\Delta)$;
    loop
        if $full\_cover(\Gamma)$
            then **return** Γ^-
            else
                $\beta := uncovered(\Gamma)$;
                $\Delta := adm\_expand(\Gamma^- \cup \{\beta\})$;
                $\Gamma := expand(\Delta \cup \Gamma)$;
        end if
    end loop

As anticipated, the procedure *expand* can be defined in various ways. If *expand* is simply the identity function, i.e. $expand(\Delta) = \Delta$ the procedure $CSM2$ collapses down to CSM. In some other cases, *expand* could also effectively perform forward reasoning, and try to produce the deductive closure of the given set of sentences. This can be achieved by defining *expand* in such a way that

$$expand(\Delta) = Th(T \cup \Delta).$$

In still other cases, $expand(\Delta)$ could be extended to be closed under the Fitting's operator.

As in the case of Theorem 2, we need to assume that the set of assumptions in the underlying framework is finite, in order to prevent non termination of the main loop.

Theorem 3 (Soundness and Completeness of $CSM2$).
Let $\langle T, Ab, \overline{} \rangle$ be a framework such that Ab is finite.

1. *If $CSM2(\alpha)$ succeeds then there exists a stable extension Δ such that $\alpha \in Th(T \cup \Delta)$.*
2. *If both CPE and adm_expand are complete then for each stable extension E such that $\alpha \in Th(T \cup E)$ there exist appropriate selections such that $CSM2(\alpha)$ returns E.*

Proof.

1. We first prove by induction that at the beginning of each iteration of the loop, Γ is coherent. The basic step is clear since Δ is admissible.
   Inductive Step: Let Γ be coherent. From $\Delta := \mathit{adm\_expand}(\Gamma^- \cup \{\beta\})$, it follows that Δ is admissible. Because $\Gamma^- \subseteq \Delta$ and $\Gamma \subseteq Th(T \cup \Gamma)$, it follows that $\Gamma \subseteq Th(T \cup \Delta)$. From $(\Delta \cup \Gamma)^- = \Delta$, it follows that $\Delta \cup \Gamma$ is coherent. Therefore $\mathit{expand}(\Delta \cup \Gamma)$ is coherent.
   It is obvious that for any coherent set of sentences Γ such that $full\_cover(\Gamma)$ holds, Γ^- is stable.
2. Let E be a stable model such that $\alpha \in Th(T \cup E)$. Because CPE is complete, there is a selection such that executing the command $\Delta := CPE(\alpha)$ yields an admissible $\Delta \subseteq E$. From the properties of expand, it follows that Γ obtained from $\Gamma := \mathit{expand}(\Delta)$, is coherent and $\Gamma^- \subseteq E$. If $full\_cover(\Gamma)$ does not hold, then we can always select a $\beta \in E - \Gamma^-$. Therefore due to the completeness of *adm_expand*, we can get a Δ that is a subset of E. Hence Γ obtained from $\Gamma := \mathit{expand}(\Delta \cup \Gamma)$, is coherent and $\Gamma^- \subseteq E$. Continuing this process until termination, which is guaranteed by the hypothesis that *Ab* is finite, will return E as the result of the procedure. □

However, if in the underlying framework every preferred extension is also stable, then CSM can be greatly simplified by dropping the main loop, namely CSM coincides with CPE. As shown in [1], this is the case if the underlying framework is order-consistent (see Appendix).

Theorem 4 (Soundness and completeness of CPE wrt stable models and order consistency).
Let the underlying framework be order-consistent.

1. *If $CPE(\alpha)$ succeeds then there exists a stable extension Δ such that $\alpha \in Th(T \cup \Delta)$.*
2. *If both support and adm_expand are complete then for each stable extension E there exist appropriate selections such that $CPE(\alpha)$ returns $\Delta \subseteq E$.*

The use of CPE instead of CSM, whenever possible, greatly simplifies the task of performing credulous reasoning under the stable semantics, in that it allows to keep the search for a stable extension "localised", as illustrated by the following example.

Example 4. Consider the following order-consistent logic program

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

which has two preferred (and stable) extensions containing p, corresponding to the sets of assumptions $\Delta_1 = \{not\ s, not\ r\}$ and $\Delta_2 = \{not\ s, not\ q\}$. The procedure $CPE(p)$ would compute the admissible extension $\{not\ s\}$ as a result, since $\{not\ s\}$ is a support for p and it is admissible (there are no attacks against $not\ s$) . On the other hand, the procedure $CSM(p)$ would produce either Δ_1 or Δ_2, which are both stable sets extending $\{not\ s\}$.

6 Computing Sceptical Consequences under Stable Extensions

First, we define the notion of "contrary of sentences", by extending the notion of "contrary of assumptions". In all concrete instances of the abstract framework, e.g. logic programming, default logic, autoepistemic logic and non-monotonic modal logic, for each non-assumption sentence β there is a unique assumption α such that $\overline{\alpha} = \beta$, so the natural way of defining the "contrary of a sentence" β which is not an assumption is

$\overline{\beta} = \alpha$ such that $\overline{\alpha} = \beta$.

But in general, it is possible that for some non-assumption sentence β there may be no assumption α such that $\overline{\alpha} = \beta$, or there may be more than one assumption α such that $\overline{\alpha} = \beta$.

Thus, for general frameworks, we define the concept of **contrary of sentences** which are not assumptions as follows. Let β be a sentence such that $\beta \notin Ab$.

- if there exists α such that $\overline{\alpha} = \beta$ then $\overline{\beta} = \{\gamma | \overline{\gamma} = \beta\}$
- if there exists no α such that $\overline{\alpha} = \beta$ then we introduce a new assumption κ_β, not already in the language, and we define
  - $\overline{\kappa_\beta} = \beta$
  - $\overline{\beta} = \{\kappa_\beta\}$

Note that, in this way, the contrary of a sentence $\beta \notin Ab$ is a *set* of assumptions.

Let us denote by $Ab' \supseteq Ab$ the new set of assumptions. It is easy to see that the original framework, $\langle T, Ab, \overline{} \rangle$, and the extended framework, $\langle T, Ab', \overline{} \rangle$, are equivalent in the following sense:

- if $\Delta \subseteq Ab$ is admissible wrt the original framework then it is also admissible wrt the new framework;
- if $\Delta' \subseteq Ab'$ is admissible wrt the new framework then $\Delta' \cap Ab$ is admissible wrt the original framework.

Therefore from now on, we will assume that for each sentence β which is not an assumption there exists at least an assumption α such that $\overline{\alpha} = \beta$.

In order to show that a sentence β is entailed by each stable model, we can proceed as follows:

- check that β is a credulous consequence under the stable model semantics

– check that the contrary of the sentence is not a credulous consequence under the stable models semantics.

Notice that if $\beta \notin Ab$ the second step amounts to checking that each $\alpha \in \overline{\beta}$ is not a credulous consequence under the stable models semantics.

Moreover, notice that the first step of the computation cannot be omitted (as one could expect) since there may be cases in which neither β nor its contrary hold in any stable model (e.g. in the framework corresponding to the logic program $p \leftarrow not\ p$).

Lemma 1. *Let E be a stable extension. Then for each non-assumption β such that $\beta \notin Th(T \cup E)$, the following statements are equivalent:*

1. $\overline{\beta} \cap E \neq \emptyset$
2. $\overline{\beta} \subseteq E$

Proof. It is clear that the second condition implies the first. We need only to prove now that the first condition implies the second one. Let $\overline{\beta} \cap E \neq \emptyset$. Suppose that $\overline{\beta} - E \neq \emptyset$. Let $\alpha \in \overline{\beta} - E$. Then it is clear that $\overline{\alpha} \in Th(T \cup E)$. Contradiction to the condition that $\overline{\alpha} = \beta$ and $\beta \notin Th(T \cup E)$. □

Proof procedure 6.1 (Sceptical Stable Models).

$SSM(\alpha)$:
  if $CSM(\alpha)$ fails **then** fail;
  select $\beta \in \overline{\alpha}$;
  if $CSM(\beta)$ succeeds **then** fail;

Notice that the SSM procedure makes use of the CSM procedure. To prevent non termination of CSM we need to assume that the set of assumptions Ab' of the underlying extended framework is finite. This guarantees the completeness of CSM (cfr. Theorem 2).

Theorem 5 (Soundness and Completeness of SSM). *Let CSM be complete.*

1. *If $SSM(\alpha)$ succeeds then $\alpha \in Th(T \cup \Delta)$, for every stable extension Δ.*
2. *If $\alpha \in Th(T \cup \Delta)$, for every stable extension Δ, and the set of stable extensions is not empty, then $SSM(\alpha)$ succeeds.*

Proof.

1. Let $SSM(\alpha)$ succeed. Assume now that α is not a skeptical consequence wrt stable semantics. There are two cases: $\alpha \in Ab$ and $\alpha \notin Ab$.
   Consider the first case where $\alpha \in Ab$. It follows that there is a stable extension E such that $\overline{\alpha} \in Th(T \cup E)$. Because of the completeness of CSM, it follows that $CSM(\overline{\alpha})$ succeeds. Hence $SSM(\overline{\alpha})$ fails, contradiction.
   Let $\alpha \notin Ab$. From lemma 1, it follows that there is a stable extension E such that $E \cap \overline{\alpha} \neq \emptyset$. That means $CSM(\beta)$ succeeds for some $\beta \in \overline{\alpha}$. Lemma 1 implies $CSM(\beta)$ succeeds for each $\beta \in \overline{\alpha}$. Hence $SMM(\alpha)$ fails. Contradiction.

2. Because CSM is complete, it is clear that $CSM(\alpha)$ succeeds. Also because of the soundness of CSM, $CSM(\beta)$ fails for each $\beta \in \overline{\alpha}$. Therefore it is obvious that SSM succeeds. $\square$

For a large class of argumentation frameworks, preferred extensions and stable models semantics coincide, e.g. if the frameworks are order-consistent [1]. In these frameworks, the procedure SSM can be simplified significantly as follows.

Proof procedure 6.2 (Sceptical Stable Models via CPE).

$SSMPE(\alpha)$:
  if $CPE(\alpha)$ fails **then** fail;
  select $\beta \in \overline{\alpha}$;
  if $CPE(\beta)$ succeeds **then** fail ;

The procedure is structurally the same as the earlier SSM, but it relies upon CPE rather than CSM, and is therefore "simpler" in the same way that CPE is "simpler" than CSM, as discussed earlier in Sect. 5.

Theorem 6 (Soundness and completeness of $SSMPE$ wrt sceptical stable semantics). *Let the underlying framework be order-consistent and CPE be complete.*

1. *If $SSMPE(\alpha)$ succeeds then $\alpha \in Th(T \cup \Delta)$, for every stable extension Δ.*
2. *If $\alpha \in Th(T \cup \Delta)$, for every stable extension Δ, then $SSMPE(\alpha)$ succeeds.*

Note that the second statement in the above theorem does not require the existence of stable extensions. This is due to the assumption that order-consistency always guarantees such condition.

7 Computing Sceptical Consequences under Preferred Extensions

The naive way of showing that a sentence is a sceptical consequence under the preferred extensions semantics is to consider each preferred extension in turn and check that the sentence is entailed by it.

The earlier procedure $SSMPE$ can be used as a simplification of the naive method only if every preferred extension is guaranteed to be stable. In general, however, the procedure $SSMPE$ is not sound under the preferred extensions semantics, since there might exist preferred extensions in which, for some assumption α, neither α nor its contrary hold, as the following example shows.

Example 5.

$$\begin{array}{l} p \leftarrow not\ p \\ p \leftarrow q \\ q \leftarrow not\ r \\ r \leftarrow not\ q \end{array}$$

Notice that there are two preferred extensions,namely $E_1 = \{not\ q, r\}$ and $E_2 = \{not\ r, q, p\}$. E_2 is also a stable extension, whereas E_1 is not since neither p nor *not* p hold in E_1. Notice that $SSMPE(p)$ would succeed, hence giving an unsound result.

Nonetheless, in the general case, the following theorem shows that it is possible to restrict the number of preferred extensions to consider. This theorem is a variant of theorem 16 in [30], as we will discuss in Sect. 8.

Theorem 7. *Given an argumentation-theoretic framework $\langle T, Ab, \overline{} \rangle$ and a sentence α in its language, α is a sceptical non-monotonic consequence of T with respect to the preferred extension semantics, i.e. $\alpha \in Th(T \cup \Delta)$ for all preferred $\Delta \subseteq Ab$, iff*

1. *$\alpha \in Th(T \cup \Delta_0)$, for some admissible set of assumptions $\Delta_0 \subseteq Ab$, and*
2. *for every set of assumptions $\Delta \subseteq Ab$,*
   if Δ is admissible and Δ attacks Δ_0,
   then $\alpha \in Th(T \cup \Delta')$ for some set of assumptions $\Delta' \subseteq Ab$ such that
   (a) $\Delta' \supseteq \Delta$, and
   (b) Δ' is admissible.

Proof. The only if half is trivial.
The if half is proved by contradiction. Suppose there exists a set of assumptions Δ^* such that Δ^* is preferred and $\alpha \notin Th(T \cup \Delta^*)$. Suppose Δ_0 is the set of assumptions provided in part 1.
If $\Delta_0 = \emptyset$ then $\alpha \in Th(T)$ and therefore $\alpha \in Th(T \cup \Delta^*)$, thus contradicting the hypothesis.
Therefore, $\Delta_0 \neq \emptyset$. Consider the following two cases:

(i) $\Delta^* \cup \Delta_0$ attacks itself, or
(ii) $\Delta^* \cup \Delta_0$ does not attack itself.

Case (ii) implies that $\Delta^* \cup \Delta_0$ is admissible, thus contradicting the hypothesis that Δ^* is preferred (and therefore maximally admissible).
Case (i) implies that

(i.1) $\Delta^* \cup \Delta_0$ attacks Δ^*, or
(i.2) $\Delta^* \cup \Delta_0$ attacks Δ_0.

Assume that (i.1) holds. .
$\Delta^* \cup \Delta_0$ attacks Δ^*
$\Rightarrow$ {by admissibility of Δ^*}
Δ^* attacks $\Delta^* \cup \Delta_0$
$\Rightarrow$ {by admissibility, Δ^* does not attack itself}
Δ^* attacks Δ_0
$\Rightarrow$ {by part 2 }
$\alpha \in Th(T \cup \Delta^*)$
thus contradicting the hypothesis.

Assume now that (i.2) holds.
$\Delta^* \cup \Delta_0$ attacks Δ_0
$\Rightarrow$ {by admissibility of Δ_0}
Δ_0 attacks $\Delta^* \cup \Delta_0$
$\Rightarrow$ {by admissibility, Δ_0 does not attack itself}
Δ_0 attacks Δ^*
$\Rightarrow$ {by admissibility of Δ^*}
Δ^* attacks Δ_0
$\Rightarrow$ {by part 2 }
$\alpha \in Th(T \cup \Delta^*)$
thus contradicting the hypothesis. $\square$

This result can be used to define the following procedure to check whether or not a given sentence is a sceptical consequence with respect to the preferred extension semantics.

Let us assume the following procedure is defined

- $attacks(\Delta)$ computes a base of the set of all attacks against the set of assumptions Δ.

Proof procedure 7.1 (Sceptical Preferred Extensions).

$SPE(\alpha)$:
  $\Delta := CPE(\alpha)$;
  for each $A := attacks(\Delta)$
    for each $\Delta' := adm\_expand(A)$
      $\Delta'' := support(\alpha, \Delta')$;
      if $adm\_expand(\Delta'')$ fails **then** fail **end if**
    end for
  end for

The following soundness theorem is a trivial corollary of theorem 7.

Theorem 8 (Soundness and Completeness of SPE). *Let adm_expand be complete.*

1. *if $SPE(\alpha)$ succeeds, then $\alpha \in Th(T \cup \Delta)$, for every preferred extension Δ.*
2. *If CPE is complete and $\alpha \in Th(T \cup \Delta)$, for every preferred extension Δ, then $SPE(\alpha)$ succeeds.*

In many cases where the framework has exactly one preferred extension that is also stable (for example when the framework is stratified), it is obvious that the CPE procedure could be used as a procedure for skeptical preferred extension semantics.

8 Related Work

The proof procedures we propose in this paper rely upon proof procedures for computing credulous consequences under the semantics of admissible extensions. A number of such procedures have been proposed in the literature.

Eshghi and Kowalski [9] (see also the revised version proposed by Dung in [5]) propose a proof procedure for logic programming based upon interleaving abductive derivations, for the generation of negative literals to "derive" goals, and consistency derivations, to check "consistency" of negative literals with atoms "derivable" from the program. The proof procedure can be understood in argumentation-theoretic terms [12], as interleaving the generation of assumptions supporting goals or counter-attacking assumptions (abductive derivations) and the generation of attacks against any admissible support (consistency derivations), while checking that the generated support does not attack itself.

Dung, Kowalski and Toni [7] propose abstract proof procedures for computing credulous consequences under the semantics of admissible extensions, defined via logic programs.

Kakas and Toni [15] propose a number of proof procedures based on the construction of trees whose nodes are sets of assumptions, and such that nodes attack their parents, if any. The proof procedures are defined in abstract terms and, similarly to the procedures we propose in this paper, can be adopted for any concrete framework that is an instance of the abstract one. The procedures allow to compute credulous consequences under the semantics of admissible extensions as well as under semantics that we have not considered in this paper, namely the semantics of weakly stable extensions, acceptable extensions, well-founded extensions. The concrete procedure for computing credulous consequences under the semantics of admissible extensions, in the case of logic programming, corresponds to the proof procedure of [9].

Dung, Kowalski and Toni [8] also propose abstract proof procedures for computing credulous consequences under the semantics of admissible extensions, that can be instantiated to any instance of the framework of [1]. These procedures are defined in terms of trees whose nodes are assumptions, as well as via derivations as in [9].

Kakas and Dimopoulos [2] propose a proof procedure to compute credulous consequences under the semantics of admissible extensions for the argumentation framework of Logic Programming without Negation as Failure proposed in [14]. Here, negation as failure is replaced and extended by priorities over logic programs with no negation as failure but with explicit negation instead.

Other proof procedures for computing credulous consequences under the stable extension semantics and sceptical consequences under the semantics of preferred and stable extensions have been proposed.

Thielscher [30] proposes a proof procedure for computing sceptical consequences under the semantics of preferred extensions for the special case of logic programming [31]. This proof procedure is based upon a version of theorem 7 (theorem 16 in [30]). However, whereas [30] uses the notion of "conflict-free set of arguments" (which is an atomic, abstract notion), we use the notion of admissible set of assumptions. Moreover, theorem 16 in [30] replaces the condition in part 2 of theorem 7 "Δ' attacks Δ_0" by the (equivalent) condition corresponding to "$\Delta' \cup \Delta_0$ attacks itself". For a formal correspondence between the two approaches see [31].

Niemelä [19] and Niemelä and Simons [20] give proof procedures for computing credulous and sceptical consequences under stable extensions, for default logic and logic programming, respectively. As discussed in Sect. 5, their proof procedures for computing credulous consequences under stable extensions rely upon forward chaining, whereas the proof procedures we propose for the same task rely either on backward chaining (CSM) or on a combination of backward and forward chaining (CSM2).

Satoh and Iwayama [28] define a proof procedure for logic programming, computing credulous consequences under the stable extension semantics for range-restricted logic programs that admit at least one stable extension. Satoh [27] adapts the proof procedure in [28] to default logic. The proof procedure applies to consistent and propositional default theories.

Inoue et al. [11] apply the model generation theorem prover to logic programming to generate stable extensions, thus allowing to perform credulous reasoning under the stable extension semantics by forward chaining.

9 Conclusions

We have presented abstract proof procedures for computing credulous and sceptical consequences under the semantics of preferred and stable extensions for non-monotonic reasoning, as proposed in [1], relying upon any proof procedure for computing credulous consequences under the semantics of admissible extensions.

The proposed proof procedures are abstract in that they can be instantiated to any concrete framework for non-monotonic reasoning which is an instance of the abstract flat framework of [1]. These include logic programming and default logic. They are abstract also in that they abstract away from implementation details.

We have compared our proof procedures with existing, state of the art procedures defined for logic programming and default logic.

We have argued that the proof procedures for computing consequences under the semantics of preferred extensions are simpler than those for computing consequences under the semantics of stable extensions, and supported our arguments with examples. However, note that the (worst-case) computational complexity of the problem of computing consequences under the semantics of stable extensions is in general no worse than that of computing consequences under the semantics of preferred extensions, and in some cases it is considerably simpler [3, 4]. In particular, in the case of autoepistemic logic, the problem of computing sceptical consequences under the semantics of preferred extensions is located at the fourth level of the polynomial hierarchy, whereas the same problem under the semantics of stable extensions is located at the second level.

Of course, these results do not contradict the expectation that in practice, in many cases, computing consequences under the semantics of preferred extensions is easier than under the semantics of stable extensions. Indeed, preferred extensions supporting a desired sentence can be constructed "locally", by restricting attention to the sentences in the language that are directly relevant to the sentence. Instead, stable extensions need to be constructed "globally", by considering all sentences in the language, whether they are directly relevant to the given sentence or not. This is due to the fact that stable extensions are not guaranteed to exist.

However, note that in all cases where stable extensions are guaranteed to exist and coincide with preferred extensions, e.g. for stratified and order-consistent frameworks [1], any proof procedure for reasoning under the latter is a correct (and simpler) computational mechanism for reasoning under the former.

Finally, the "locality" feature in the computation of consequences under the preferred extension semantics renders it a feasible alternative to the computation of consequences under the stable extension semantics in the non-propositional case, when the language is infinite. Indeed, both CPE and SPE do not require that the given framework be propositional.

Acknowledgements

This research has been partially supported by the EC KIT project "Computational Logic for Flexible Solutions to Applications". The third author has been supported by the UK EPSRC project "Logic-based multi-agent systems".

References

1. A. Bondarenko, P. M. Dung, R. A. Kowalski, F. Toni, An abstract, argumentation-theoretic framework for default reasoning. *Artificial Intelligence*, 93:63-101, 1997.
2. Y. Dimopoulos, A. C. Kakas, Logic Programming without Negation as Failure, Proceedings of the 1995 International Symposium on Logic Programming, pp. 369-383, 1995.
3. Y. Dimopoulos, B. Nebel, F. Toni, Preferred Arguments are Harder to Compute than Stable Extensions, Proc. of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI 99, (T. Dean ed.), pp. 36-43, 1999.
4. Y. Dimopoulos, B. Nebel, F. Toni, Finding Admissible and Preferred Arguments Can Be Very Hard, Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning, KR 2000, (A. G. Cohn, F. Giunchiglia, B. Selman eds.), pp. 53-61, Morgan Kaufmann Publishers, 2000.
5. P. M. Dung, Negation as hypothesis: an abductive foundation for logic programming. *Proceedings of the 8th International Conference on Logic Programming*, Paris, France (K. Furukawa, ed.), MIT Press, pp. 3–17, 1991.
6. P. M. Dung, On the acceptability of arguments and its fundamental role in non-monotonic reasoning, logic programming and n-person games *Artificial Intelligence,*, 77:321-357, Elsevier, 1993.

7. P. M. Dung, R. A. Kowalski, F. Toni, Synthesis of proof procedures for default reasoning, *Proc. LOPSTR'96, International Workshop on Logic Program Synthesis and Transformation*, (J. Gallagher ed.), pp. 313–324, LNCS 1207, Springer Verlag, 1996.
8. P. M. Dung, R. A. Kowalski, F. Toni, Proof procedures for default reasoning. In preparation, 2002.
9. K. Eshghi, R. A. Kowalski, Abduction compared with negation as failure. *Proceedings of the 6th International Conference on Logic Programming*, Lisbon, Portugal (G. Levi and M. Martelli, eds), MIT Press, pp. 234–254, 1989
10. M. Gelfond, V. Lifschitz, The stable model semantics for logic programming. *Proceedings of the 5th International Conference on Logic Programming*, Washington, Seattle (K. Bowen and R. A. Kowalski, eds), MIT Press, pp. 1070–1080, 1988
11. K. Inoue, M. Koshimura, R. Hasegawa, Embedding negation as failure into a model generation theorem-prover. Proc. *CADE'92*, pp. 400-415, LNCS 607, Springer, 1992.
12. A. C. Kakas, R. A. Kowalski, F. Toni, The role of abduction in logic programming. *Handbook of Logic in Artificial Intelligence and Logic Programming* (D.M. Gabbay, C.J. Hogger and J.A. Robinson eds.), 5: 235-324, , Oxford University Press, 1998.
13. A. C. Kakas, P. Mancarella. Preferred extensions are partial stable models. *Journal of Logic Programming* 14(3,4), pp.341–348, 1993.
14. A. C. Kakas, P. Mancarella, P. M. Dung, The Acceptability Semantics for Logic Programs, Proceedings of the Eleventh International Conference on Logic Programming, pp. 504-519, 1994.
15. A. C. Kakas, F. Toni, Computing Argumentation in Logic Programming. *Journal of Logic and Computation* 9:515-562, Oxford University Press, 1999.
16. J. McCarthy, Circumscription – a form of non-monotonic reasoning. *Artificial Intelligence*, 1327–39, 1980.
17. D. McDermott, Nonmonotonic logic II: non-monotonic modal theories. *Journal of ACM* 29(1), pp. 33–57, 1982.
18. R. Moore, Semantical considerations on non-monotonic logic. *Artificial Intelligence* 25:75–94, 1985.
19. I. Niemelä, Towards efficient default reasoning. *Proc. IJCAI'95*, pp. 312–318, Morgan Kaufman, 1995.
20. I. Niemelä, P. Simons, Efficient implementation of the well-founded and stable model semantics. *Proc. JICSLP'96*, pp. 289–303, MIT Press, 1996.
21. J. L. Pollock. Defeasible reasoning. *Cognitive Science*, 11(4):481–518, 1987.
22. D. Poole, A logical framework for default reasoning. *Artificial Intelligence* 36:27–47, 1988.
23. H. Prakken and G. Sartor. A system for defeasible argumentation, with defeasible priorities. *Artificial Intelligence Today*, (M. Wooldridge and M. M. Veloso, eds.), LNCS 1600, pp. 365–379, Springer, 1999.
24. H. Prakken and G. Vreeswijk. Logical systems for defeasible argumentation. *Handbook of Philosophical Logic*, 2nd edition, (D. Gabbay and F. Guenthner eds.), Vol. 4, Kluwer Academic Publishers, 2001.
25. R. Reiter, A logic for default reasoning. *Artificial Intelligence* 13:81–132, Elsevier, 1980).
26. 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*, ACM Press, pp. 205–217, 1990.

27. K. Satoh, A top-down proof procedure for default logic by using abduction. *Proceedings of the Eleventh European Conference on Artificial Intelligence*, pp. 65-69, John Wiley and Sons, 1994.
28. K. Satoh and N. Iwayama. A Query Evaluation Method for Abductive Logic Programming. Proceedings of the Joint International Conference and Symposium on Logic Programming, pp. 671 – 685, 1992.
29. G.R. Simari and R.P. Loui. A mathematical treatment of defeasible reasoning and its implementation. *Artificial Intelligence*, 52:125–257, 1992.
30. M. Thielscher, A nonmonotonic disputation-based semantics and proof procedure for logic programs. *Proceedings of the 1996 Joint International Conference and Symposium on Logic Programming* (M. Maher ed.), pp. 483–497, 1996.
31. F. Toni, Argumentation-theoretic proof procedures for logic programming. Technical Report, Department of Computing, Imperial College, 1997.
32. G. Vreeswijk. The feasibility of defeat in defeasible reasoning. *Proceedings of the 2nd Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'91)*, (J.F. Allen, R. Fikes, E. Sandewall, eds.), pp. 526–534, 1991.

A Stratified and order consistent frameworks

We recall the definitions of stratified and order consistent flat argumentation-theoretic frameworks, and theire semantics properties, ad given in [1]. Both classes are characterized in terms of their attack relationship graphs.

The *attack relationship graph* of a flat assumption-based framework $\langle T,\ Ab,\ \overline{}\rangle$ is a directed graph whose nodes are the assumptions in Ab and such that there exists an edge from an assumption δ to an assumption α if and only if δ belongs to a minimal (with respect to set inclusion) attack Δ against α.

A flat assumption-based framework is *stratified* if and only if its attack relationship graph is well-founded, i.e. it contains no infinite path of the form $\alpha_1, \ldots, \alpha_n, \ldots$, where for every $i \geq 0$ there is an edge from α_{i+1} to α_i.

The notion of order-consistency requires some more auxiliary definitions. Given a flat assumption-based framework $\langle T,\ Ab,\ \overline{}\rangle$ let $\delta, \alpha \in Ab$.

- δ is *friendly* (resp. *hostile*) to α if and only if the attack relationship graph for $\langle T,\ Ab,\ \overline{}\rangle$ contains a path from δ to α with an even (resp. odd) number of edges.
- δ is *two-sided* to α, written $\delta \prec \alpha$, if δ is both friendly and hostile to α.

A flat assumption-based framework $\langle T,\ Ab,\ \overline{}\rangle$ is *order-consistent* if the relation $\prec$ is well-founded, i.e. there exists no infinite sequence of the form $\alpha_1, \ldots, \alpha_n, \ldots$, where for every $i \geq 0$, $\alpha_{i+1} \prec \alpha_i$.

The following proposition summarizes some of the semantics results of [1] as far as stratified and order-consistent frameworks are concerned.

Proposition 1 (see [1]).

- *for any stratified assumption-based framework there exists a unique stable set of assumptions, which coincides with the well-founded set of assumptions.*
- *for any order-consistent assumption-based framework stable sets of assumptions are preferred sets of assumptions and viceversa.*

It is worth recalling that the abstract notions of stratification and order-consistency generalize the notions of stratification and order-consistency for logic programming.