Semi-stable semantics

MARTIN W. A. CAMINADA, *Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg, Luxembourg.*
E-mail: martin.caminada@uni.lu

WALTER A. CARNIELLI, *Centre for Logic, Epistemology and the History of Science and Department of Philosophy, Unicamp, Brazil.*
E-mail: walter.carnielli@cle.unicamp.br

PAUL E. DUNNE, *Department of Computer Science, University of Liverpool, UK.*
E-mail: ped@csc.liv.ac.uk

Abstract

In this article, we examine an argument-based semantics called *semi-stable semantics*. Semi-stable semantics is quite close to traditional stable semantics in the sense that every stable extension is also a semi-stable extension. One of the advantages of semi-stable semantics is that for finite argumentation frameworks there always exists at least one semi-stable extension. Furthermore, if there also exists at least one stable extension, then the semi-stable extensions coincide with the stable extensions. Semi-stable semantics can be seen as a general approach that can be applied to abstract argumentation, as well as to fields like default logic and answer set programming, yielding an interpretation with properties very similar to those of paraconsistent logic, including the properties of *crash resistance* and *backward compatibility*.

Keywords: Computational argumentation, default logic, logic programming, nonmonotonic reasoning, paraconsistent reasoning.

1 Introduction

The stable model semantics, a concept that goes back to [52], has been applied in fields like formal argumentation [26], default reasoning [48], (normal) logic programming [33] and answer set programming [34]. During the last two decades, various different semantics have been stated as alternatives for stable semantics, such as regular or preferred semantics [26, 56] or well-founded or grounded semantics [26, 50]. Although some of those semantics, e.g. grounded and preferred, have become popular in the domain of formal argumentation, stable semantics still enjoys strong support in fields like (normal) logic programming and answer set programming, where various interpretations of the stable model semantics exist [37, 40].

The varying levels of support for either stable semantics or its alternatives can to some extent be explained by the nature of the application domain. In the field of formal argumentation the emphasis is often on how to combine various principles and rules of thumb, so as to yield an overall coherent outcome. That is, one would like to have the most reasonable outcome in the presence of defeasible and possibly conflicting information. In this context stable semantics, with its fundamental property that relatively small difficulties in the input-data can cause the total absence of stable extensions,

Vol. 22 No. 5,
Published online September 14, 2011 doi:10.1093/logcom/exr033

may not be an attractive option. This has led most of the argumentation research to shift its focus on other semantics, of which grounded and preferred are among the most well known.

The situation is quite different in, for instance, the field of (normal) logic programming and answer set programming. Here, the emphasis is often on computable forms of constraint satisfaction. If one applies answer set programming to solve a particular constraint satisfaction problem, then one wishes the models of the answer set program to correspond to the solutions of the original problem. Consequently, if the original problem does not have any solutions, then one also wants to obtain no answer sets. The fact that stable semantics sometimes yields no extensions could in this respect even be seen as an advantage rather than a weakness.

One of the assumptions underlying the stable model semantics, however, is that the original problem has been modelled in a way that is fully complete and correct. For if it is not, then the result can be no outcome instead of a merely imperfect outcome. In essence, the situation is not that different from classical logic, where syntactically small imperfections in the consistency of the input-data can lead to an overall collapse of all entailment.

To deal with the possible collapse of classical logic entailment in the presence of imperfect information, various forms of paraconsistent logics have been proposed. The idea here is that relatively small problems in the original specification should no longer lead to a global 'collapse' of all entailment. That is, the formalism should be what we call 'crash resistant'. Furthermore, one would like to have the same outcome as classical logic in situations where the input-specification is free from flaws that lead to such collapses: this requirement is referred to as 'backward compatibility'.

In this article, we explore the notion of paraconsistency as a guide in the context of formalisms that use stable semantics, such as default logic and answer set programming. We are interested in ways that make them tolerant for flaws in their respective input-specifications ('crash resistant'), yet at the same time yield the same outcome as under stable semantics in cases where stable extensions do exist ('backward compatibility'). We provide a general solution, called *semi-stable semantics*, with which one can obtain these properties. Not only do we specify the abstract solution, but we also illustrate how it can be applied in the domains of abstract argumentation, default reasoning and logic programming. Moreover, we also provide a complexity analysis for our solution.

This article is structured as follows. First, in Section 2 we provide a formal account of what we see as a number of desirable properties regarding logical formalisms. Our treatment builds on [15], where a number of *rationality postulates* were given. Our current article, however, takes a more general approach and no longer assumes the input-specification to be of a particular form or syntax. Then, in Section 3, we present our approach of semi-stable semantics and show how it satisfies the requirements outlined in Section 2. We have chosen initially to specify our approach using the framework of *formal argumentation* [26], which has gained popularity as a general way of describing various forms of non-monotonic reasoning. In Section 4, we describe how the approach of semi-stable semantics can be implemented in the context of logic programming, and that the result satisfies the earlier mentioned desirable properties. In Section 5, we do the same for default logic. In Section 6, we give a complexity analysis of various decision problems related to our new approach. In Section 7, we discuss how semi-stable semantics relates to other approaches. In Section 8, we round off the discussion with a few concluding issues and remarks.

The concept of semi-stable semantics was first presented at COMMA 2006 [11] and its computational complexity at JELIA 2008 [30]. What is new within the current article, however, is that we explain the applicability of semi-stable semantics by specifying the postulates it satisfies, for abstract argumentation as well as for logic programming and default logic. Furthermore, we discuss an alternative (stage semantics) for satisfying these postulates, and examine its relationship to semi-stable semantics.

2 On Logics, Crashes and Contamination

In this section, we describe a number of desirable properties, inspired by the field of paraconsistent reasoning, which can be defined for any logical formalism. We define the notion of a logical formalism in a very broad way; in particular, we do not specify any properties that define the consequence relationship Cn. Furthermore, in contrast to [15], we make no assumptions regarding the particular syntax of the logical formalism under review.

DEFINITION 1
A *logical formalism* is a triple ($\mathcal{A}$toms, $\mathcal{F}$ormulas, Cn) where $\mathcal{A}$toms is a countably (finite or infinite) set of atoms, $\mathcal{F}$ormulas is the set of all well-formed formulas that can be constructed using $\mathcal{A}$toms, and $Cn : 2^{\mathcal{F}\text{ormulas}} \rightarrow 2^{2^{\mathcal{F}\text{ormulas}}}$ is the consequence function.

Notice that the consequence function takes as input a set of formulas and has as output a set of sets of formulas. This is to accommodate formalisms like default logic (or answer set programming) where a single default theory (answer set program) can generate several extensions (answer sets). For formalisms that generate only one set of formulas (like for instance classical logic), we sometimes abuse notation and write $Cn(\Psi) = \Phi$ instead of $Cn(\Psi) = \{\Phi\}$ (where Ψ and Φ are sets of formulas).

In the following definitions, we write $atoms(\mathcal{F})$ for the atoms that occur in a set of formulas $\mathcal{F}$. For instance: $atoms(\{p \wedge q; r \vee p\}) = \{p, q, r\}$ and $atoms(\{p \leftarrow q; r \leftarrow s, t\}) = \{p, q, r, s, t\}$. Furthermore, if $\mathcal{A}$ is a set of atoms and $\mathcal{F}$ a set of formulas then we write $\mathcal{F}_{|\mathcal{A}}$ for those formulas in $\mathcal{F}$ that contain only atoms from $\mathcal{A}$. For instance: $\{p \wedge q; q \supset r; s \vee t; q\}_{|\{p,q\}} = \{p \wedge q; q\}$. Likewise, if $\mathcal{E}$ is a set of sets of formulas, we define $\mathcal{E}_{|\mathcal{A}}$ as $\{\mathcal{F}_{|\mathcal{A}} \mid \mathcal{F} \in \mathcal{E}\}$. We say that two sets of formulas $\mathcal{F}_1$ and $\mathcal{F}_2$ are syntactically disjoint iff $atoms(\mathcal{F}_1) \cap atoms(\mathcal{F}_2) = \emptyset$.

The first property to be stated is that of *non-interference*. Non-interference roughly means that, for two completely independent knowledge bases $\mathcal{F}_1$ and $\mathcal{F}_2$, $\mathcal{F}_1$ does not influence the outcome with respect to the language of $\mathcal{F}_2$.

DEFINITION 2 (non-interference)
We say that a logical formalism ($\mathcal{A}$toms, $\mathcal{F}$ormulas, Cn) satisfies *non-interference* iff for every $\mathcal{F}_1, \mathcal{F}_2 \subseteq \mathcal{F}$ormulas such that $\mathcal{F}_1$ and $\mathcal{F}_2$ are syntactically disjoint it holds that $Cn(\mathcal{F}_1)_{|\text{atoms}(\mathcal{F}_1)} = Cn(\mathcal{F}_1 \cup \mathcal{F}_2)_{|\text{atoms}(\mathcal{F}_1)}$ and $Cn(\mathcal{F}_2)_{|\text{atoms}(\mathcal{F}_2)} = Cn(\mathcal{F}_1 \cup \mathcal{F}_2)_{|\text{atoms}(\mathcal{F}_2)}$.

A property closely related to non-interference is that of *contamination*. Informally, a set of formulas is said to be contaminating iff it yields the same outcome when merged with a totally unrelated set of formulas. That is, a contaminating set of formulas makes all other unrelated sets of formulas irrelevant when being merged with it.

DEFINITION 3 (contamination)
Let ($\mathcal{A}$toms, $\mathcal{F}$ormulas, Cn) be a logical formalism. A set $\mathcal{F}_1 \subseteq \mathcal{F}$ormulas, with $atoms(\mathcal{F}_1) \subsetneqq \mathcal{A}$toms, is called *contaminating* iff for every $\mathcal{F}_2 \subseteq \mathcal{F}$ormulas such that $\mathcal{F}_1$ and $\mathcal{F}_2$ are syntactically disjoint it holds that $Cn(\mathcal{F}_1) = Cn(\mathcal{F}_1 \cup \mathcal{F}_2)$.

Based on the concept of contamination, it is then possible to define the property of crash resistance.

DEFINITION 4 (crash resistance)
We say that a logical formalism satisfies *crash resistance* iff there does not exist a set of formulas $\mathcal{F}$ that is contaminating.

The property of crash resistance is perhaps best understood by making an analogy. As any experienced computer user knows, it sometimes can occur that a program misbehaves. Under some operating systems, however, the fact that one program misbehaves (like executing illegal instructions) causes the entire operating system to collapse, which then also has consequences for all other programs that were running, even if they are totally unrelated to the program that caused the original problem. The main point here is that one wants to avoid local problems having global effects, and rendering all other things irrelevant. It is this property that is expressed in the above definition of crash resistance.

We say that a logical formalism is *non-trivial* if, first of all, the entailment is never fully determined by the atoms alone. That is, it should be possible for two sets of formulas, with the same atoms, to have different entailment. This is satisfied by most formalisms we know of. For instance, in classical logic we have that $Cn(\{a \wedge b\}) \neq Cn(\{a \vee b\})$, and in logic programming we have that $Cn(\{a \leftarrow\}) \neq Cn(\{a \leftarrow a\})$. Furthermore, for each set of atoms $\mathcal{A}$, there should be a set of formulas whose atoms are exactly $\mathcal{A}$. Formally, this can be expressed as follows.

DEFINITION 5
We say that a logical formalism ($\mathcal{A}$toms, $\mathcal{F}$ormulas, Cn) is *non-trivial* iff for each $\mathcal{A} \subseteq \mathcal{A}$toms such that $\mathcal{A} \neq \emptyset$ there exist $\mathcal{F}_1, \mathcal{F}_2 \subseteq \mathcal{F}$ormulas such that $atoms(\mathcal{F}_1) = atoms(\mathcal{F}_2) = \mathcal{A}$ and $Cn(\mathcal{F}_1)_{|\mathcal{A}} \neq Cn(\mathcal{F}_2)_{|\mathcal{A}}$.

For any non-trivial formalism, non-interference implies crash resistance.

THEOREM 1
Each non-trivial logical formalism ($\mathcal{A}$toms, $\mathcal{F}$ormulas, Cn) that satisfies non-interference also satisfies crash resistance.

PROOF. We prove this by *modus tollens*. Suppose the logical formalism does not satisfy crash resistance. Then there exists a set of formulas (say, $\mathcal{F}_1$) that is contaminating. That is, it holds that $atoms(\mathcal{F}_1) \subsetneq \mathcal{A}$toms and for every $\mathcal{F}_2 \subseteq \mathcal{F}$ormulas with $atoms(\mathcal{F}_1) \cap atoms(\mathcal{F}_2) = \emptyset$ it holds that $Cn(\mathcal{F}_1) = Cn(\mathcal{F}_1 \cup \mathcal{F}_2)$. Let $\mathcal{A}$ be $\mathcal{A}$toms$\backslash atoms(\mathcal{F}_1)$. From the fact that $atoms(\mathcal{F}_1) \subsetneq \mathcal{A}$toms it follows that $\mathcal{A} \neq \emptyset$. The fact that the formalism is non-trivial implies that there exist $\mathcal{F}_3, \mathcal{F}_4 \subseteq \mathcal{F}$ormulas such that $atoms(\mathcal{F}_3) = atoms(\mathcal{F}_4) = \mathcal{A}$ and $Cn(\mathcal{F}_3)_{|\mathcal{A}} \neq Cn(\mathcal{F}_4)_{|\mathcal{A}}$. From the fact that $\mathcal{F}_1$ is contaminating, it follows that $Cn(\mathcal{F}_1) = Cn(\mathcal{F}_1 \cup \mathcal{F}_3)$ and $Cn(\mathcal{F}_1) = Cn(\mathcal{F}_1 \cup \mathcal{F}_4)$. It then follows that $Cn(\mathcal{F}_1 \cup \mathcal{F}_3)_{|\mathcal{A}} = Cn(\mathcal{F}_1 \cup \mathcal{F}_4)_{|\mathcal{A}}$. This, together with the fact that $Cn(\mathcal{F}_3)_{|\mathcal{A}} \neq Cn(\mathcal{F}_4)_{|\mathcal{A}}$ then implies that $Cn(\mathcal{F}_3)_{|\mathcal{A}} \neq Cn(\mathcal{F}_1 \cup \mathcal{F}_3)_{|\mathcal{A}}$ or $Cn(\mathcal{F}_4)_{|\mathcal{A}} \neq Cn(\mathcal{F}_1 \cup \mathcal{F}_4)_{|\mathcal{A}}$. From the fact that $atoms(\mathcal{F}_3) = atoms(\mathcal{F}_4) = \mathcal{A}$ it then immediately follows that $Cn(\mathcal{F}_3)_{|atoms(\mathcal{F}_3)} \neq Cn(\mathcal{F}_1 \cup \mathcal{F}_3)_{|atoms(\mathcal{F}_3)}$ or $Cn(\mathcal{F}_4)_{|atoms(\mathcal{F}_4)} \neq Cn(\mathcal{F}_1 \cup \mathcal{F}_4)_{|atoms(\mathcal{F}_4)}$. In either case, non-interference is violated. ■

The converse of Theorem 1 does not hold. That is, it is not the case that each non-trivial logical formalism that satisfies crash resistance also satisfies non-interference. As an example, Pollock's OSCAR [43] satisfies crash resistance but for reasons explained in [9] does not satisfy non-interference.

The last property to be discussed is that of backward compatibility. The idea is, roughly, that if a formalism (like paraconsistent logic) is to improve on an existing, possibly non crash resisting formalism (like propositional logic) it should yield the same outcome for all non-contaminating input in the latter formalism.

DEFINITION 6 (backward compatibility)
Let ($\mathcal{A}$toms, $\mathcal{F}$ormulas, Cn_1) and ($\mathcal{A}$toms, $\mathcal{F}$ormulas, Cn_2) be two logical formalisms. We say that ($\mathcal{A}$toms, $\mathcal{F}$ormulas, Cn_2) is *backward compatible* with ($\mathcal{A}$toms, $\mathcal{F}$ormulas, Cn_1) iff for each set of formulas $\mathcal{F}$ that is not contaminating under Cn_1, it holds that $Cn_2(\mathcal{F}) = Cn_1(\mathcal{F})$.

As an example of how the above postulates can be applied, consider the case of classical logic. Classical logic does not satisfy non-interference (Definition 2) since an inconsistent set of formulas (say, $\{p, \neg p\}$) interferes with any consistent set of formulas (say, $\{q\}$). Also, any set of inconsistent formulas is contaminating in the sense of Definition 3. Hence, classical logic is not crash resistant (Definition 4). The issue of how to define an alternative form of entailment that satisfies non-interference and crash resistance has been studied in the field of paraconsistent logic. The first generation of paraconsistent logics, such as [22], did satisfy non-interference and crash resistance, but was not backward compatible with respect to classical logic. That is, even in situations that were classically consistent, the new paraconsistent formalism could yield different outcomes than classical logic. Other formalisms, such as those of Arieli and Avron [3] and of Carnielli *et al*. [18], do satisfy non-interference and crash resistance but also remain backward compatible with classical logic.

One can apply the same postulates to non-monotonic formalisms like default logic and answer set programming. These formalisms do not satisfy non-interference (Definition 2) since a default like $\texttt{true}:\neg p/p$ or a rule like $p \leftarrow \texttt{not}\, p$ can easily cause the absence of any default extensions or answer sets. This default (and rule) is also contaminating in the sense of Definition 3. Hence, default logic and answer set programming are not crash resistant (Definition 4). It can be mentioned that several alternative semantics for default logic and logic programming have been specified, such as [8, 50, 56]. None of these alternatives, however, is backward compatible with the original formalisms of default logic and answer set programming.[1] That is, even in situations where default extensions or answer sets do exist, the alternative formalisms can yield different outcomes. The interesting question, therefore, is whether one can find a general principle that can be used to modify a wide range of formalisms, including abstract argumentation under stable semantics, default logic and answer set programming, such that the result satisfies non-interference and crash resistance and at the same time backward compatibility with respect to the original formalism. Hence, the general idea of the current article is to examine some of the properties that the field of paraconsistent reasoning has been trying to achieve (the postulates provided earlier in this section) and implement these in a wide range of logical formalisms where traditionally the concept of paraconsistency has not played a major role. The first example of such a formalism will be treated in Section 3.

To understand the value of the postulates of non-interference and crash resistance, imagine that one is to apply a knowledge-based system in a safety-critical situation, e.g for the purpose of medical diagnosis. If one then has two completely different patients who have nothing in common, then one does not want the reasoning process regarding the first patient somehow to influence the reasoning process regarding the second patient. In fact, these reasoning processes should have no influence on each other at all. This is the intuition formalized by the postulates of non-interference and crash resistance.

The postulate of backward compatibility is based on a different idea. In various domains of formal reasoning, approaches have emerged that became the 'gold standard' for their particular domain. Examples of these are classical logic as well as logic programming under stable model semantics. Both approaches enjoy a strong commitment, for reasons outside the scope of this article, and any violations regarding non-interference and crash resistance are therefore to be dealt with in a way that makes minimal changes to the original approach. That is, ideally one should only alter the part where the approach 'fails' (violates crash resistance) while keeping its outcome unaltered in all situations where such a failure does not occur. This is the intuition formalized by the postulate of backward compatibility.

[1]As with abstract argumentation, the approaches of preferred, grounded and complete semantics are not backward compatible with stable semantics.

3 An Abstract Account of Semi-Stable Semantics

In this section, we introduce the notion of semi-stable semantics. We choose to introduce it using the formalism of abstract argumentation, and later show how it can be applied to logic programming in Section 4 and to default logic in Section 5.

3.1 Preliminaries

We first start with some basic definitions regarding abstract argumentation based on [26]. In line with [5, 6], we restrict ourselves to argumentation frameworks with a finite set of arguments.

DEFINITION 7 (argumentation framework)
An *argumentation framework* is a pair (Ar, att) where Ar is a finite set of abstract entities called *arguments* and $att \subseteq Ar \times Ar$ is called the *attack* relation.

An argumentation framework can be represented as a directed graph in which the arguments are represented as nodes and the attack relation is represented as arrows. In several examples throughout this article, we will use this graph representation.

If $\mathcal{A}rgs \subseteq Ar$ then we write $(Ar, att)_{|\mathcal{A}rgs}$ as a shorthand for $(\mathcal{A}rgs, \{\langle A, B\rangle \mid \langle A, B\rangle \in att \text{ and } A, B \in \mathcal{A}rgs\})$. In the definition below, $F(\mathcal{A}rgs)$ stands for the set of arguments that are acceptable in the sense of [26].

DEFINITION 8 (defence/conflict-free)
Let (Ar, att) be an argumentation framework, $A \in Ar$ and $\mathcal{A}rgs \subseteq Ar$.
We define A^+ as $\{B \in Ar \mid A \, att \, B\}$ and $\mathcal{A}rgs^+$ as $\{B \in Ar \mid A \, att \, B \text{ for some } A \in \mathcal{A}rgs\}$.
We define A^- as $\{B \in Ar \mid B \, att \, A\}$ and $\mathcal{A}rgs^-$ as $\{B \in Ar \mid B \, att \, A \text{ for some } A \in \mathcal{A}rgs\}$.
$\mathcal{A}rgs$ is *conflict-free* iff $\mathcal{A}rgs \cap \mathcal{A}rgs^+ = \emptyset$. $\mathcal{A}rgs$ *defends* an argument A iff $A^- \subseteq \mathcal{A}rgs^+$.
We define the function $F : 2^{Ar} \to 2^{Ar}$ as $F(\mathcal{A}rgs) = \{A \in Ar \mid A \text{ is defended by } \mathcal{A}rgs\}$.

In the definition below, definitions of grounded, preferred and stable semantics are described in terms of complete semantics, which has the advantage of making the proofs in the remainder of this article more straightforward. These descriptions are not literally the same as the ones provided by Dung [26], so it will be proved that they are in fact equivalent to Dung's original versions of grounded, preferred and stable semantics.

DEFINITION 9 (acceptability semantics)
Let (Ar, att) be an argumentation framework and $\mathcal{A}rgs \subseteq Ar$ be a conflict-free set of arguments.

- $\mathcal{A}rgs$ is *admissible* iff $\mathcal{A}rgs \subseteq F(\mathcal{A}rgs)$.
- $\mathcal{A}rgs$ is a *complete* extension iff $\mathcal{A}rgs = F(\mathcal{A}rgs)$.
- $\mathcal{A}rgs$ is a *grounded* extension iff $\mathcal{A}rgs$ is the minimal (w.r.t. set-inclusion) complete extension.
- $\mathcal{A}rgs$ is a *preferred* extension iff $\mathcal{A}rgs$ is a maximal (w.r.t. set-inclusion) complete extension.
- $\mathcal{A}rgs$ is a *stable* extension iff $\mathcal{A}rgs$ is a complete extension that attacks every argument in $Ar \setminus \mathcal{A}rgs$.

A well-known property of argumentation theory is that for each argumentation framework there exists exactly one grounded extension. It contains all the arguments that are not attacked, as well as those arguments which are directly or indirectly defended by non-attacked arguments. Furthermore,

for each argumentation framework there exists at least one complete extension, at least one preferred extension and zero or more stable extensions.

As an example of how the different semantics operate, consider the argumentation framework of Figure 2 (p. 1216). Here, $\emptyset$, $\{A\}$, $\{B\}$ and $\{B,D\}$ are admissible sets. The complete extensions are $\emptyset$, $\{A\}$ and $\{B,D\}$. The grounded extension is $\emptyset$. The preferred extensions are $\{A\}$ and $\{B,D\}$. The only stable extension is $\{B,D\}$. Similarly, in the argumentation framework of Figure 1 (p. 1215) the admissible sets are $\emptyset$, $\{B\}$ and $\{B,D\}$. There is just one complete extension $\{B,D\}$ which is also the grounded and the only preferred extension. The argumentation framework does not have any stable extensions.

We say that an argument is *credulously justified* under a particular semantics iff it is in at least one extension under this semantics. We say that an argument is *sceptically justified* under a particular semantics iff it is in each extension under this semantics.

Grounded, preferred and stable semantics can be stated in various equivalent ways. For grounded semantics, for instance, one does not actually need to explicitly state the requirement of conflict-freeness.

PROPOSITION 1
Let (Ar, att) be an argumentation framework and let $\mathcal{A}rgs \subseteq Ar$. The following statements are equivalent:

(1) $\mathcal{A}rgs$ is the minimal complete extension (Definition 9)
(2) $\mathcal{A}rgs$ is the minimal fixpoint of F [26, Definition 20]

PROOF.
from 1 to 2: Let $\mathcal{A}rgs$ be the minimal complete extension. Suppose that $\mathcal{A}rgs$ is not a minimal fixpoint of F. Then there exists a proper subset $\mathcal{A}rgs' \subsetneq \mathcal{A}rgs$ which is a fixpoint of F. As $\mathcal{A}rgs$ is already the smallest fixpoint of F that is conflict-free, this can only mean that $\mathcal{A}rgs'$ is not conflict-free. But this is impossible as a subset of a conflict-free set is also conflict-free. Contradiction.
from 2 to 1: Let $\mathcal{A}rgs$ be the minimal fixpoint of F. As the monotonic increasing function F has a unique minimal fixpoint, the minimal fixpoint of F must be unique. From the previous point of this proof ('from 1 to 2'), it then follows that the minimal complete extension is equivalent to this fixpoint.

■

As for preferred semantics, our definition is equivalent to that of [26].

PROPOSITION 2
Let (Ar, att) be an argumentation framework and let $\mathcal{A}rgs \subseteq Ar$. The following statements are equivalent:

(1) $\mathcal{A}rgs$ is a maximal complete extension (Definition 9)
(2) $\mathcal{A}rgs$ is a maximal admissible set [26, Definition 7]

PROOF. This follows from Theorem 25 of [26]. ■

As for stable semantics, there exist at least four equivalent ways of describing it.

PROPOSITION 3
Let (Ar, att) be an argumentation framework and let $\mathcal{A}rgs \subseteq Ar$. The following statements are equivalent:

(1) $\mathcal{A}rgs$ is a complete extension that attacks every argument in $Ar \setminus \mathcal{A}rgs$ (Definition 9)

(2) $\mathcal{A}rgs$ is a preferred extension that attacks every argument in $Ar \backslash \mathcal{A}rgs$
(3) $\mathcal{A}rgs$ is an admissible set that attacks every argument in $Ar \backslash \mathcal{A}rgs$
(4) $\mathcal{A}rgs$ is a conflict-free set that attacks every argument in $Ar \backslash \mathcal{A}rgs$ [26, Definition 13]

PROOF.
from 1 to 2: Let $\mathcal{A}rgs$ be a stable extension. This means that $\mathcal{A}rgs$ is a complete extension that attacks every argument in $Ar \backslash \mathcal{A}rgs$. Suppose that $\mathcal{A}rgs$ is not a preferred extension. That means that there is a complete extension $\mathcal{A}rgs' \supsetneq \mathcal{A}rgs$. But as $\mathcal{A}rgs$ attacks every argument in $Ar \backslash \mathcal{A}rgs$, this means that $\mathcal{A}rgs'$ would not be conflict-free and therefore could not be a complete extension. Contradiction.
from 2 to 1: Trivial (every preferred extension is also a complete extension).
from 2 to 3: From Proposition 2, it follows that a preferred extension is a (maximal) admissible set.
from 3 to 2: Let $\mathcal{A}rgs$ be an admissible set that attacks all arguments in $Ar \backslash \mathcal{A}rgs$. Suppose that $\mathcal{A}rgs$ is not a preferred extension. This means that there exists an admissible set $\mathcal{A}rgs' \supseteq \mathcal{A}rgs$. But as $\mathcal{A}rgs$ attacks all arguments in $Ar \backslash \mathcal{A}rgs$, this would mean that $\mathcal{A}rgs'$ is not conflict-free and therefore could not be an admissible set. Contradiction.
from 3 to 4: This follows directly from the fact that an admissible set is conflict-free.
from 4 to 3: Let $\mathcal{A}rgs$ be a conflict-free set that attacks all arguments in $Ar \backslash \mathcal{A}rgs$. Then, every argument that attacks $\mathcal{A}rgs$ is also attacked by $\mathcal{A}rgs$. This means that $\mathcal{A}rgs$ is an admissible set. ■

The advantage of Propositions 1, 2 and 3 is that they offer a lot of flexibility for choosing the definition of a particular semantics that is best suited for a particular proof. For most purposes, we will apply the descriptions of the semantics as defined in Definition 9. It will be explicitly mentioned where we do otherwise.

3.2 Semi-stable semantics for abstract argumentation

The notion of semi-stable semantics, as put forward in the current article, is quite similar to that of preferred semantics. The only difference is that instead of maximizing $\mathcal{A}rgs$, one maximizes $\mathcal{A}rgs \cup \mathcal{A}rgs^+$.

DEFINITION 10
Let (Ar, att) be an argumentation framework and $\mathcal{A}rgs \subseteq Ar$. $\mathcal{A}rgs$ is called a *semi-stable extension* iff $\mathcal{A}rgs$ is a complete extension where $\mathcal{A}rgs \cup \mathcal{A}rgs^+$ is maximal.

If $\mathcal{A}rgs$ is a set of arguments, then $\mathcal{A}rgs \cup \mathcal{A}rgs^+$ is called its *range*—a notion first introduced by Bart Verheij [51].

For every (finite) argumentation framework, there exists at least one semi-stable extension. This is because there exists at least one complete extension (the grounded) and the fact that the argumentation framework is finite implies that there exist at most a finite number of complete extensions. The semi-stable extensions are then simply those complete extensions in which some property (its range) is maximal.

Just like preferred semantics can be expressed as a maximal complete extension or as a maximal admissible set, semi-stable semantics can be expressed as a complete extension with maximal range or as an admissible set with maximal range.

PROPOSITION 4
Let (Ar, att) be an argumentation framework and $\mathcal{A}rgs \subseteq Ar$. The following statements are equivalent:

(1) $\mathcal{A}rgs$ is a complete extension such that $\mathcal{A}rgs \cup \mathcal{A}rgs^+$ is maximal (Definition 10)
(2) $\mathcal{A}rgs$ is an admissible set such that $\mathcal{A}rgs \cup \mathcal{A}rgs^+$ is maximal

PROOF.
from 2 to 1: Being a complete extension is a stronger condition than being an admissible set, so we only need to prove that an admissible set $\mathcal{A}rgs$ where $\mathcal{A}rgs \cup \mathcal{A}rgs^+$ is maximal is also a complete extension. Suppose this is not the case. Then there must be an argument $B \notin \mathcal{A}rgs$ that is defended by $\mathcal{A}rgs$. This means that every argument C that attacks B is attacked by an argument in $\mathcal{A}rgs$. Therefore, $B \notin \mathcal{A}rgs^+$ (otherwise $\mathcal{A}rgs$ would not be conflict-free). This means that $\mathcal{A}rgs \cup \{B\}$ is conflict-free and self-defending, and thus an admissible set. But this would mean that $\mathcal{A}rgs$ is not an admissible set for which $\mathcal{A}rgs \cup \mathcal{A}rgs^+$ is maximal. Contradiction.
from 1 to 2: Being an admissible set is a weaker condition than being a complete extension. We therefore only need to prove that maximality still holds under this weaker condition. Suppose that $\mathcal{A}rgs \cup \mathcal{A}rgs^+$ would not be maximal. This means there exists an admissible set $\mathcal{A}rgs'$ such that $(\mathcal{A}rgs' \cup \mathcal{A}rgs'^+) \supsetneq (\mathcal{A}rgs \cup \mathcal{A}rgs^+)$. Assume without loss of generality that $(\mathcal{A}rgs' \cup \mathcal{A}rgs'^+)$ is maximal. From the previous point ('from 2 to 1'), it then follows that $\mathcal{A}rgs'$ would be a complete extension. But then $\mathcal{A}rgs$ would not have been a complete extension where $\mathcal{A}rgs \cup \mathcal{A}rgs^+$ is maximal. Contradiction. ■

It is not difficult to see that being a stable extension is a stronger condition than being a semi-stable extension.

THEOREM 2
Let $\mathcal{A}rgs$ be a stable extension of an argumentation framework (Ar, att). $\mathcal{A}rgs$ is also a semi-stable extension of (Ar, att).

PROOF. Let $\mathcal{A}rgs$ be a stable extension of (Ar, att). Then $\mathcal{A}rgs$ is a complete extension that attacks every argument in $Ar \backslash \mathcal{A}rgs$. This means that $\mathcal{A}rgs \cup \mathcal{A}rgs^+ = Ar$. Therefore, $\mathcal{A}rgs \cup \mathcal{A}rgs^+$ is maximal (it cannot be a proper superset of Ar). Therefore, $\mathcal{A}rgs$ is a semi-stable extension. ■

It is in general not the case that each semi-stable extension is also a stable extension. This is illustrated by the following example.

EXAMPLE 1
Let (Ar, att) be the argumentation framework of which a graphical representation is shown in Figure 1. Here, $\{B, D\}$ is a semi-stable extension that is not a stable extension.

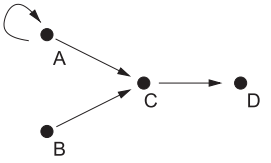

FIGURE 1. $\{B, D\}$ is a semi-stable extension but not a stable extension.

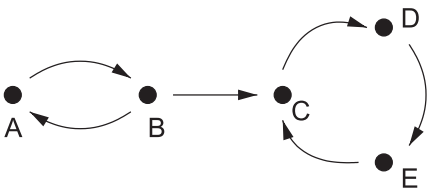

FIGURE 2. $\{A\}$ is a preferred extension but not a semi-stable extension.

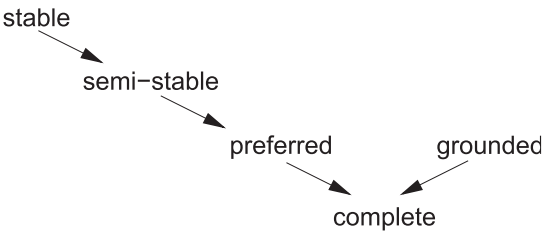

FIGURE 3. A brief overview of argument-based semantics.

It can also be observed that being a semi-stable extension is a stronger condition than being a preferred extension.

THEOREM 3
Let $\mathcal{A}rgs$ be a semi-stable extension of argumentation framework (Ar, att). Then $\mathcal{A}rgs$ is also a preferred extension of (Ar, att).

PROOF. Let $\mathcal{A}rgs$ be a semi-stable extension of (Ar, att). Suppose $\mathcal{A}rgs$ is not a preferred extension of (Ar, att). Then there exists a set $\mathcal{A}rgs' \supsetneq \mathcal{A}rgs$ such that $\mathcal{A}rgs'$ is a complete extension. From $\mathcal{A}rgs' \supsetneq \mathcal{A}rgs$ it follows, however, that $\mathcal{A}rgs'^{+} \supseteq \mathcal{A}rgs^{+}$. Therefore, $(\mathcal{A}rgs' \cup \mathcal{A}rgs'^{+}) \supsetneq (\mathcal{A}rgs \cup \mathcal{A}rgs^{+})$. This implies that $\mathcal{A}rgs$ is not a semi-stable extension, since $\mathcal{A}rgs \cup \mathcal{A}rgs^{+}$ would not be maximal. Contradiction. ■

It is in general not the case that every preferred extension is also a semi-stable extension. This is illustrated by the following example.

EXAMPLE 2
Let (Ar, att) be the argumentation framework of which a graphical representation is shown in Figure 2. Here, $\{A\}$ is a preferred extension that is not a semi-stable extension. The only semi-stable extension is $\{B, D\}$.

The overall position of semi-stable semantics is shown in Figure 3. Each stable extension is a semi-stable extension; each semi-stable extension is a preferred extension; each preferred extension is a complete extension and the grounded extension is a complete extension.

Since each semi-stable extension is also a preferred extension, a straightforward way of solving decision problems related to semi-stable semantics (like credulous or sceptical acceptance) would be to compute all preferred extensions (using for instance the algorithm specified in [25]) and then to focus on those that are also semi-stable. This essentially means selecting the preferred extensions with a maximal range. In many cases, however, there also exist alternative ways of determining whether an argument is credulously or sceptically justified under semi-stable semantics.

THEOREM 4
Let (Ar, att) be an argumentation framework, and let $A \in Ar$.

(1) If A is in the grounded extension, then A is in every semi-stable extension.
(2) If A is not in any admissible set, then A is not in any semi-stable extension.
(3) If A is in an admissible set and is not attacked by any admissible set then A is in at least one semi-stable extension.

PROOF.
(1) This follows from the fact that the grounded extension is a subset of each complete extension [26], and the fact that each semi-stable extension is a complete extension.
(2) This follows from the fact that each semi-stable extension is an admissible set.
(3) The fact that A is not attacked by an admissible set also means that A is not attacked by a complete extension, and therefore that A is also not attacked by a semi-stable extension. That is, for any semi-stable extension $\mathcal{A}rgs$, it holds that $A \notin \mathcal{A}rgs^+$. The fact that A is part of an admissible set means that there is a preferred extension containing A. Let $\mathcal{A}rgs'$ be a preferred extension that contains A and where (within the constraint that it contains A) $\mathcal{A}rgs' \cup \mathcal{A}rgs'^+$ is maximal. As for any semi-stable extension $\mathcal{A}rgs$ it holds that $A \notin \mathcal{A}rgs^+$, it also holds for any semi-stable extension not containing A that $A \notin \mathcal{A}rgs \cup \mathcal{A}rgs^+$. Thus, $\mathcal{A}rgs' \cup \mathcal{A}rgs'^+$ cannot be enlarged without losing A. Therefore, $\mathcal{A}rgs'$ is a semi-stable extension. ■

An example of Theorem 4(3) can be found in Figure 2. Here, argument D is in an admissible set but is not attacked by an admissible set. This is because its only attacker (C) is not part of any admissible set. Hence, D is part of a semi-stable extension.

In situations where Theorem 4 is not applicable, it is possible to apply an algorithm for computing all semi-stable extensions. This algorithm, described in [12, 13, 39], yields all semi-stable extensions, without necessarily having to compute all preferred extensions.

It turns out that in argumentation frameworks where there exists at least one stable extension, the semi-stable extensions coincide with the stable extensions, as is expressed by the following theorem.

THEOREM 5
Let (Ar, att) be an argumentation framework that has at least one stable extension. Let SE be the set of stable extensions and let SSE be the set of semi-stable extensions. It holds that $SE = SSE$.

PROOF. We need to prove that: (1) $SE \subseteq SSE$ and (2) $SSE \subseteq SE$.

(1) $SE \subseteq SSE$
This follows directly from Theorem 2.
(2) $SSE \subseteq SE$
Let $SE_i \in SE$ (such an SE_i exists since it is assumed that (Ar, att) has at least one stable extension). It holds that $SE_i \cup SE_i^+ = Ar$. Therefore, every semi-stable extension SSE_i will also have to satisfy that $SSE_i \cup SSE_i^+ = Ar$ (otherwise $SSE_i \cup SSE_i^+$ would not be maximal). This means that every semi-stable extension is also a stable extension. ■

In order to understand the intuition behind semi-stable semantics, it can be useful to express it in terms of *argument labellings* [10, 16]. Given an argumentation framework (Ar, att), an argument labelling is a total function $\mathcal{L}ab \rightarrow \{\texttt{in}, \texttt{out}, \texttt{undec}\}$. In essence, an argument labelling expresses an opinion on which arguments are accepted (labelled `in`), which arguments are rejected (labelled `out`) and which arguments one abstains from having an explicit opinion about (labelled `undec`). Based on the concept of an argument labelling, one can subsequently formulate additional conditions

that aim to describe what it means for such an opinion to be reasonable. One such condition is to require that for each argument $A \in Ar$ it holds that:

(1) if $\mathcal{L}ab(A) = \texttt{in}$ then
for each attacker B of A it holds that $\mathcal{L}ab(B) = \texttt{out}$,
(2) if $\mathcal{L}ab(A) = \texttt{out}$ then
there exists an attacker B of A such that $\mathcal{L}ab(B) = \texttt{in}$, and
(3) if $\mathcal{L}ab(A) = \texttt{undec}$ then
not for each attacker B of A it holds that $\mathcal{L}ab(B) = \texttt{out}$,
and there does not exist an attacker B of A such that $\mathcal{L}ab(B) = \texttt{in}$.

A labelling satisfying this condition is called a *complete labelling* [16]. The idea is that (i) for an argument to be accepted, one has to reject all its attackers; (ii) for an argument to be rejected, one has to accept at least one attacker; and (iii) for an argument to be undecided, one has to have insufficient grounds to accept it (i.e. not all its attackers are rejected) and insufficient grounds to reject it (i.e. it does not have an attacker that is accepted). Complete labellings correspond in a one-to-one relation to complete extensions [16]. Given a complete labelling, one can construct the associated complete extension simply by selecting all `in`-labelled arguments. Similarly, given a complete extension, one can construct the associated complete labelling by having all arguments in the extension labelled `in`, having all arguments attacked by the extension labelled `out` and having all other arguments (that are neither in the extension nor attacked by the extension) labelled `undec` [10, 16].

Just like one can describe the notion of complete semantics in terms of argument labellings (as described above), one can also describe the notions of preferred and grounded semantics in terms of argument labellings. A preferred labelling is defined as a complete labelling where the set of `in`-labelled arguments is maximal (w.r.t. set inclusion) among all complete labellings [10, 16]. Similarly, the grounded labelling can be described as the (unique) complete labelling where the set of `undec`-labelled arguments is maximal (w.r.t. set inclusion) among all complete labellings [10, 16]. That is, if one interprets a complete labelling as a reasonable position one can take based on the conflicting information in the argumentation framework, then the idea of preferred semantics is to try to accept as much as possible, whereas the idea of grounded semantics is to abstain as much as possible. As an aside, the one-to-one correspondence between labellings and extensions also holds for preferred semantics and grounded semantics [16].

In a similar way, one can describe the concepts of stable and semi-stable semantics in terms of argument labellings. A stable labelling is a complete labelling where the set of `undec`-labelled arguments is empty (hence, where each argument is labelled either `in` or `out`), whereas a semi-stable labelling is a complete labelling where the set of `undec`-labelled arguments is minimal (w.r.t. set inclusion) among all complete labellings [10, 16]. In essence, the idea behind stable semantics can be described as a black-and-white view of the world, in which there is no room for shades of grey, and in which neutrality does not exist. An argument is either accepted or rejected, and nothing else. Such a point of view can create difficulties when interpreting the world, and it should come as no surprise that there exist argumentation frameworks where no stable labellings (or extensions) exist. The idea of semi-stable semantics is then to soften up the requirements of stable semantics a bit. Instead of requiring that the set of arguments where one abstains is *empty*, one merely requires that this set is *minimal*. It then follows that every stable labelling is also a semi-stable labelling, and that if there exists at least one stable labelling, then every semi-stable labelling is also a stable labelling (using similar reasoning as in the proof of Theorem 5). It has also been proven that the usual one-to-one correspondence between labellings and extensions also holds for stable and semi-stable semantics [16].

In essence, the idea of semi-stable semantics is that one wants to interpret the information in the argumentation framework in a meaningful way (satisfying the conditions of a complete labelling), whereas at the same time staying as close as possible to the concept of stable semantics. It is like one wants to 'extend' the notion of stable semantics to make it applicable to a wider range of problem descriptions, even those that originally could not be interpreted by stable semantics, while still yielding the same outcome for those problem descriptions that could be interpreted by stable semantics.

Also from a procedural viewpoint, semi-stable semantics can be seen as an extended version of stable semantics. Suppose that, given an argumentation framework, one computes the semi-stable labellings (for instance using the algorithm described in [13]). Now pick any arbitrary semi-stable labelling that one has computed, and examine whether it has at least one argument that is labelled `undec`. If no such argument exists, then the semi-stable labelling that one has inspected is in fact stable. It then follows that all other labellings that have been computed are also stable. Hence, what has been computed is in fact stable semantics. If, on the other hand, the selected labelling *does* have an argument that is labelled `undec`, then one knows that this labelling is not stable. It then follows that no stable labelling exists (so also all other labellings that were computed are not stable). Hence, when applying semi-stable semantics, one can determine whether the result is stable or not just by examining one arbitrarily selected semi-stable labelling (or extension). If this labelling contains only `in`-labelled and `out`-labelled arguments, then what has been computed is in fact stable semantics. If this labelling also contains an `undec`-labelled argument, then it is clear that no stable labellings existed, but that the results are still as close to stable semantics as possible. Semi-stable semantics should therefore not be seen as strictly in contrast with stable semantics. It is more like one concept naturally flows into the other.

3.3 Satisfying the postulates

The next thing to show is that semi-stable semantics satisfies non-interference, crash resistance and backward compatibility. To do so, we first describe how abstract argumentation constitutes a logical formalism in the sense of Definition 1. We assume a universe $\mathcal{U}$ containing all possible arguments. That is, for each argumentation framework (Ar, att) it holds that $Ar \subseteq \mathcal{U}$. We then define $\mathcal{F}$ormulas (Definition 1) as $\mathcal{U} \cup \{\langle A_1, A_2\rangle \mid A_1, A_2 \in \mathcal{U}\}$. An argumentation framework (Ar, att) can then be represented as the set of formulas $Ar \cup att$. If Φ is the set of formulas representing argumentation framework $AF = (Ar, att)$ then we define:

- $Cn_{\text{admissible}}(\Phi)$ to be the set of all admissible sets of AF.
- $Cn_{\text{complete}}(\Phi)$ to be the set of all complete extensions of AF.
- $Cn_{\text{grounded}}(\Phi)$ to be the set containing the grounded extension of AF as its single element.
- $Cn_{\text{preferred}}(\Phi)$ to be the set containing all preferred extensions of AF.
- $Cn_{\text{stable}}(\Phi)$ to be the (possibly empty) set containing all stable extensions of AF.
- $Cn_{\text{semi-stable}}(\Phi)$ to be the set containing all semi-stable extensions of AF.

As an example, the argumentation framework of Figure 2 can be represented as a set of formulas $\Phi = \{A, B, C, D, E, \langle A,B\rangle, \langle B,A\rangle, \langle B,C\rangle, \langle C,D\rangle, \langle D,E\rangle, \langle E,C\rangle\}$ and it holds that $Cn_{\text{admissible}}(\Phi) = \{\emptyset, \{A\}, \{B\}, \{B,D\}\}$, $Cn_{\text{complete}}(\Phi) = \{\emptyset, \{A\}, \{B,D\}\}$, $Cn_{\text{grounded}} = \{\emptyset\}$, $Cn_{\text{preferred}} = \{\{A\}, \{B,D\}\}$, and $Cn_{\text{stable}} = Cn_{\text{semi-stable}} = \{\{B,D\}\}$. We sometimes abuse notation and write things like $Cn_{\text{semi-stable}}(AF)$ instead of $Cn_{\text{semi-stable}}(\Phi)$ where Φ is the set of formulas associated with AF.

The first property to be proved is that of backward compatibility.

THEOREM 6
In abstract argumentation, semi-stable semantics is backward compatible with stable semantics.

PROOF. Let $AF=(Ar,att)$ be an argumentation framework that is not contaminating under Cn_{stable}. It then follows that AF has at least one stable extension. From Theorem 5, it then follows that the set of semi-stable extensions of AF is equal to the set of stable extensions of AF. That is, $Cn_{\text{semi-stable}}(AF)=Cn_{\text{stable}}(AF)$. ■

It should be noticed that semantics like grounded, preferred and complete are not backward compatible with stable semantics. The argumentation framework of Figure 2, for instance, is not contaminating under stable semantics, so backward compatibility would require the same outcome as stable semantics, where precisely one extension is yielded: $\{B,D\}$. Yet, the grounded extension is $\emptyset$, the preferred extensions are $\{A\}$ and $\{B,D\}$, and the complete extensions are $\emptyset$, $\{A\}$ and $\{B,D\}$. Hence, grounded, preferred and complete semantics are not backward compatible with stable semantics.

Apart from backward compatibility with stable semantics, semi-stable semantics also satisfies non-interference.

THEOREM 7
Abstract argumentation under semi-stable semantics satisfies non-interference.

PROOF. Let $\mathcal{F}_1$ and $\mathcal{F}_2$ be the sets of formulas associated with, respectively, argumentation framework $AF_1=(Ar_1,att_1)$ and $AF_2=(Ar_2,att_2)$ such that $\mathcal{F}_1$ and $\mathcal{F}_2$ are syntactically disjoint. In order to show non-interference, we have to show that:

- $Cn_{\text{semi-stable}}(\mathcal{F}_1)_{|\text{atoms}(\mathcal{F}_1)}=Cn_{\text{semi-stable}}(\mathcal{F}_1\cup\mathcal{F}_2)_{|\text{atoms}(\mathcal{F}_1)}$, and
- $Cn_{\text{semi-stable}}(\mathcal{F}_2)_{|\text{atoms}(\mathcal{F}_2)}=Cn_{\text{semi-stable}}(\mathcal{F}_1\cup\mathcal{F}_2)_{|\text{atoms}(\mathcal{F}_2)}$

It holds that $atoms(\mathcal{F}_1)=Ar_1$ and $atoms(\mathcal{F}_2)=Ar_2$. From the fact that $\mathcal{F}_1$ and $\mathcal{F}_2$ are syntactically disjoint it then follows that $Ar_1\cap Ar_2=\emptyset$. Let $\mathcal{F}_3=\mathcal{F}_1\cup\mathcal{F}_2$. It then holds that $\mathcal{F}_3$ is the set of formulas associated with argumentation framework $AF_3=(Ar_1\cup Ar_2,att_1\cup att_2)$. In essence, AF_3 consists of two disjoint graphs (AF_1 and AF_2) with no connections between them. That is, an argument originating from AF_1 cannot attack any arguments originating from AF_2, and vice versa. In order to prove non-interference, it then suffices to prove that:

- $Cn_{\text{semi-stable}}(AF_1)_{|Ar_1}=Cn_{\text{semi-stable}}(AF_3)_{|Ar_1}$, and
- $Cn_{\text{semi-stable}}(AF_2)_{|Ar_2}=Cn_{\text{semi-stable}}(AF_3)_{|Ar_2}$

We now prove the first property (the proof of the second property is similar).
'$\subseteq$': let S_1 be a semi-stable extension of AF_1. We now have to prove that there exists a semi-stable extension S_3 of AF_3 such that $S_3\cap Ar_1=S_1$. Let S_2 be a semi-stable extension of AF_2, and let $S_3=S_1\cup S_2$. We now prove that S_3 is a semi-stable extension of AF_3. First of all, S_3 is conflict-free. This follows from the fact that S_1 and S_2 are conflict-free and that no argument in S_1 attacks any argument in S_2 and vice versa (this is because AF_1 and AF_2 are syntactically disjoint). The next thing to prove is that S_3 is a fixpoint of F under AF_3.

$S_3\subseteq F(S_3)$: Let $A\in S_3$. We distinguish two cases.

(1) $A\in Ar_1$. Then $A\in S_1$. From the fact that S_1 is a semi-stable and therefore also complete extension of AF_1, it then follows that $A\in F(S_1)$. Since F is a monotonic function and $S_1\subseteq S_3$ it follows that $A\in F(S_3)$.

(2) $A \in Ar_2$. Then $A \in S_2$. From the fact that S_2 is a semi-stable and therefore also complete extension of AF_2, it then follows that $A \in F(S_2)$. Since F is a monotonic function and $S_2 \subseteq S_3$ it follows that $A \in F(S_3)$.

$F(S_3) \subseteq S_3$: Let $A \in F(S_3)$. We distinguish two cases.

(1) $A \in Ar_1$. Then for every B that attacks A there exists a $C \in S_3$ that attacks B. From the fact that AF_1 and AF_2 are syntactically disjoint, it follows that $B \in Ar_1$ and $C \in Ar_1$. The fact that $C \in Ar_1$ and $C \in S_3$ imply that $C \in S_1$, hence $A \in F(S_1)$. From the fact that S_1 is a complete extension it then follows that $A \in S_1$, which together with the fact that $S_1 \subseteq S_3$ implies that $A \in S_3$.
(2) $A \in Ar_2$. Then for every B that attacks A there exists a $C \in S_3$ that attacks B. From the fact that AF_1 and AF_2 are syntactically disjoint, it follows that $B \in Ar_2$ and $C \in Ar_2$. The fact that $C \in Ar_2$ and $C \in S_3$ imply that $C \in S_2$, hence $A \in F(S_2)$. From the fact that S_2 is a complete extension it then follows that $A \in S_2$, which together with the fact that $S_2 \subseteq S_3$ implies that $A \in S_3$.

From the fact that S_3 is a conflict-free set with $S_3 \subseteq F(S_3)$ and $F(S_3) \subseteq S_3$ it then follows that S_3 is a complete extension of AF_3. We now prove that this complete extension also has a maximal range. Suppose there exists a complete extension S_3' with a bigger range than S_3. That is, $S_3 \cup S_3^+ \subsetneq S_3' \cup S_3'^+$. Let $S_1' = S_3' \cap Ar_1$ and $S_2' = S_3' \cap Ar_2$. We first prove that S_1' is a complete extension of AF_1 (the proof that S_2' is a complete extension of AF_2 is similar and will therefore be omitted). Conflict-freeness of S_1' follows from the fact that S_3' is conflict-free. We now prove that S_1' is a fixpoint of F.

$S_1' \subseteq F(S_1')$: Let $A \in S_1'$. Then from the fact that $S_1' \subseteq S_3'$ it follows that $A \in S_3'$. From the fact that S_3' is a complete extension it follows that $A \in F(S_3')$. That is, for each B that attacks A, there exists a $C \in S_3'$ that attacks B. From the fact that $A \in S_1'$ it also follows that $A \in Ar_1$, and from the fact that AF_1 and AF_2 are syntactically disjoint it then follows that $B \in Ar_1$ and $C \in Ar_1$. From $C \in Ar_1$ and $C \in S_3'$ it follows that $C \in S_1'$, so $A \in F(S_1')$.
$F(S_1') \subseteq S_1'$: Let $A \in F(A_1')$. Then for each B that attacks A there exists a $C \in S_1'$ that attacks B. From the fact that $S_1' \subseteq S_3'$ and that $S_1' \subseteq Ar_1$ it follows that $C \in S_3'$ and $C \in Ar_1$. From the fact that $C \in S_3'$ it follows that $A \in F(S_3')$. The fact that S_3' is a complete extension then implies that $A \in S_3'$. This, together with the fact that $A \in Ar_1$, then implies that $A \in S_1'$.

From the fact that S_1' is a conflict-free set with $S_1' \subseteq F(S_1')$ and $F(S_1') \subseteq S_1'$, it follows that S_1' is a complete extension of AF_1. For similar reasons it also holds that S_2' is a complete extension of AF_2. From the facts that $S_3 = S_1 \cup S_2$ and $S_3' = S_1' \cup S_2'$, together with the earlier assumed property that $S_3 \cup S_3^+ \subsetneq S_3' \cup S_3'^+$ it follows that $(S_1 \cup S_2) \cup (S_1 \cup S_2)^+ \subsetneq (S_1' \cup S_2') \cup (S_1' \cup S_2)^+$, which can be rewritten as $S_1 \cup S_1^+ \cup S_2 \cup S_2^+ \subsetneq S_1' \cup S_1'^+ \cup S_2' \cup S_2'^+$. The fact that AF_1 and AF_2 are syntactically disjoint implies that $S_1 \cup S_1^+ \subsetneq S_1' \cup S_1'^+$ or $S_2 \cup S_2^+ \subsetneq S_2' \cup S_2'^+$. But then either S_1 would not be a semi-stable extension of AF_1 or S_2 would not be a semi-stable extension of AF_2. Contradiction.
'$\supseteq$': Let S_3 be a semi-stable extension of AF_3 and let $S_1 = S_3 \cap Ar_1$. We have to prove that S_1 is a semi-stable extension of AF_1. The fact that S_1 is conflict-free follows directly from the fact that S_3 is conflict-free. We now prove that S_1 is a fixpoint of F.

$S_1 \subseteq F(S_1)$: Let $A \in S_1$. Then from the fact that $S_1 \subseteq S_3$ it follows that $A \in S_3$. From the fact that S_3 is a complete extension it follows that $A \in F(S_3)$. That is, for each B that attacks A, there exists a $C \in S_3$ that attacks B. From the fact that $A \in S_1$ it also follows that $A \in Ar_1$, and from the fact that

AF_1 and AF_2 are syntactically disjoint it then follows that $B \in Ar_1$ and $C \in Ar_1$. From $C \in Ar_1$ and $C \in S_3$ it follows that $C \in S_1$, so $A \in F(S_1)$.

$F(S_1) \subseteq S_1$: Let $A \in F(A_1)$. Then for each B that attacks A there exists a $C \in S_1$ that attacks B. From the fact that $S_1 \subseteq S_3$ and that $S_1 \subseteq Ar_1$ it follows that $C \in S_3$ and $C \in Ar_1$. From the fact that $C \in S_3$ it follows that $A \in F(S_3)$. The fact that S_3 is a complete extension then implies that $A \in S_3$. This, together with the fact that $A \in Ar_1$, then implies that $A \in S_1$.

From the fact that S_1 is a conflict-free set with $S_1 \subseteq F(S_1)$ and $F(S_1) \subseteq S_1$, it follows that S_1 is a complete extension of AF_1. We now prove that S_1 also has a maximal range. Suppose this was not the case. Then there would exist a complete extension S_1' of AF_1 with a range bigger than S_1. That is, $S_1 \cup S_1^+ \subsetneqq S_1' \cup S_1'^+$. Let $S_3' = S_1' \cup (S_3 \backslash Ar_1)$. We now prove that S_3' is a complete extension with a bigger range than S_3. We first prove that S_3' is conflict-free. Suppose there exists arguments $A, B \in S_3'$ such that A attacks B. We distinguish four possibilities.

(1) $A \in S_1$ and $B \in (S_3 \backslash Ar_1)$. This implies that $A \in Ar_1$ and $B \in Ar_2$. But since AF_1 and AF_2 are syntactically disjoint, it follows that A cannot attack B under AF_3. Contradiction.
(2) $B \in S_1$ and $A \in (S_3 \backslash Ar_1)$. This implies that $B \in Ar_1$ and $A \in Ar_2$. But since AF_1 and AF_2 are syntactically disjoint, it follows that A cannot attack B under AF_3. Contradiction.
(3) $A, B \in S_1$. Then S_1 would not be conflict-free. Contradiction.
(4) $A, B \in (S_3 \backslash Ar_1)$. Then S_3 would not be conflict-free. Contradiction.

Since all possibilities for S_3' not to be conflict-free result in a contradiction, it follows that S_3' is conflict-free. Before continuing to prove that S_3' is a complete extension of AF_3, we first prove that S_1' is a complete extension of AF_1 and $S_3 \backslash Ar_1$ is a complete extension of AF_2. The fact that S_1' is a complete extension of AF_1 follows directly from the fact that it is a semi-stable extension of AF_1. The fact that $S_3 \backslash Ar_1$ is a complete extension of AF_2 can be seen as follows.

$S_3 \backslash Ar_1 \subseteq F(S_3 \backslash Ar_1)$: Let $A \in S_3 \backslash Ar_1$. Then $A \in S_3$. From the fact that S_3 is a complete extension it follows that $A \in F(S_3)$. So for every B that attacks A, there exists a $C \in S_3$ such that C attacks B. From the fact that $A \in S_3 \backslash Ar_1$ it follows that $A \in Ar_2$. From the fact that AF_1 and AF_2 are syntactically disjoint it then also follows that $B \in Ar_2$ and $C \in Ar_2$. Therefore, $C \in S_3 \backslash Ar_1$, so $A \in F(S_3 \backslash Ar_1)$.

$F(S_3 \backslash Ar_1) \subseteq S_3 \backslash Ar_1$: Let $A \in F(S_3 \backslash Ar_1)$. Then for each B that attacks A, there exists a $C \in S_3 \backslash Ar_1$ that attacks B. The fact that $C \in S_3 \backslash Ar_1$ implies that $C \in Ar_2$, and from the fact that AF_1 and AF_2 are syntactically disjoint it also follows that $B \in Ar_2$ and $A \in Ar_2$. From the fact that F is a monotonic function and that $A \in F(S_3 \backslash Ar_1)$ it follows that $A \in F(S_3)$. From the fact that S_3 is a complete extension (as a consequence of being a semi-stable extension) it then follows that $A \in S_3$. From this, together with the facts that $A \in Ar_2$ and that AF_1 and AF_2 are syntactically disjoint, it follows that $A \in S_3 \backslash Ar_1$.

From the facts that $S_3 \backslash Ar_1$ is conflict-free (as a direct consequence of S_3 being conflict-free), $S_3 \backslash Ar_1 \subseteq F(S_3 \backslash Ar_1)$ and $F(S_3 \backslash Ar_1) \subseteq S_3 \backslash Ar_1$ it follows that $S_3 \backslash Ar_1$ is a complete extension of AF_2. The next thing to prove is that S_3' is a fixpoint of F under AF_3.

$S_3' \subseteq F(S_3')$: Let $A \in S_3'$. We distinguish two cases.

(1) $A \in S_1'$. Then from the fact that S_1' is a complete extension of AF_1 it follows that $S_1' = F(S_1')$, so $A \in F(S_1')$. And since F is a monotonic function and $S_1' \subseteq S_3'$ it then follows that $A \in F(S_3')$.

(2) $A \in S_3 \backslash Ar_1$. Then, from the earlier observed fact that $S_3 \backslash Ar_1$ is a complete extension of AF_2 it follows that $S_3 \backslash Ar_1 = F(S_3 \backslash Ar_1)$, so $A \in F(S_3 \backslash Ar_1)$. From the facts that F is a monotonic function and that $S_3 \backslash Ar_1 \subseteq S_3'$ it then follows that $A \in F(S_3')$.

$F(S_3') \subseteq S_3'$: Let $A \in F(S_3')$. Then for each B that attacks A there exists a $C \in S_3'$ that attacks B. We distinguish two possibilities.

(1) $C \in S_3 \backslash Ar_1$. Then $C \in Ar_2$, and as a result of AF_1 and AF_2 being syntactically disjoint, it follows that $B \in Ar_2$ and $A \in Ar_2$. It then follows that $A \in F(S_3 \backslash Ar_1)$. From the earlier observed fact that $S_3 \backslash Ar_1$ is a complete extension it then follows that $A \in S_3 \backslash Ar_1$, and from the fact that $S_3 \backslash Ar_1 \subseteq S_3'$ it then follows that $A \in S_3'$.
(2) $C \in S_1'$. Then $C \in Ar_1$, and as a result of AF_1 and AF_2 being syntactically disjoint, it then follows that $B \in S_1'$ and $A \in S_1'$. It then follows that $A \in F(S_1')$. From the fact that S_1' is a complete extension, it then follows that $A \in S_1'$ and from the fact that $S_1' \subseteq S_3'$ it then follows that $A \in S_3'$.

From the facts that S_3' is conflict-free, $S_3' \subseteq F(S_3')$ and $F(S_3') \subseteq S_3'$ it follows that S_3' is a complete extension of AF_3. We now prove that S_3' has a bigger range than S_3. First, it can be observed that $S_3 = (S_3 \cap Ar_1) \cup (S_3 \backslash Ar_1)$, and since $S_1 = S_3 \cap Ar_1$ it follows that $S_3 = S_1 \cup (S_3 \backslash Ar_1)$. This means that the range of S_3 can be described as $S_1 \cup (S_3 \backslash Ar_1) \cup (S_1 \cup (S_3 \backslash Ar_1))^+$, which is equal to

$$S_1 \cup (S_3 \backslash Ar_1) \cup S_1^+ \cup (S_3 \backslash Ar_1)$$

From the fact that $S_3' = S_1' \cup (S_3 \backslash Ar_1)$ it follows that the range of S_3' is $S_1' \cup (S_3 \backslash Ar_1) \cup (S_1' \cup (S_3 \backslash Ar_1))^+$, which is equal to

$$S_1' \cup (S_3 \backslash Ar_1) \cup S_1'^+ \cup (S_3 \backslash Ar_1)$$

Our assumption that S_1' has a bigger range than S_1 means that $S_1 \cup S_1^+ \subsetneqq S_1' \cup S_1^+$, from which it directly follows that $S_1 \cup (S_3 \backslash Ar_1) \cup S_1^+ \cup (S_3 \backslash Ar_1) \subsetneqq S_1' \cup (S_3 \backslash Ar_1) \cup S_1'^+ \cup (S_3 \backslash Ar_1)$. So S_3' is a complete extension with a bigger range than S_3. But then S_3 would not be a semi-stable extension. Contradiction. ■

From the fact that semi-stable semantics satisfies non-interference, crash resistance follows directly, since semi-stable semantics for formal argumentation is non-trivial.

LEMMA 1
Abstract argumentation under semi-stable semantics is non-trivial.

PROOF. Let $Ar \subseteq \mathcal{U}$ with $Ar \neq \emptyset$. We now have to prove that there exist two argumentation frameworks $AF_1 = (Ar, att_1)$ and $AF_2 = (Ar, att_2)$ such that $Cn_{\text{semi-stable}}(AF_1) \neq Cn_{\text{semi-stable}}(AF_2)$. This is obtained with $att_1 = \emptyset$ and $att_2 = \{(A, A) \mid A \in Ar\}$. In that case, $Cn_{\text{semi-stable}}(AF_1) = \{Ar\}$ and $Cn_{\text{semi-stable}}(AF_2) = \{\emptyset\}$. ■

THEOREM 8
Abstract argumentation under semi-stable semantics satisfies crash resistance.

PROOF. Lemma 1 states that abstract argumentation under semi-stable semantics is non-trivial. Theorem 7 states that abstract argumentation under semi-stable semantics satisfies non-interference. Theorem 1 states that each non-trivial formalism that satisfies non-interference also satisfies crash resistance. ■

4 Applying semi-stable semantics to logic programming

In this section, we apply semi-stable semantics to logic programming. This is done by describing logic programming in terms of formal argumentation, in line with [26]. We then change the semantics from stable to semi-stable and show that the resulting formalism satisfies the paraconsistent properties (non-interference, crash resistance and backward compatibility), which are implemented, as mentioned earlier, in the formal semantics of several paraconsistent logics.

4.1 Preliminaries

We first describe some basic concepts in the field of logic programming.

DEFINITION 11
A *rule* is an expression of the form
$c \leftarrow a_1, \ldots, a_n, \texttt{not}\, b_1, \ldots, \texttt{not}\, b_m$ $(n \geq 0, m \geq 0)$
where c as well as each a_i $(1 \leq i \leq n)$ and each b_j $(1 \leq j \leq m)$ are atoms. We assume that for each $i,j \in \{1,\ldots,n\}$, if $i \neq j$ then $a_i \neq a_j$, and that for each $k,l \in \{1,\ldots,m\}$, if $k \neq l$ then $b_k \neq b_l$. c is called the *head* of the rule, $a_1, \ldots, a_n, \texttt{not}\, b_1, \ldots, \texttt{not}\, b_m$ is called the *body* of the rule. A rule is called *definite* iff $m = 0$.
A *logic program* is a finite set of rules. A logic program is called definite iff each of its rules is definite.
If P is a strict logic program, then $Cl(P)$ (the closure of P) is defined as the smallest set such that for each rule $c \leftarrow a_1, \ldots, a_n$ in P it holds that if $a_1, \ldots, a_n \in Cl(P)$ then $c \in Cl(P)$.
If P is a logic program and S is a set of atoms, then P^S (the Gelfond-Lifschitz reduct) is defined as $\{c \leftarrow a_1, \ldots, a_n \mid c \leftarrow a_1, \ldots, a_n, \texttt{not}\, b_1, \ldots, \texttt{not}\, b_m \in P \text{ and } \neg\exists b_i (1 \leq i \leq m) : b_i \in S\}$. The stable operator $\gamma_P(S)$ is defined as $Cl(P^S)$. S is a stable model of P iff S is a fixpoint of γ_P.

We now provide an argumentation interpretation of logic programming. Our approach differs from that in [26] in that we use tree-based arguments. The advantage of using tree-based arguments is that every rule in the argument is relevant for the derivation of the main conclusion, which is necessary if applying semi-stable semantics should satisfy the properties of non-interference and crash resistance.

DEFINITION 12
Let P be a logic program. An argument A is a tree where each node is labelled with a rule in P such that if a node is labelled with $c \leftarrow a_1, \ldots, a_n, \texttt{not}\, b_1, \ldots, \texttt{not}\, b_m$ then its children are labelled with rules $r_1, \ldots, r_n$ such that $head(r_1) = a_1, \ldots, head(r_n) = a_n$. Furthermore, every rule is allowed to occur at most once in each branch of the tree. If A is an argument then $conc(A)$ is defined as the head of the rule of the root. If $\mathcal{A}rgs$ is a set of arguments then $concs(\mathcal{A}rgs)$ is defined as $\{conc(A) \mid A \in \mathcal{A}rgs\}$. We say that an argument A attacks argument B iff B contains a node labelled with rule $c \leftarrow a_1, \ldots, a_n, \texttt{not}\, b_1, \ldots, \texttt{not}\, b_m$ and $conc(A) = b_j$ for some $1 \leq j \leq m$. We write $AF_P = (Ar_P, att_P)$ for the thus defined argumentation framework associated with P.

We sometimes abuse terminology and talk about a set of atoms 'attacking' an argument, a set of arguments, a rule or a set of rules.

As an example of how Definition 12 is applied, consider the following logic program P_1:
$a \leftarrow$
$c \leftarrow a, \texttt{not}\, b$
$d \leftarrow a, \texttt{not}\, c, \texttt{not}\, d$
$e \leftarrow \texttt{not}\, d$

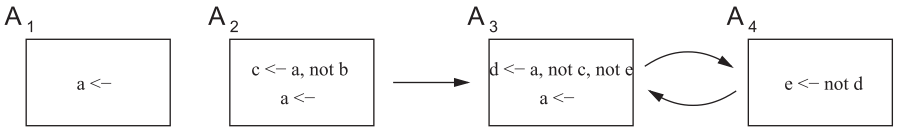

FIGURE 4. The argumentation framework associated with logic program P_1.

Based on P_1, one can construct four arguments, which attack each other according to the argumentation framework of Figure 4.

We are now ready to specify the entailment yielded by our argumentation interpretation of logic programming under stable semantics.

DEFINITION 13
Let P be a logic program. We define $Cn_{\text{stable}}(P)$ as $\{concs(\mathcal{A}rgs) \mid \mathcal{A}rgs$ is a stable extension of $AF_P\}$.

In the argumentation framework of Figure 4, there exists just one stable extension $\{A_1, A_2, A_4\}$ with associated conclusions $\{a, c, e\}$. Hence, $Cn_{\text{stable}}(P) = \{\{a, c, e\}\}$.

Our argumentation interpretation of logic programming under stable semantics is equivalent to the standard Gelfond–Lifschitz stable model semantics.

THEOREM 9
Let P be a logic program. It holds that $S \in Cn_{\text{stable}}(P)$ iff S is a stable model of P.

PROOF.
'$\Rightarrow$': Let $S \in Cn_{\text{stable}}(P)$. This means there exists a stable extension $\mathcal{A}rgs$ of AF_P such that $concs(\mathcal{A}rgs) = S$. We now show that S is also a stable model of P. For this, we need to show that:

(1) $S \subseteq \gamma_P(S)$. Let $e \in S$, so $e \in concs(\mathcal{A}rgs)$. Let $A \in \mathcal{A}rgs$ be an argument with $conc(A) = e$. The fact that A is in the stable extension $\mathcal{A}rgs$ means that A is not attacked by $\mathcal{A}rgs$. That is, A is not 'attacked' by $S = concs(\mathcal{A}rgs)$. This means that each rule of A is not 'attacked' by S. So each rule of A is represented in P^S (formally: for each rule $c \leftarrow a_1, \ldots, a_n, \texttt{not}\, b_1, \ldots, \texttt{not}\, b_m$ of A there exists a rule $c \leftarrow a_1, \ldots, a_n$ in P^S). This then implies that the conclusion of A (e) can still be derived using the rules in P^S. That is, $e \in Cl(P^S)$, which can be rewritten as $e \in \gamma_P(S)$.
(2) $\gamma_P(S) \subseteq S$. Let $e \notin S$, so $e \notin concs(\mathcal{A}rgs)$. Then there is no argument $A \in \mathcal{A}rgs$ with $conc(A) = e$. The fact that $\mathcal{A}rgs$ is a stable extension then implies that each argument A with $conc(A) = e$ is attacked by $\mathcal{A}rgs$. So each argument A with $conc(A) = e$ is 'attacked' by S. This implies that there is no derivation for e using the rules in P^S. That is, $e \notin Cl(P^S)$, so $e \notin \gamma_P(S)$.

'$\Leftarrow$': Let S be a stable model of P. That is, $S = \gamma_P(S)$. We now need to show that $S \in Cn_{\text{stable}}(P)$. We do this by constructing a stable extension $\mathcal{A}rgs$ of AF_P such that $concs(\mathcal{A}rgs) = S$. Let $\mathcal{A}rgs$ be the set of arguments that are not 'attacked' by S. We first prove that $concs(\mathcal{A}rgs) = S$.

- $concs(\mathcal{A}rgs) \subseteq S$. Let $e \in concs(\mathcal{A}rgs)$. Then $\mathcal{A}rgs$ contains an argument A with conclusion e that is not 'attacked' by S. It then follows that $e \in Cl(P^S)$, so $e \in \gamma_P(S)$. From the fact that S is a fixpoint of γ_P it then follows that $e \in S$.
- $S \subseteq concs(\mathcal{A}rgs)$. Let $e \in S$. From the fact that S is a fixpoint of γ_P it follows that $e \in \gamma_P(S)$, so $e \in Cl(P^S)$, meaning that there exists a derivation for e using the rules in P^S. It then also follows that there is an argument A with conclusion e that is not 'attacked' by S, so $A \in \mathcal{A}rgs$, so $e \in concs(\mathcal{A}rgs)$.

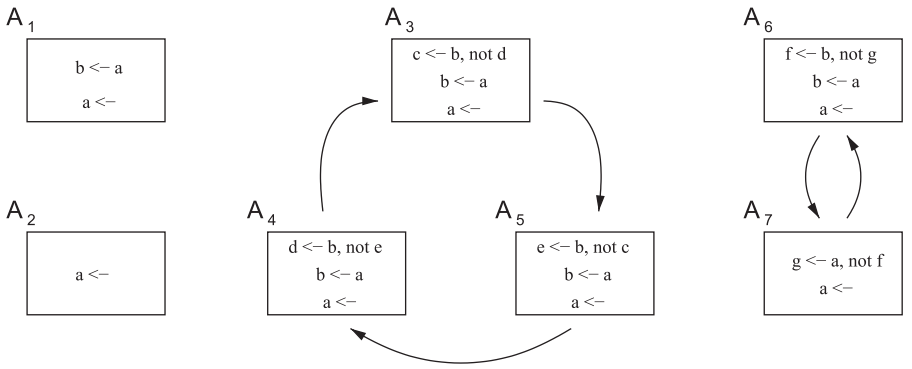

FIGURE 5. The argumentation framework associated with logic program P_2.

Now that it has been proved that $concs(\mathcal{A}rgs) = S$, the next thing to be proved is that $\mathcal{A}rgs$ is a stable extension of AF_P:

(1) $\mathcal{A}rgs$ is conflict-free. Suppose this is not the case. Then there exists $A, B \in \mathcal{A}rgs$ such that A attacks B, so $conc(A)$ 'attacks' B, so $concs(\mathcal{A}rgs)$ 'attacks' B, so S 'attacks' $B \in \mathcal{A}rgs$. But $\mathcal{A}rgs$ is the set of all arguments not 'attacked' by S. Contradiction.
(2) $\mathcal{A}rgs$ attacks each argument in $Ar_P \setminus \mathcal{A}rgs$. Let $A \in Ar_P \setminus \mathcal{A}rgs$. Then from the fact that $A \notin \mathcal{A}rgs$ it follows that A is 'attacked' by S, so A is 'attacked' by $concs(\mathcal{A}rgs)$, so A is attacked by $\mathcal{A}rgs$. ■

4.2 *Semi-stable semantics for logic programming*

We now apply the concept of semi-stable semantics to the argumentation interpretation of logic programming.

DEFINITION 14
Let P be a logic program. We define $Cn_{\text{semi-stable}}(P)$ as $\{concs(\mathcal{A}rgs) \mid \mathcal{A}rgs$ is a semi-stable extension of $AF_P\}$.

As an example of how Definition 14 is applied, consider the following logic program P_2.
$a \leftarrow$
$b \leftarrow a$
$c \leftarrow b, \texttt{not}\, d$
$d \leftarrow b, \texttt{not}\, e$
$e \leftarrow b, \texttt{not}\, c$
$f \leftarrow b, \texttt{not}\, g$
$g \leftarrow a, \texttt{not}\, f$
The associated argumemtation framework AF_{P_2} is shown in Figure 5.

Here, stable semantics yields no extensions at all. That is, $Cn_{\text{stable}}(P_2) = \emptyset$. Semi-stable semantics, on the other hand, yields two extensions: $\{A_1, A_2, A_6\}$ and $\{A_1, A_2, A_7\}$. Hence, $Cn_{\text{semi-stable}}(P_2) = \{\{a, b, f\}, \{a, b, g\}\}$. One could argue that the conflict between c, d and e should not influence the outcome regarding f and g, and this is precisely how things are handled by semi-stable semantics.

We are now ready to prove that logic programming under semi-stable semantics satisfies the postulates of backward compatibility, non-interference and crash resistance.

THEOREM 10
For logic programs, $Cn_{\text{semi-stable}}$ is backward compatible with the Gelfond–Lifschitz stable model semantics.

PROOF. Let P be a logic program that is not contaminating under stable model semantics. This implies that there exists at least one stable model S of P. From Theorem 9, it follows that $S \in Cn_{\text{stable}}(P)$, so there exists a stable extension $\mathcal{A}rgs$ of AF_P such that $concs(\mathcal{A}rgs) = S$. From Theorem 5 it follows that $Cn_{\text{stable}}(AF_P) = Cn_{\text{semi-stable}}(AF_P)$, so also $Cn_{\text{stable}}(P) = Cn_{\text{semi-stable}}(P)$. Since $Cn_{\text{stable}}(P)$ is equivalent to the standard Gelfond–Lifschitz stable model semantics (Theorem 9), it then follows that (still for logic programs with at least one stable model) $Cn_{\text{semi-stable}}$ is equivalent with stable model semantics. ■

LEMMA 2
Let P_1 and P_2 be two syntactically disjoint programs, and let $P_3 = P_1 \cup P_2$. It holds that $Ar_{P_3} = Ar_{P_1} \cup Ar_{P_2}$ and $att_{P_3} = att_{P_1} \cup att_{P_2}$.

PROOF. We first observe that $Ar_{P_1} \subseteq Ar_{P_3}$, because every argument that can be constructed using P_1 can also be constructed using $P_1 \cup P_2$ (= P_3). Similarly, it holds that $Ar_{P_2} \subseteq Ar_{P_3}$, so it follows that $Ar_{P_1} \cup Ar_{P_2} \subseteq Ar_{P_3}$. We now prove that $Ar_{P_3} \subseteq Ar_{P_1} \cup Ar_{P_2}$. Let $A \in Ar_{P_3}$. We distinguish two cases:

(1) The root of A is labelled with a rule from P_1. Then from the fact that P_1 and P_2 are syntactically disjoint, it follows that all children of the root are labelled with rules from P_1, and no children are labelled with rules from P_2. It then follows by induction that each node in the tree is labelled with a rule from P_1 and no node is labelled with a rule from P_2. Therefore, $A \in Ar_{P_1}$ so also $A \in Ar_{P_1} \cup Ar_{P_2}$.
(2) The root of A is labelled with a rule from P_2. Then from the fact that P_1 and P_2 are syntactically disjoint, it follows that all children of the root are labelled with rules from P_2, and no children are labelled with rules from P_1. It then follows by induction that each node in the tree is labelled with a rule from P_2 and no node is labelled with a rule from P_1. Therefore, $A \in Ar_{P_2}$ so also $A \in Ar_{P_1} \cup Ar_{P_2}$.

From the thus observed fact that $Ar_{P_3} \subseteq Ar_{P_1} \cup Ar_{P_2}$, together with the earlier observed fact that $Ar_{P_1} \cup Ar_{P_2} \subseteq Ar_{P_3}$, it follows that $Ar_{P_3} = Ar_{P_1} \cup Ar_{P_2}$.
We can also observe that $att_{P_1} \subseteq att_{P_3}$, because if A and B are arguments based on P_1 such that A attacks B, then A and B are also arguments based on $P_1 \cup P_2$ (= P_3) such that A attacks B. For similar reasons, it holds that $att_{P_2} \subseteq att_{P_3}$, so that it follows that $att_{P_1} \cup att_{P_2} \subseteq att_{P_3}$. We now prove that $att_{P_3} \subseteq att_{P_1} \cup att_{P_2}$. Let A and B be two arguments based in P_3 such that A attacks B. We distinguish two cases:

(1) $conc(A) \in atoms(P_1)$. Then it follows that the root of A is labelled with a rule from P_1, so each node of A is labelled with a rule from P_1, so $A \in Ar_{P_1}$. The fact that A attacks B means that B contains a rule with a weakly negated literal from $atoms(P_1)$. The fact that P_1 and P_2 are syntactically disjoint then implies that each rule in B comes from P_1, so $B \in Ar_{P_1}$. This means that A attacks B under AF_{P_1}.
(2) $conc(A) \in atoms(P_2)$. Then it follows that the root of A is labelled with a rule from P_2, so each node of A is labelled with a rule from P_2, so $A \in Ar_{P_2}$. The fact that A attacks B means that B contains a rule with a weakly negated literal from $atoms(P_2)$. The fact that P_1 and P_2 are

syntactically disjoint then implies that each rule in B comes from P_2, so $B \in Ar_{P_2}$. This means that A attacks B under AF_{P_2}.

From the thus observed fact that $att_{P_3} \subseteq att_{P_1} \cup att_{P_2}$, together with the earlier observed fact that $att_{P_1} \cup att_{P_2} \subseteq att_{P_3}$, it follows that $att_{P_3} = att_{P_1} \cup att_{P_2}$. ■

THEOREM 11
For logic programs, $Cn_{\text{semi-stable}}$ satisfies non-interference.

PROOF. Let P_1 and P_2 be two logic programs that are syntactically disjoint. In order to show non-interference, we need to prove that:

- $Cn_{\text{semi-stable}}(P_1)_{|\text{atoms}(P_1)} = Cn_{\text{semi-stable}}(P_1 \cup P_2)_{|\text{atoms}(P_1)}$ and
- $Cn_{\text{semi-stable}}(P_2)_{|\text{atoms}(P_2)} = Cn_{\text{semi-stable}}(P_1 \cup P_2)_{|\text{atoms}(P_2)}$.

We prove only the first property (the proof of the second property is similar).
'$\subseteq$': Let $S \in Cn_{\text{semi-stable}}(P_1)_{|\text{atoms}(P_1)}$. Since $Cn_{\text{semi-stable}}(P_1)_{|\text{atoms}(P_1)} = Cn_{\text{semi-stable}}(P_1)$ it immediately follows that $S \in Cn_{\text{semi-stable}}(P_1)$. This means there exists a semi-stable extension $\mathcal{A}rgs$ of AF_{P_1} such that $concs(\mathcal{A}rgs) = S$. The fact that semi-stable semantics for abstract argumentation satisfies non-interference (Theorem 7) means that (given that AF_{P_1} and AF_{P_2} are syntactically disjoint):

$$Cn_{\text{semi-stable}}((Ar_{P_1}, att_{P_1}))_{|Ar_{P_1}} = Cn_{\text{semi-stable}}((Ar_{P_1} \cup Ar_{P_2}, att_{P_1} \cup att_{P_2}))_{|Ar_{P_1}}$$

From Lemma 2, it follows that $AF_{P_1 \cup P_2} = (Ar_{P_1} \cup Ar_{P_2}, att_{P_1} \cup att_{P_2})$, so it holds that

$$Cn_{\text{semi-stable}}(AF_{P_1})_{Ar_{P_1}} = Cn_{\text{semi-stable}}(AF_{P_1 \cup P_2})_{|Ar_{P_1}}$$

From the fact that $\mathcal{A}rgs$ is a semi-stable extension of AF_{P_1}, it follows that $\mathcal{A}rgs \in Cn_{\text{semi-stable}}(AF_{P_1})_{|Ar_{P_1}}$, so $\mathcal{A}rgs \in Cn_{\text{semi-stable}}(AF_{P_1 \cup P_2})_{|Ar_{P_1}}$, which then implies that there exists a semi-stable extension $\mathcal{A}rgs'$ of $AF_{P_1 \cup P_2}$ such that $\mathcal{A}rgs' \cap Ar_{P_1} = \mathcal{A}rgs$. The fact that $\mathcal{A}rgs'$ is a semi-stable extension of $(Ar_{P_1} \cup Ar_{P_2}, att_{P_1} \cup att_{P_2})$ implies that $concs(\mathcal{A}rgs') \in Cn_{\text{semi-stable}}(P_1 \cup P_2)$, so $concs(\mathcal{A}rgs')_{|\text{atoms}(P_1)} \in Cn_{\text{semi-stable}}(P_1 \cup P_2)_{|\text{atoms}(P_1)}$. We now prove that $concs(\mathcal{A}rgs')_{|\text{atoms}(P_1)} = S$.

'$\subseteq$': Let $e \in concs(\mathcal{A}rgs')_{|\text{atoms}(P_1)}$. Then $e \in atoms(P_1)$. Let A be an argument in $\mathcal{A}rgs'$ with $conc(A) = e$. Then $A \in Ar_{P_1}$, so $A \in \mathcal{A}rgs$, so $e \in concs(\mathcal{A}rgs)$, so $e \in S$.
'$\supseteq$': Let $e \in S$. Then $e \in concs(\mathcal{A}rgs)$. So there is an $A \in \mathcal{A}rgs$ with $conc(A) = e$. From the fact that $A \in \mathcal{A}rgs$ it follows that $A \in \mathcal{A}rgs'$, so $e \in concs(\mathcal{A}rgs')$. From the fact that $A \in \mathcal{A}rgs$ it also follows that $A \in Ar_{P_1}$, so $e \in atoms(P_1)$, so from the fact that $e \in concs(\mathcal{A}rgs')$ it follows that $e \in concs(\mathcal{A}rgs')_{|\text{atoms}(P_1)}$.

From the thus proved fact that $concs(\mathcal{A}rgs')_{|\text{atoms}(P_1)} = S$, together with the earlier observed fact that $concs(\mathcal{A}rgs')_{|\text{atoms}(P_1)} \in Cn_{\text{semi-stable}}(P_1 \cup P_2)_{|\text{atoms}(P_1)}$ it follows that $S \in Cn_{\text{semi-stable}}(P_1 \cup P_2)_{|\text{atoms}(P_1)}$.
'$\supseteq$': Let $S \in Cn_{\text{semi-stable}}(P_1 \cup P_2)_{|\text{atoms}(P_1)}$. Then there exists a semi-stable extension $\mathcal{A}rgs'$ of $AF_{P_1 \cup P_2}$ with $concs(\mathcal{A}rgs') \cap atoms(P_1) = S$. Like was explained before, it holds that:

$$Cn_{\text{semi-stable}}(AF_{P_1})_{|Ar_{P_1}} = Cn_{\text{semi-stable}}(AF_{P_1 \cup P_2})_{|Ar_{P_1}}$$

Let $\mathcal{A}rgs = \mathcal{A}rgs' \cap Ar_{P_1}$. From the fact that $\mathcal{A}rgs'$ is a semi-stable extension of $AF_{P_1 \cup P_2}$, it follows that $\mathcal{A}rgs \in Cn_{\text{semi-stable}}(AF_{P_1 \cup P_2})_{|Ar_{P_1}}$, so $\mathcal{A}rgs \in Cn_{\text{semi-stable}}(AF_{P_1})_{|Ar_{P_1}}$, which together with the fact

that $\mathcal{A}rgs \subseteq Ar_{P_1}$ implies that $\mathcal{A}rgs \in Cn_{\text{semi-stable}}(AF_{P_1})$, so $concs(\mathcal{A}rgs) \in Cn_{\text{semi-stable}}(P_1)$. Since $concs(\mathcal{A}rgs) \subseteq atoms(P_1)$ it follows that $concs(\mathcal{A}rgs) \in Cn_{\text{semi-stable}}(P_1)_{|\text{atoms}(P_1)}$. We now prove that $concs(\mathcal{A}rgs) = S$.

'$\subseteq$': Let $e \in concs(\mathcal{A}rgs)$. Then there is an $A \in \mathcal{A}rgs$ with $conc(A) = e$. Then $e \in atoms(P_1)$ and $A \in Ar_{P_1}$. From $A \in \mathcal{A}rgs$ it follows that $A \in \mathcal{A}rgs'$, so $e \in concs(\mathcal{A}rgs')$. This, together with $e \in atoms(P_1)$ implies that $e \in concs(\mathcal{A}rgs') \cap atoms(P_1)$, so $e \in S$.

'$\supseteq$': Let $e \in S$. Then $e \in concs(\mathcal{A}rgs') \cap atoms(P_1)$. So there exists an argument $A \in \mathcal{A}rgs'$ with $conc(A) = e$. Moreover, the fact that $c \in atoms(P_1)$ implies that $A \in Ar_{P_1}$. It then follows that $A \in \mathcal{A}rgs' \cap Ar_{P_1}$, so $A \in \mathcal{A}rgs$, so $e \in concs(\mathcal{A}rgs)$.

From the fact that $concs(\mathcal{A}rgs) = S$, together with the fact that $concs(\mathcal{A}rgs) \in Cn_{\text{semi-stable}}(P_1)_{|\text{atoms}(P_1)}$ it follows that $S \in Cn_{\text{semi-stable}}(P_1)_{|\text{atoms}(P_1)}$. ■

LEMMA 3
Logic programming under semi-stable semantics is non-trivial.

PROOF. Let S be a non-empty set of atoms. We now have to prove that there exists two logic programs P_1 and P_2 with $atoms(P_1) = atoms(P_2) = S$, such that $Cn_{\text{semi-stable}}(P_1) \neq Cn_{\text{semi-stable}}(P_2)$. This is obtained with $P_1 = \{e \leftarrow | \, e \in S\}$ and $P_2 = \{e \leftarrow e \, | \, e \in S\}$. In that case, $Cn_{\text{semi-stable}}(P_1) = \{S\}$ and $Cn_{\text{semi-stable}}(P_2) = \{\emptyset\}$. ■

THEOREM 12
Logic programming under semi-stable semantics satisfies crash resistance.

PROOF. Lemma 3 states that abstract argumentation under semi-stable semantics is non-trivial. Theorem 11 states that logic programming under semi-stable semantics satisfies non-interference. Theorem 1 states that each non-trivial logical formalism that satisfies non-interference also satisfies crash resistance. ■

Nearly 20 different approaches to paraconsistent semantics as related to logic programming are surveyed in [23]. Most of them are based on *ad hoc* multivalued models just to provide meaningful queries even when classical models of a knowledge base are unavailable. Others use a mixture of logical mechanisms and computational devices that allow contradictions to be detected, but do not offer sufficient control over real reasoning with contradictions, or fully cope with default negations. An alternative has been proposed by Sakama and Inoue (cf. [49]) by means of their 'semi-stable models', defined to handle situations where p-stable models do not exist. However, this approach works by translating (potentially problematic) disjunctive logic programs into positive disjunctive logic programs, and then by treating semantically the resulting restricted type of programs. As a consequence, the approach by Sakama and Inoue cannot satisfactorily solve problems such as the question of undefined literals, and their treatment of this problem is local and program dependent. Since our proposal does not use translations, it is conservative in the sense that the semi-stable extensions coincide with the stable extensions in the cases where there also exists at least one. To sum up, none of the formalisms that intend to offer paraconsistent semantics for logic programs in the literature simultaneously guarantee non-interference, crash resistance and backwards compatibility.

In our current treatment of semi-stable semantics for logic programming, we have taken the path of applying an argumentation interpretation of logic programming. One can ask the question whether it is also possible to apply semi-stable semantics directly, in native logic programming terms, without going through the 'detour' of formal argumentation theory. Although we do not yet have a definite answer to this question, our hypothesis is that the three-valued stable model

semantics [47] could be a good starting point. In [55] it is proved that the entailment yielded by the three-valued stable model semantics is equivalent to the entailment yielded under complete semantics of the associated argumentation framework (using essentially the same way of constructing the argumentation framework as in Definition 12 of the current article). Hence, what is called three-valued stable semantics in logic programming coincides with what is called complete semantics in formal argumentation [55]. This result is of particular interest, since complete semantics can be seen as the basis to describe various other well known argumentation semantics [10, 16], just like the three-valued stable model semantics can be used as a basis to describe various other well known logic programming semantics. For instance, as was discussed at the end of Section 3.2, preferred labellings can be described as the complete labellings with maximal `in`, the grounded labelling as the (unique) complete labelling with maximal `undec` and stable labellings as complete labellings without `undec`. Similarly, regular models [32] can be described as three-valued stable models with maximal `in`,[2] the well-founded model can be described as the three-valued stable model with maximal `undec` and traditional (two-valued) stable models can be described as three-valued stable models without `undec`. Using the correspondence between three-valued stable model semantics (logic programming) and complete semantics (argumentation) as a basis, one then obtains not only the existing correspondences between the well-founded semantics (logic programming) and grounded semantics (argumentation) [26], and between the stable model semantics (logic programming) and stable semantics (argumentation) [26], but we have recently also been able to obtain the correspondence between regular semantics (logic programming) and preferred semantics (argumentation).

The above-described correspondences between logic programming semantics and argumentation semantics lead to the hopeful expectation that semi-stable semantics for argumentation can be expressed equivalently as selecting the three-valued stable models with minimal `undec`. Providing a formal proof, however, seems to be significantly more difficult than for any of the other correspondences, for reasons that are beyond the scope of the current article. However, even if no formal correspondence can be found, the approach of selecting the three-valued stable models with minimal `undec` seems to yield properties that are very close to those of semi-stable semantics. We hope to be able to report more on this in the near future.

5 Applying Semi-stable Semantics to Default Logic

In this section, we provide an overview of default logic and provide an alternative that satisfies non-interference and crash resistance, while at the same time remaining backward compatible with Reiter's original account of default logic. Our approach will be to apply semi-stable semantics to the argumentation interpretation of default logic.

5.1 Preliminaries

In order to simplify the discussion, we restrict ourselves to the propositional variant of default logic.

DEFINITION 15
A *default* d is an expression $p:j_1,\ldots,j_n/c$ $(n \geq 0)$ where p (the *prerequisite*, $pre(d)$), $j_1,\ldots,j_n$ (the *justification*, $jus(d)$) and c (the *consequent*, $cons(d)$) are propositional formulas. A default is called

[2] We have taken the liberty to refer to the truth values of three-valued stable models as `in`, `out` and `undec`, instead of using Przymusinski's original notation of `t`, `f` and `u`.

normal iff $n=1$ and $j_1=c$. A default is called *semi-normal* iff $j_i=c$ for some $1\leq i\leq n$. A *default theory* T is a pair $(\mathcal{W},\mathcal{D})$ where $\mathcal{W}$ is a finite set of propositional formulas and $\mathcal{D}$ is a finite set of defaults. A default theory is called *normal* iff every default in $\mathcal{D}$ is normal. A default theory is called *semi-normal* iff every default in $\mathcal{D}$ is semi-normal. A default theory is called *consistent* iff $\mathcal{W}$ is consistent.

In the following definition, $Cn(E)$ stands for the propositional consequences of the set of propositions E. That is: $Cn(E)=\{p\mid E\vDash p\}$.

DEFINITION 16 ([48])
Let $T=(\mathcal{W},\mathcal{D})$ be a default theory and E be a set of formulas. Let $i\geq 0: E_0=\mathcal{W}$ and for $i\geq 0$: $E_{i+1}=Cn(E_i)\cup\{c\mid (p:j_1,\ldots,j_n/c)\in\mathcal{D}$ where $p\in E_i$ and $\neg j_1,\ldots,\neg j_n\notin E\}$.
E is a *default extension* of $(\mathcal{W},\mathcal{D})$ iff $E=\cup_{i=0}^{\infty}E_i$.

We write $\mathcal{E}$ for the set of extensions of a given default theory.

There are several possible interpretations of default logic in terms of formal argumentation. In this section, we treat two of such interpretations. In the first interpretation, an argument is seen as a sequence of defaults. In the second interpretation, an argument is seen as a set of trees of defaults. Both interpretations can be used to model the original version of default logic, which in essence implements stable semantics. However, it is the tree-based interpretation that is most suited for changing the semantics of default logic from stable to semi-stable, since it will allow us to apply results of abstract argumentation (in particular Theorem 7) to prove non-interference and crash resistance.

In order to obtain a formalism that satisfies non-interference and crash resistance, and that is backward compatible with standard default logic, we need to take the tree-based interpretation of default logic, change the semantics from stable to semi-stable and rule out all inconsistent arguments. For semi-normal consistent default theories, this approach will then satisfy all three postulates. For our purposes, the sequence-based interpretation of default logic serves merely as a bridge between Reiter's original definition of default logic and our tree-based interpretation.

We first define the sequence-based interpretation of default logic.

DEFINITION 17
A *sequence-based argument* A under default theory $(\mathcal{W},\mathcal{D})$ is a sequence of defaults $[d_1,\ldots,d_n]$ $(n\geq 0)$ where $d_i\neq d_j$ whenever $i\neq j$, such that for each d_i $(1\leq i\leq n)$ it holds that $\mathcal{W}\cup\{cons(d_1),\ldots,cons(d_{i-1})\}\vDash pre(d_i)$. The set of conclusions $concs(A)$ of argument A is defined as $Cn(\mathcal{W}\cup\{cons(d)\mid d$ is a default in $A\})$. Let $A=[d_1,\ldots,d_n]$ $(n\geq 0)$ and $A'=[d'_1,\ldots,d'_m]$ $(m\geq 1)$ be two arguments under default theory $(\mathcal{W},\mathcal{D})$. We say that A *attacks* A' iff A' contains a default d'_i $(1\leq i\leq m)$ such that $\neg j\in concs(A)$ for some $j\in jus(d'_i)$.

If $\mathcal{A}rgs$ is a set of sequence-based arguments, then we write $concs(\mathcal{A}rgs)$ for $Cn(\mathcal{W}\cup\{concs(A)\mid A\in\mathcal{A}rgs\})$. We write Ar_T^{seq} for the set of all sequence-based arguments under $T=(\mathcal{W},\mathcal{D})$ and att_T^{seq} for the attack relation under T.

DEFINITION 18
Let $T=(\mathcal{W},\mathcal{D})$ be a default theory. We define the consequences of T under sequence-based interpretation using stable semantics $Cn_{\text{stdl}_{\text{seq}}}(T)$ as $\{concs(\mathcal{A}rgs)\mid\mathcal{A}rgs$ is a stable extension of $(Ar_T^{\text{seq}},att_T^{\text{seq}})\}$.

The following theorem states that the sequence-based argumentation interpretation of default logic under stable semantics is equivalent with Reiter's original notion of default logic. The equivalence

between argumentation and default logic has also been observed in [26]; however, since our argument form is slightly different than in [26] (which seems to be more in line with the assumption-based approach of [7]), we have added a separate proof in this article.

THEOREM 13
Let $T=(\mathcal{W},\mathcal{D})$ be a default theory and $\mathcal{E}$ be its set of default extensions. It holds that $Cn_{\text{stdl}_{\text{seq}}}(T)=\mathcal{E}$.

PROOF.
'$\subseteq$': Let $E\in Cn_{\text{stdl}_{\text{seq}}}(T)$. This means there exists a stable extension $\mathcal{A}rgs$ of sequence-based arguments under T such that $concs(\mathcal{A}rgs)=E$. We now show that E is also a default extension in the sense of Definition 16. So we need to prove that:

(1) $\cup_{i=0}^{\infty}E_i\subseteq E$. Let $e\notin E$. Then $\mathcal{A}rgs$ contains no argument with conclusion e. The fact that $\mathcal{A}rgs$ is a stable extension then implies that for each argument A with conclusion e, A is attacked by an argument in $\mathcal{A}rgs$. Therefore, the derivation of e will be blocked in $\cup_{i=0}^{\infty}E_i$. So $e\notin\cup_{i=0}^{\infty}E_i$.
(2) $E\subseteq\cup_{i=0}^{\infty}E_i$. Let $e\in E$. Then there exists an argument (say A) for e. That is, $e\in concs(A)$. The fact that $A\in\mathcal{A}rgs$ implies that A is not attacked by any argument in $\mathcal{A}rgs$. Therefore, there will be an $i\geq 0$ such that $e\in E_i$, so $e\in\cup_{i=0}^{\infty}E_i$.

'$\supseteq$': Let $E\in\mathcal{E}$. This means that E is a default extension of $(\mathcal{W},\mathcal{D})$. We now prove that there exists a stable extension $\mathcal{A}rgs$ of the argumentation framework $(Ar_T^{\text{seq}},att_T^{\text{seq}})$ such that $concs(\mathcal{A}rgs)=E$. Let $\mathcal{A}rgs$ be the set of all arguments that are not 'attacked' by E. That is, $A\in Ar_T^{\text{seq}}$ is in $\mathcal{A}rgs$ iff A does not contain a default d with $e\in jus(d)$ for some $\neg e\in E$. It holds that the conclusions of $\mathcal{A}rgs$ correspond with $\cup_{i=0}^{\infty}E_i$, so from the fact that $E=\cup_{i=0}^{\infty}E_i$ it follows that $concs(\mathcal{A}rgs)=E$. We now show that $\mathcal{A}rgs$ is a stable extension. First of all, $\mathcal{A}rgs$ is conflict-free. Suppose $\exists A,B\in\mathcal{A}rgs:A$ attacks B. Since A and B are not 'attacked' by E, it holds that $concs(A)\not\subseteq E$. But this contradicts with $concs(\mathcal{A}rgs)=E$. Secondly, $\mathcal{A}rgs$ attacks each argument C that is not in $\mathcal{A}rgs$. Let $C\in Ar_T^{\text{seq}}$ such that $C\notin\mathcal{A}rgs$. Then C is 'attacked' by E. But since $concs(\mathcal{A}rgs)=E$, it means that C is also attacked by $\mathcal{A}rgs$. ■

We now define the tree-based interpretation of default logic.

DEFINITION 19
A *pre-argument* A under a default theory $(\mathcal{W},\mathcal{D})$ is a (possibly empty) tree of defaults such that for each default d it holds that its set of children $\{d_1,\ldots,d_n\}$ $(n\geq 0)$ is a minimal set such that $\mathcal{W}\cup\{cons(d_1),\ldots,cons(d_n)\}\vDash pre(d)$. Furthermore, every default is allowed to occur at most once in each branch of the tree.

Let A be a pre-argument. We define $concs(A)$ as $Cn(\mathcal{W}\cup\{cons(d)\,|\,d$ is a default in $A\})$. Let S be a set of pre-arguments. We define $concs(S)$ as $Cn(\cup\{concs(A)\,|\,A\in S\})$.

A *tree-based argument* is either of the form $\{A\}$ where A is a pre-argument, or a minimal set of pre-arguments $\{A_1,\ldots,A_n\}$ such that $concs(\{A_1,\ldots,A_n\})$ contains $\neg j$ with $j\in jus(d)$ for some $d\in\mathcal{D}$.

Let A and B be tree-based arguments under $(\mathcal{W},\mathcal{D})$. We say that A *attacks* B iff $\neg j\in concs(A)$ with $j\in jus(d)$ for some d occurring in B.

As an example of how Definition 19 is applied, consider the default theory $T_1=(\mathcal{W}_1,\mathcal{D}_1)$ with $\mathcal{W}_1=\emptyset$ and $\mathcal{D}_1=\{\top:\top/a;\ a:\neg b/c;\ a:\neg c,\neg e/d;\ \top:\neg d/e\}$. Based on T_1, one can construct four tree-based arguments that attack each other according to the argumentation framework of Figure 6.

If $\mathcal{A}rgs$ is a set of tree-based arguments, then we write $concs(\mathcal{A}rgs)$ for $Cn(\cup\{concs(A)\,|\,A\in\mathcal{A}rgs\})$. We write Ar_T^{tree} for the set of all tree-based arguments under $T=(\mathcal{W},\mathcal{D})$ and att_T^{tree} for the attack relation under T.

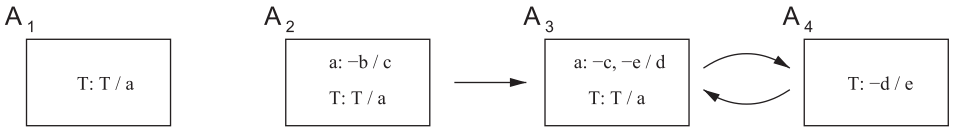

FIGURE 6. The argumentation framework associated with default theory T_1.

DEFINITION 20
Let $T=(\mathcal{W},\mathcal{D})$ be a default theory. We define the consequences of T under tree-based interpretation using stable semantics as $Cn_{\text{stdl}_{\text{tree}}}(T)=\{concs(\mathcal{A}rgs)\mid \mathcal{A}rgs \text{ is a stable extension of } (Ar_T^{\text{tree}},att_T^{\text{tree}})$.

In the argumentation framework of Figure 6, there exists just one stable extension $\{A_1,A_2,A_4\}$ with associated conclusions $Cn(\{a,c,e\})$. Hence, $Cn_{\text{stable}}(T_1)=\{Cn(\{a,c,e\})\}$.

Under stable semantics, it does not matter for the entailment whether one applies sequence-based or tree-based arguments, as is stated by the following theorem.

LEMMA 4
Let $T=(\mathcal{W},\mathcal{D})$ be a default theory. It holds that $Cn_{\text{stdl}_{\text{seq}}}(T)=Cn_{\text{stdl}_{\text{tree}}}(T)$.

PROOF.
'$\subseteq$': Let $E\in Cn_{\text{stdl}_{\text{seq}}}(T)$. This means that there exists a stable extension $\mathcal{A}rgs^{\text{seq}}$ of $(Ar_T^{\text{seq}},att_T^{\text{seq}})$ such that $concs(\mathcal{A}rgs^{\text{seq}})=E$. We now prove that there also exists a stable extension $\mathcal{A}rgs^{\text{tree}}$ of $(Ar_T^{\text{tree}},att_T^{\text{tree}})$ with $concs(\mathcal{A}rgs^{\text{tree}})=E$. Let $\mathcal{A}rgs^{\text{tree}}$ be the set of all possible tree-based arguments that can be constructed using the defaults in $\mathcal{A}rgs^{\text{seq}}$. We now prove that $\mathcal{A}rgs^{\text{tree}}$ is a stable extension.

conflict freeness: Let $A,B\in\mathcal{A}rgs^{\text{tree}}$ such that A attacks B. Then there also exists two arguments $A',B'\in Ar_T^{\text{seq}}$ where A' contains the same defaults as A and B' contains the same defaults as B. These arguments are not attacked by $\mathcal{A}rgs^{\text{seq}}$ (this is because their defaults occur in $\mathcal{A}rgs^{\text{seq}}$) so from the fact that $\mathcal{A}rgs^{\text{seq}}$ is a stable extension, it follows that $A',B'\in\mathcal{A}rgs^{\text{seq}}$. However, since $concs(A)=concs(A')$ and $defaults(B)=defaults(B')$, it follows that A' attacks B' under $(Ar_T^{\text{seq}},att_T^{\text{seq}})$. So $\mathcal{A}rgs^{\text{seq}}$ would not be conflict-free. Contradiction.

attacking any argument not in it: Let $B\in Ar_T^{\text{tree}}$ such that $B\notin\mathcal{A}rgs^{\text{tree}}$. Let B' be an argument of Ar_T^{seq} that contains the same defaults as B. It holds that $B'\notin\mathcal{A}rgs^{\text{seq}}$. This can be seen as follows. Suppose $B'\in\mathcal{A}rgs^{\text{seq}}$. Then B could be constructed using the defaults in $\mathcal{A}rgs^{\text{seq}}$ so $B\in\mathcal{A}rgs^{\text{tree}}$. Contradiction. So $B'\notin\mathcal{A}rgs^{\text{seq}}$. From the fact that $\mathcal{A}rgs^{\text{seq}}$ is a stable extension and $B'\notin\mathcal{A}rgs^{\text{seq}}$ it follows that $\mathcal{A}rgs^{\text{seq}}$ contains an argument A' that attacks B'. This implies that A' has a conclusion $\neg j$, whereas B' contains a default d with $j\in jus(d)$. From the fact that $A'\in\mathcal{A}rgs^{\text{seq}}$ it follows that one can construct a tree-based argument A, using the defaults from $\mathcal{A}rgs^{\text{seq}}$, that has conclusion $\neg j$ and therefore attacks B. So, $\mathcal{A}rgs^{\text{tree}}$ contains an argument A that attacks B.

Now that we have proved that $\mathcal{A}rgs^{\text{tree}}$ is a stable extension, the next thing to prove is that $concs(\mathcal{A}rgs^{\text{tree}})=concs(\mathcal{A}rgs^{\text{seq}})$.

'$concs(\mathcal{A}rgs^{\text{tree}})\subseteq concs(\mathcal{A}rgs^{\text{seq}})$': Let $c\in concs(\mathcal{A}rgs^{\text{tree}})$. Let A be the argument in Ar^{seq} that contains all defaults of $\mathcal{A}rgs^{\text{tree}}$. This argument is an element of $\mathcal{A}rgs^{\text{seq}}$, because $\mathcal{A}rgs^{\text{seq}}$ does not 'attack' any of its defaults, and hence $\mathcal{A}rgs^{\text{seq}}$ does not attack A. From the fact that $\mathcal{A}rgs^{\text{seq}}$

is a stable extension it then follows that $A \in \mathcal{A}rgs^{\text{seq}}$. It then follows that $c \in concs(A)$ and that therefore $c \in concs(\mathcal{A}rgs^{\text{seq}})$.

'$concs(\mathcal{A}rgs^{\text{tree}}) \supseteq concs(\mathcal{A}rgs^{\text{seq}})$': For each default occurring in $\mathcal{A}rgs^{\text{seq}}$, there exists an argument in $\mathcal{A}rgs^{\text{tree}}$ that contains this default. This means that $defaults(\mathcal{A}rgs^{\text{seq}}) \subseteq defaults(\mathcal{A}rgs^{\text{tree}})$. From this, it follows that $concs(\mathcal{A}rgs^{\text{seq}}) \subseteq concs(\mathcal{A}rgs^{\text{tree}})$.

'$\supseteq$': Let $E \in Cn_{\text{stdl}_{\text{tree}}}(T)$. This means there exists a stable extension $\mathcal{A}rgs^{\text{tree}}$ under $(Ar_T^{\text{tree}}, att_T^{\text{tree}})$ such that $concs(\mathcal{A}rgs^{\text{tree}}) = E$. We now prove that there also exists a stable extension $\mathcal{A}rgs^{\text{seq}}$ of $(Ar_T^{\text{seq}}, att_T^{\text{seq}})$ with $concs(\mathcal{A}rgs^{\text{seq}}) = E$. Let $\mathcal{A}rgs^{\text{seq}}$ be the set of all possible sequence-based arguments that can be constructed using the defaults in $\mathcal{A}rgs^{\text{tree}}$. We now prove that $\mathcal{A}rgs^{\text{seq}}$ is a stable extension.

conflict freeness: Let $A, B \in \mathcal{A}rgs^{\text{seq}}$ such that A attacks B. This means that A has a conclusion $\neg j$ and B contains a default d, with $j \in jus(d)$. Let A' be an argument under Ar_T^{tree} with conclusion $\neg j$ (Definition 19 makes sure such an argument exists). Since all defaults in A' are in $\mathcal{A}rgs^{\text{tree}}$ and $\mathcal{A}rgs^{\text{tree}}$ is a stable extension, it follows that $\mathcal{A}rgs^{\text{tree}}$ contains an argument (say B') that contains d. But then A' attacks B' and $\mathcal{A}rgs^{\text{tree}}$ would not be conflict-free. Contradiction.

attacking every argument not in it: Let $B \in Ar_T^{\text{seq}}$ be an argument that is not in $\mathcal{A}rgs^{\text{seq}}$. Then B contains at least one default d, which does not occur in $\mathcal{A}rgs^{\text{tree}}$. Let B' be an argument in Ar_T^{tree} that contains only defaults from B, including d. The fact that such an argument exists follows from the fact that B exists. From the fact that d does not occur in $\mathcal{A}rgs^{\text{tree}}$, it follows that $B' \notin \mathcal{A}rgs^{\text{tree}}$. From the fact that $\mathcal{A}rgs^{\text{tree}}$ is a stable extension, it follows that $\mathcal{A}rgs^{\text{tree}}$ contains an argument (say A') that attacks B'. So A' has a conclusion $\neg j$ and B' contains a default d' (possibly equal to d) with $j \in jus(d')$. Let A be an argument in Ar_T^{seq} that contains the same defaults as A'. It then follows that A also has a conclusion $\neg j$. And since B' only contains defaults from B, it follows that $defaults(B') \subseteq defaults(B)$ so $d' \in defaults(B)$. This means that A attacks B.

Now that we have proved that $\mathcal{A}rgs^{\text{seq}}$ is a stable extension, the next thing to prove is that $concs(\mathcal{A}rgs^{\text{seq}}) = concs(\mathcal{A}rgs^{\text{tree}})$.

'$concs(\mathcal{A}rgs^{\text{seq}}) \subseteq concs(\mathcal{A}rgs^{\text{tree}})$': Let $c \notin concs(\mathcal{A}rgs^{\text{tree}})$. Then c is not a consequence of $\cup\{concs(A) \mid A \in \mathcal{A}rgs^{\text{tree}}\}$. Therefore, c is not a consequence of $\mathcal{W} \cup \{cons(d) \mid d$ occurs in some $A \in \mathcal{A}rgs^{\text{tree}}\}$. And since the defaults that occur in $\mathcal{A}rgs^{\text{tree}}$ are the same as that occur in $\mathcal{A}rgs^{\text{seq}}$, it follows that c is not a consequence of $\mathcal{W} \cup \{cons(d) \mid d$ occurs in some $A \in \mathcal{A}rgs^{\text{seq}}\}$, so c is not a consequence of $\mathcal{W} \cup \{concs(A) \mid A \in \mathcal{A}rgs^{\text{seq}}\}$ so $c \notin concs(\mathcal{A}rgs^{\text{seq}})$.

'$concs(\mathcal{A}rgs^{\text{seq}}) \supseteq concs(\mathcal{A}rgs^{\text{tree}})$': Let A be the argument in Ar_T^{seq} containing all defaults from $\mathcal{A}rgs^{\text{tree}}$. This argument is an element of $\mathcal{A}rgs^{\text{seq}}$, because it is simply one of the possible arguments that can be constructed using the defaults from $\mathcal{A}rgs^{\text{tree}}$. It holds that $concs(\mathcal{A}rgs^{\text{seq}}) \supseteq concs(A)$. From the fact that $concs(A) = concs(\mathcal{A}rgs^{\text{tree}})$, it then follows that $concs(\mathcal{A}rgs^{\text{seq}}) \supseteq concs(\mathcal{A}rgs^{\text{tree}})$. ■

From the fact that $Cn_{\text{stdl}_{\text{seq}}}$ is equivalent with Reiter's original definition of default logic (Theorem 13) and the fact that the $Cn_{\text{stdl}_{\text{tree}}}$ is equivalent with $Cn_{\text{stdl}_{\text{seq}}}$ (Lemma 4), it follows that $Cn_{\text{stdl}_{\text{tree}}}$ is equivalent to Reiter's original definition of default logic.

Since $Cn_{\text{stdl}_{\text{seq}}}$ is equivalent with $Cn_{\text{stdl}_{\text{tree}}}$, we sometimes simply write Cn_{stdl} without explicitly mentioning whether we refer to tree-based or sequence-based argumentation.

The following definition aims at filtering out the inconsistent arguments from the argumentation framework associated with a default theory.

DEFINITION 21
Let $(\mathcal{W},\mathcal{D})$ be a default theory and $(Ar_T^{\text{tree}}, att_T^{\text{tree}})$ be its associated argumentation framework. We define Ar_T^{ctree} as $\{A \mid A \in Ar_T^{\text{tree}} \text{ and } \bot \notin concs(A)\}$ and att_T^{ctree} as $att_T^{\text{tree}} \cap (Ar_T^{\text{ctree}} \times Ar_T^{\text{ctree}})$.

The first thing to be proved is that for semi-normal default theories under stable semantics, the entailment does not change when inconsistent arguments are ruled out.

DEFINITION 22
A tree-based structure is a set of pre-arguments $\{A_1,\ldots,A_n\}$ $(n \geq 1)$. If S_1 and S_2 are tree-based structures, then we say that S_1 is a substructure of S_2, notated as $S_1 \sqsubseteq S_2$ iff for each $A \in S_1$ there exists an $A' \in S_2$ such that A is a subtree of A'. S_1 is a strict substructure of S_2, notated as $S_1 \sqsubset S_2$, iff $S_1 \sqsubseteq S_2$ and $S_1 \neq S_2$.

LEMMA 5
Let $(\mathcal{W},\mathcal{D})$ be a consistent semi-normal default theory. It holds that $\mathcal{A}rgs$ is a stable extension of $(Ar_T^{\text{tree}}, att_T^{\text{tree}})$ iff $\mathcal{A}rgs$ is a stable extension of $(Ar_T^{\text{ctree}}, att_T^{\text{ctree}})$.

PROOF.
'$\Longrightarrow$': Let $\mathcal{A}rgs$ be a stable extension of $(Ar_T^{\text{tree}}, att_T^{\text{tree}})$. Then $\mathcal{A}rgs \subseteq Ar_T^{\text{tree}}$, $\mathcal{A}rgs$ is conflict-free and $\mathcal{A}rgs$ attacks each argument in $Ar_T^{\text{tree}} \setminus \mathcal{A}rgs$. We observe that any inconsistent argument $A \in Ar_T^{\text{tree}}$ has to contain at least one default (it cannot be empty), because $\mathcal{W}$ is consistent. Therefore, any inconsistent argument is also self-attacking. From the fact that $\mathcal{A}rgs$ is conflict-free, it then follows that $\mathcal{A}rgs$ does not contain any inconsistent arguments (which would be self-attacking). Therefore, $\mathcal{A}rgs \subseteq Ar_T^{\text{ctree}}$. From the fact that $\mathcal{A}rgs$ attacks each argument in $Ar_T^{\text{tree}} \setminus \mathcal{A}rgs$ and that $Ar_T^{\text{ctree}} \subseteq Ar_T^{\text{tree}}$, it follows that $\mathcal{A}rgs$ attacks each argument in $Ar_T^{\text{ctree}} \setminus \mathcal{A}rgs$. Furthermore, the facts that $\mathcal{A}rgs$ is conflict-free under $(Ar_T^{\text{tree}}, att_T^{\text{tree}})$ and that $att_T^{\text{ctree}} \subseteq att_T^{\text{tree}}$ imply that $\mathcal{A}rgs$ is also conflict-free under $(Ar_T^{\text{ctree}}, att_T^{\text{ctree}})$. Therefore, $\mathcal{A}rgs$ is a stable extension of $(Ar_T^{\text{ctree}}, att_T^{\text{ctree}})$.
'$\Longleftarrow$': Let $\mathcal{A}rgs$ be a stable extension of $(Ar_T^{\text{ctree}}, att_T^{\text{ctree}})$. Then $\mathcal{A}rgs \subseteq Ar_T^{\text{ctree}}$, $\mathcal{A}rgs$ is conflict-free and $\mathcal{A}rgs$ attacks each argument in $Ar_T^{\text{ctree}} \setminus \mathcal{A}rgs$. From the fact that $\mathcal{A}rgs \subseteq Ar_T^{\text{ctree}}$ and that $Ar_T^{\text{ctree}} \subseteq Ar_T^{\text{tree}}$, it follows that $\mathcal{A}rgs \subseteq Ar_T^{\text{tree}}$. From the fact that $\mathcal{A}rgs$ is conflict-free under $(Ar_T^{\text{ctree}}, att_T^{\text{ctree}})$, it follows that $\mathcal{A}rgs$ is still conflict-free under $(Ar_T^{\text{tree}}, att_T^{\text{tree}})$. We now prove that $\mathcal{A}rgs$ attacks each argument in $Ar_T^{\text{tree}} \setminus \mathcal{A}rgs$. Let $A \in Ar_T^{\text{tree}} \setminus \mathcal{A}rgs$. We distinguish two cases:

(1) A is a consistent argument ($\bot \notin concs(A)$). Then $A \in Ar_T^{\text{ctree}}$. Since $\mathcal{A}rgs$ attacks each argument in $Ar_T^{\text{ctree}} \setminus \mathcal{A}rgs$ it follows that S attacks A.
(2) A is an inconsistent argument ($\bot \in Concs(A)$). This implies that $\mathcal{W} \cup \{cons(d) \mid d \text{ is in } A\} \vDash \bot$. From the fact that $\mathcal{W}$ is consistent, it follows that A contains at least one default. Let A' be a maximal substructure of A that is still consistent. That is: $A' \sqsubseteq A$ and $\forall A'' : (A' \sqsubset A'' \sqsubseteq A \rightarrow \bot \in concs(A''))$. Let A'' be such that $A' \sqsubset A'' \sqsubseteq A$ where A'' contains exactly one additional default not contained in A'. It then holds that A'' is inconsistent. Let $d_1,\ldots,d_n$ be the defaults of A'. Let d_{n+1} be the additional default in A''. It then holds that $\mathcal{W} \cup \{cons(d_1),\ldots,cons(d_n)\}$ is consistent, but $\mathcal{W} \cup \{cons(d_1),\ldots,cons(d_n),cons(d_{n+1})\}$ is inconsistent. It then follows that $\mathcal{W} \cup \{cons(d_1),\ldots,cons(d_n)\} \vDash \neg cons(d_{n+1})$. Since T is a semi-normal default theory, it holds that the consequent of d_{n+1} is also contained in the justification of d_{n+1}. That is, there exists a $j \in jus(d_{n+1})$ such that $\mathcal{W} \cup \{cons(d_1),\ldots,cons(d_n)\} \vDash \neg j$. From the fact that A'' is a substructure of A it follows that A also contains default d_{n+1}. Therefore, A' attacks A. Since A' is consistent

(and that it attacks A), it holds that $A' \in Ar_T^{\text{ctree}}$. From the fact that $\mathcal{A}rgs$ is a stable extension of $(Ar_T^{\text{ctree}}, att_T^{\text{ctree}})$, it then follows that either $A' \in \mathcal{A}rgs$ or $\mathcal{A}rgs$ attacks A'. In the first case ($A' \in \mathcal{A}rgs$), it holds that $\mathcal{A}rgs$ attacks A (since A' attacks A). In the second case ($\mathcal{A}rgs$ attacks A'), it holds that $\mathcal{A}rgs$ attacks A (since A' is a subargument of A). In both cases, $\mathcal{A}rgs$ attacks A.

■

5.2 *Semi-stable semantics for default logic*

We are now ready to apply semi-stable semantics to default logic, for which we use the tree-based argument form in which inconsistent arguments have been removed.

DEFINITION 23
Let $T = (\mathcal{W}, \mathcal{D})$ be a default theory. We define $Cn_{\text{ssdl}}(\mathcal{W}, \mathcal{D})$ as $\{concs(\mathcal{A}rgs) \mid \mathcal{A}rgs$ is a semi-stable extension of $(Ar_T^{\text{ctree}}, att_T^{\text{ctree}})\}$.

We now prove that for consistent semi-normal default theories, Cn_{ssdl} satisfies the postulate of backward compatibility with standard default logic which (by Theorem 13 and Lemma 4) is equivalent with Cn_{stdl}.

THEOREM 14
For consistent semi-normal default theories, Cn_{ssdl} is backward compatible with Cn_{stdl}.

PROOF. Let $T = (\mathcal{W}, \mathcal{D})$ be a consistent semi-normal default theory that is not contaminating. The fact that it is not contaminating implies that it has at least one default extension. Theorem 13 and Lemma 4 then imply that its associated argumentation framework $(Ar_T^{\text{tree}}, att_T^{\text{tree}})$ has at least one stable extension. Since, by Lemma 5, the stable extensions are unchanged when inconsistent arguments are removed, it holds that $(Ar_T^{\text{ctree}}, att_T^{\text{ctree}})$ also has at least one stable extension. From Theorem 5 it then follows that the stable extensions of $(Ar_T^{\text{ctree}}, att_T^{\text{ctree}})$ are the same as its semi-stable extensions. This, together with the fact that the conclusions of the stable extensions of $(Ar_T^{\text{ctree}}, att_T^{\text{ctree}})$ are the same as the conclusions of the stable extensions of $(Ar_T^{\text{tree}}, att_T^{\text{tree}})$ (Lemma 5) implies that the conclusions of the stable extensions of $(Ar_T^{\text{tree}}, att_T^{\text{tree}})$ are the same as the conclusions of the semi-stable extensions of $(Ar_T^{\text{ctree}}, att_T^{\text{ctree}})$, so we have that $Cn_{\text{stdl}} = Cn_{\text{ssdl}}$. ■

For consistent semi-normal default theories, not only is Cn_{ssdl} backward compatible with Cn_{stdl}, it also satisfies crash resistance and non-interference. For this, we first need a lemma (Lemma 6) that allows us to apply to default logic the results for non-interference for abstract argumentation. It should be mentioned that Lemma 6 only holds for the tree-based interpretation of default logic, in which inconsistent arguments have been ruled out.

LEMMA 6
Let $T_1 = (\mathcal{W}_1, \mathcal{D}_1)$ and $T_2 = (\mathcal{W}_2, \mathcal{D}_2)$ be syntactically disjoint consistent semi-normal default theories and let $T_3 = (\mathcal{W}_1 \cup \mathcal{W}_2, \mathcal{D}_1 \cup \mathcal{D}_2)$. It holds that $Ar_{T_3}^{\text{ctree}} = Ar_{T_1}^{\text{ctree}} \cup Ar_{T_2}^{\text{ctree}}$ and $att_{T_3}^{\text{ctree}} = att_{T_1}^{\text{ctree}} \cup att_{T_2}^{\text{ctree}}$.

PROOF. Naturally, each argument that can be constructed under T_1 can also be constructed under T_3, and every argument that can be constructed under T_2 can also be constructed under T_3. Therefore, it holds that $Ar_{T_1}^{\text{ctree}} \subseteq Ar_{T_3}^{\text{ctree}}$ and $Ar_{T_2}^{\text{ctree}} \subseteq Ar_{T_3}^{\text{ctree}}$, so $Ar_{T_1}^{\text{ctree}} \cup Ar_{T_2}^{\text{ctree}} \subseteq Ar_{T_3}^{\text{ctree}}$. We now prove that $Ar_{T_3}^{\text{ctree}} \subseteq Ar_{T_1}^{\text{ctree}} \cup Ar_{T_2}^{\text{ctree}}$. Let $A \in Ar_{T_3}^{\text{ctree}}$. We distinguish three cases.

(1) $defaults(A) \subseteq \mathcal{D}_1$. In that case, $A \in Ar_{T_1}^{\text{ctree}}$, so $A \in Ar_{T_1}^{\text{ctree}} \cup Ar_{T_2}^{\text{ctree}}$.
(2) $defaults(A) \subseteq \mathcal{D}_2$. In that case, $A \in Ar_{T_2}^{\text{ctree}}$, so $A \in Ar_{T_1}^{\text{ctree}} \cup Ar_{T_2}^{\text{ctree}}$.

(3) $defaults(A) \not\subseteq \mathcal{D}_1$ and $defaults(A) \not\subseteq \mathcal{D}_2$. In that case, A contains at least one default from $\mathcal{D}_1$ and at least one default from $\mathcal{D}_2$. We will now show that this cannot be the case. First of all, we observe that A cannot have any pre-argument that contains a default from $\mathcal{D}_1$ and a default from $\mathcal{D}_2$. Suppose A contains a pre-argument A' with $d_1 \in \mathcal{D}_1$ as one of the children of $d_2 \in \mathcal{D}_2$ (or vice versa). Then, the fact that T_1 and T_2 are syntactically disjoint and the fact that A' is consistent (since A is consistent) imply that the consequent of d_1 does not play any role in the derivation of the prerequisite of d_2. Therefore, d_1 is an unneccesary child of d_2, so the set of children of d_2 is not minimal, which conflicts with the definition of tree-based arguments (Definition 19). So each pre-argument of A consists either entirely of defaults from $\mathcal{D}_1$ or entirely of defaults from $\mathcal{D}_2$. The next thing to prove is that either all pre-arguments of A contain only defaults from $\mathcal{D}_1$ or all pre-arguments of A contain only defaults from $\mathcal{D}_2$. Suppose this is not the case. Then A contains a pre-argument A_1 consisting of only defaults from $\mathcal{D}_1$ and a pre-argument A_2 consisting of only defaults from $\mathcal{D}_2$. The fact that A consists of more than one pre-argument means that there must exist a default d in $\mathcal{D}_1 \cup \mathcal{D}_2$ whose justification is attached by A; i.e. $\neg j \in concs(A)$ for some $j \in jus(d)$. Assume without loss of generality that $d \in \mathcal{D}_1$ (the case of $d \in \mathcal{D}_2$ goes similar). Since T_1 and T_2 are syntactically disjoint and A is consistent, then it implies that A_2 does not play any role in attacking default d (in deriving $\neg j$). But then A would have a redundant pre-argument, which conflicts with the minimality requirement in Definition 19.

From the earlier observed fact that $Ar_{T_1}^{\text{ctree}} \cup Ar_{T_2}^{\text{ctree}} \subseteq Ar_{T_3}^{\text{ctree}}$ and from the newly observed fact that $Ar_{T_3}^{\text{ctree}} \subseteq Ar_{T_1}^{\text{ctree}} \cup Ar_{T_2}^{\text{ctree}}$, it follows that $Ar_{T_3}^{\text{ctree}} = Ar_{T_1}^{\text{ctree}} \cup Ar_{T_2}^{\text{ctree}}$.
The next thing to prove is that $att_{T_3}^{\text{ctree}} = att_{T_1}^{\text{ctree}} \cup att_{T_2}^{\text{ctree}}$. First of all, we observe that if A attacks B under $(Ar_{T_1}^{\text{ctree}}, att_{T_1}^{\text{ctree}})$ then A also attacks B under $(Ar_{T_3}^{\text{ctree}}, att_{T_3}^{\text{ctree}})$. Similarly, if A attacks B under $(Ar_{T_2}^{\text{ctree}}, att_{T_2}^{\text{ctree}})$, then A also attacks B under $(Ar_{T_3}^{\text{ctree}}, att_{T_3}^{\text{ctree}})$. So it holds that $att_{T_1}^{\text{ctree}} \cup att_{T_2}^{\text{ctree}} \subseteq att_{T_3}^{\text{ctree}}$. We now prove that it also holds that $att_{T_3}^{\text{ctree}} \subseteq att_{T_1}^{\text{ctree}} \cup att_{T_2}^{\text{ctree}}$. Suppose A attacks B under $(Ar_{T_3}^{\text{ctree}}, att_{T_3}^{\text{ctree}})$. We distinguish two possibilities.

(1) $A \in Ar_{T_1}^{\text{ctree}}$. Since A is consistent and T_1 and T_2 are syntactically disjoint, A can only attack B on a default $d \in \mathcal{D}_1$. Therefore, $B \in Ar_{T_1}^{\text{ctree}}$, so A attacks B under $(Ar_{T_1}^{\text{ctree}}, att_{T_1}^{\text{ctree}})$.
(2) $A \in Ar_{T_2}^{\text{ctree}}$. Since A is consistent and T_1 and T_2 are syntactically disjoint, A can only attack B on a default $d \in \mathcal{D}_2$. Therefore, $B \in Ar_{T_2}^{\text{ctree}}$, so A attacks B under $(Ar_{T_2}^{\text{ctree}}, att_{T_2}^{\text{ctree}})$.

From the thus observed property that $att_{T_3}^{\text{ctree}} \subseteq att_{T_1}^{\text{ctree}} \cup att_{T_2}^{\text{ctree}}$, together with the earlier observed fact that $att_{T_1}^{\text{ctree}} \cup att_{T_2}^{\text{ctree}} \subseteq att_{T_3}^{\text{ctree}}$, it follows that $att_{T_3}^{\text{ctree}} = att_{T_1}^{\text{ctree}} \cup att_{T_2}^{\text{ctree}}$. ■

THEOREM 15
For consistent semi-normal default theories, Cn_{ssdl} satisfies non-interference.

PROOF. Let $T_1 = (\mathcal{W}_1, \mathcal{D}_1)$ and $T_2 = (\mathcal{W}_2, \mathcal{D}_2)$ be two syntactically disjoint default theories and let $T_3 = (\mathcal{W}_1 \cup \mathcal{W}_2, \mathcal{D}_1 \cup \mathcal{D}_2)$. In order to prove non-interference, we need to prove that:

- $Cn_{\text{ssdl}}(T_1)_{|\text{atoms}(T_1)} = Cn_{\text{ssdl}}(T_3)_{|\text{atoms}(T_1)}$ and
- $Cn_{\text{ssdl}}(T_2)_{|\text{atoms}(T_2)} = Cn_{\text{ssdl}}(T_3)_{|\text{atoms}(T_2)}$

We only prove the first property (the proof of the second property is similar).
'$\subseteq$': Let $S \in Cn_{\text{ssdl}}(T_1)_{|\text{atoms}(T_1)}$. This means there exists a semi-stable extension $\mathcal{A}rgs$ of $AF_{T_1}^{\text{ctree}}$ such that $concs(\mathcal{A}rgs)_{|\text{atoms}(T_1)} = S$. The fact that semi-stable semantics for abstract argumentation

satisfies non-interference (Theorem 7) means that (given that $AF_{T_1}^{\text{ctree}}$ and $AF_{T_2}^{\text{ctree}}$ are syntactically disjoint):

$$Cn_{\text{semi-stable}}(Ar_{T_1}^{\text{ctree}}, att_{T_1}^{\text{ctree}})_{|Ar_{T_1}^{\text{ctree}}} =$$

$$Cn_{\text{semi-stable}}(Ar_{T_1}^{\text{ctree}} \cup Ar_{T_2}^{\text{ctree}}, att_{T_1}^{\text{ctree}} \cup att_{T_2}^{\text{ctree}})_{|Ar_{T_1}^{\text{ctree}}}$$

From Lemma 6 it then follows $AF_{T_3} = (Ar_{T_1} \cup Ar_{T_2}, att_{T_1} \cup att_{T_2})$ so it holds that:

$$Cn_{\text{semi-stable}}(AF_{T_1}^{\text{ctree}})_{|Ar_{T_1}^{\text{ctree}}} = Cn_{\text{semi-stable}}(AF_{T_3}^{\text{ctree}})_{|Ar_{T_1}^{\text{ctree}}}$$

From the fact that $\mathcal{A}rgs$ is a semi-stable extension of $AF_{T_1}^{\text{ctree}}$ it follows that $\mathcal{A}rgs \in Cn_{\text{semi-stable}}(AF_{T_1}^{\text{ctree}})_{|Ar_{T_1}^{\text{ctree}}}$ so $\mathcal{A}rgs \in Cn_{\text{semi-stable}}(AF_{T_3}^{\text{ctree}})_{|Ar_{T_1}}$, which then implies that there exists a semi-stable extension $\mathcal{A}rgs'$ of $AF_{T_3}^{\text{ctree}}$ such that $\mathcal{A}rgs' \cap Ar_{T_1}^{\text{ctree}} = \mathcal{A}rgs$. The fact that $\mathcal{A}rgs'$ is a semi-stable extension of $AF_{T_3}^{\text{ctree}}$ implies that $concs(\mathcal{A}rgs') = Cn_{\text{ssdl}}(T_3)$, so $concs(\mathcal{A}rgs')_{|\text{atoms}(T_1)} \in Cn_{\text{ssdl}}(T_3)_{|\text{atoms}(T_1)}$. We now prove that $concs(\mathcal{A}rgs')_{|\text{atoms}(T_1)} = S$.

'$\subseteq$': Let $e \in concs(\mathcal{A}rgs')_{|\text{atoms}(T_1)}$. Then $e \in atoms(T_1)$. Let A be an argument in $\mathcal{A}rgs'$ with $e \in concs(A)$. Then $A \in Ar_{T_1}^{\text{ctree}}$, so A *in*$\mathcal{A}rgs$, so $e \in concs(\mathcal{A}rgs)$, so $e \in S$.

'$\supseteq$': Let $e \in S$. Then $e \in concs(\mathcal{A}rgs)$. So there exists an $A \in \mathcal{A}rgs$ with $e \in concs(A)$. From the fact that $A \in \mathcal{A}rgs$ it follows that $A \in \mathcal{A}rgs'$, so $e \in concs(\mathcal{A}rgs')$. From the fact that $e \in S$, it also follows that $e \in atoms(T_1)$, so from the fact that $e \in concs(\mathcal{A}rgs')$ it follows that $e \in concs(\mathcal{A}rgs')_{|\text{atoms}(T_1)}$.

From the thus proved fact that $concs(\mathcal{A}rgs')_{|\text{atoms}(T_1)} = S$, together with the earlier observed fact that $concs(\mathcal{A}rgs')_{|\text{atoms}(T_1)} \in Cn_{\text{ssdl}}(T_3)_{|\text{atoms}(T_1)}$, it follows that $S \in Cn_{\text{ssdl}}(T_3)_{|\text{atoms}(T_1)}$.

'$\supseteq$': Let $S \in Cn_{\text{ssdl}}(T_3)_{|\text{atoms}(T_1)}$. Then there exists a semi-stable extension $\mathcal{A}rgs'$ of $AF_{T_3}^{\text{ctree}}$ with $concs(\mathcal{A}rgs')_{|\text{atoms}(T_1)} = S$. Like was explained before, it holds that:

$$Cn_{\text{semi-stable}}(AF_{T_1}^{\text{ctree}})_{|Ar_{T_1}^{\text{ctree}}} = Cn_{\text{semi-stable}}(AF_{T_3}^{\text{ctree}})_{|Ar_{T_1}^{\text{ctree}}}$$

Let $\mathcal{A}rgs = \mathcal{A}rgs' \cap Ar_{T_1}^{\text{ctree}}$. From the fact that $\mathcal{A}rgs'$ is a semi-stable extension of $AF_{T_3}^{\text{ctree}}$, it follows that $\mathcal{A}rgs \in Cn_{\text{semi-stable}}(AF_{T_3}^{\text{ctree}})_{|\text{atoms}(T_1)}$, so $\mathcal{A}rgs \in Cn_{\text{semi-stable}}(AF_{T_1}^{\text{ctree}})_{|\text{atoms}(T_1)}$, which together with the fact that $\mathcal{A}rgs \subseteq Ar_{T_1}^{\text{ctree}}$ implies that $\mathcal{A}rgs \in Cn_{\text{semi-stable}}(AF_{T_1}^{\text{ctree}})$, so $concs(\mathcal{A}rgs) \in Cn_{\text{ssdl}}(T_1)$, so $concs(\mathcal{A}rgs)_{|\text{atoms}(T_1)} \in Cn_{\text{ssdl}}(T_1)_{|\text{atoms}(T_1)}$. We now prove that $concs(\mathcal{A}rgs)_{|\text{atoms}(T_1)} = S$.

'$\subseteq$': Let $e \in concs(\mathcal{A}rgs)_{|\text{atoms}(T_1)}$. Then there is an $A \in \mathcal{A}rgs$ with $e \in concs(A)$, so $A \in Ar_{T_1}^{\text{ctree}}$. From $A \in \mathcal{A}rgs$ it follows that $A \in \mathcal{A}rgs'$, so $e \in concs(\mathcal{A}rgs')$. This, together with the fact that all atoms of e are from $atoms(T_1)$, implies that $e \in concs(\mathcal{A}rgs')_{|\text{atoms}(T_1)}$, so $e \in S$.

'$\supseteq$': Let $e \in S$. Then $e \in concs(\mathcal{A}rgs')_{|\text{atoms}(T_1)}$, so there exists an argument $A \in \mathcal{A}rgs'$ with $e \in concs(A)$. Moreover, the fact that e is composed of atoms only from $atoms(T_1)$ implies that $A \in Ar_{T_1}^{\text{ctree}}$. It then follows that $A \in \mathcal{A}rgs' \cap Ar_{T_1}^{\text{ctree}}$, so $A \in \mathcal{A}rgs$, so $e \in concs(\mathcal{A}rgs)$. From the fact that e is composed of atoms only from $atoms(T_1)$, it then follows that $e \in concs(\mathcal{A}rgs)_{|\text{atoms}(T_1)}$.

From the fact that $concs(\mathcal{A}rgs) = S$, together with the fact that $concs(\mathcal{A}rgs) \in Cn_{\text{ssdl}}(T_1)_{|\text{atoms}(T_1)}$, it follows that $S \in Cn_{\text{ssdl}}(T_1)_{|\text{atoms}(T_1)}$. ■

LEMMA 7
Default logic under semi-stable semantics is non-trivial.

PROOF. Let S be a non-empty set of atoms. We now have to prove that there exists two default theories $T_1=(\mathcal{W}_1,\mathcal{D}_1)$ and $T_2=(\mathcal{W}_2,\mathcal{D}_2)$ with $atoms(T_1)=atoms(T_2)=S$, such that $Cn_{\text{ssdl}}(T_1)\neq Cn_{\text{ssdl}}(T_2)$. This is obtained with $\mathcal{W}_1=S$, $\mathcal{W}_2=\{\neg e \mid e\in S\}$ and $\mathcal{D}_1=\mathcal{D}_2=\emptyset$. In that case, it holds that $Cn_{\text{ssdl}}(T_1)=\{Cn(\mathcal{W}_1)\}$ and $Cn_{\text{ssdl}}(T_2)=\{Cn(\mathcal{W}_2)\}$. Since S is not empty, there exists an $e\in S$. It then follows that $e\in Cn_{\text{ssdl}}(T_1)$ but $e\notin Cn_{\text{ssdl}}(T_2)$. So $Cn_{\text{ssdl}}(T_1)_{|S}\neq Cn_{\text{ssdl}}(T_2)_{|S}$. ■

THEOREM 16
Default logic under semi-stable semantics satisfies crash resistance.

PROOF. Lemma 7 states that default logic under semi-stable semantics is non-trivial. Theorem 15 states that default logic under semi-stable semantics satisfies non-interference. Theorem 1 states that each non-trivial logical formalism that satisfies non-interference also satisfies crash resistance. ■

5.3 Default logic versus logic programming

As has been observed in [34, 38], there exists a clear connection between logic programming and default logic. In particular, when each rule $c\leftarrow a_1,\ldots,a_n,\texttt{not}\,b_1,\ldots,\texttt{not}\,b_m$ of a logic program is translated into a default of the form $a_1\wedge\ldots\wedge a_n:\neg b_1,\ldots,\neg b_m/c$ then the default extensions of the corresponding default theory (assuming that $\mathcal{W}=\emptyset$) will consist of the logical closure of the stable models of the original logic program.

PROPOSITION 5 ([34])
Let P be a logic program and $T_P=(\mathcal{W}_P,\mathcal{D}_P)$ be its associated default theory such that $\mathcal{W}_P=\emptyset$ and $\mathcal{D}_P=\{a_1\wedge\ldots\wedge a_n:\neg b_1,\ldots,\neg b_m/c \mid c\leftarrow a_1,\ldots,a_n,\texttt{not}\,b_1,\ldots,\texttt{not}\,b_m\in P\}$.
Let $M_1,\ldots,M_k$ be the stable models of P. It holds that $Cn(M_1),\ldots,Cn(M_k)$ are the default extensions of T_P.

One can ask whether the above-described correspondence between logic programming and default logic continues to hold when changing the semantics from stable to semi-stable. The answer to this question is positive, as is stated by the following proposition.

PROPOSITION 6
Let P be a logic program and $T_P=(\mathcal{W}_P,\mathcal{D}_P)$ be its associated default theory such that $\mathcal{W}_P=\emptyset$ and $\mathcal{D}_P=\{a_1\wedge\ldots\wedge a_n:\neg b_1,\ldots,\neg b_m/c \mid c\leftarrow a_1,\ldots,a_n,\texttt{not}\,b_1,\ldots,\texttt{not}\,b_m\in P\}$.
Let $Cn_{\text{semi-stable}}(P)=\{M_1,\ldots,M_k\}$ (that is, $M_1,\ldots,M_k$ are the semi-stable models of P). It holds that $Cn_{\text{ssdl}}(T_P)=\{Cn(M_1),\ldots,Cn(M_k)\}$.

The validity of Proposition 6 can be verified as follows. First, consider the argumentation framework that is generated by logic program P following the procedure that is described in Definition 12. Now construct a copy of this argumentation framework, and replace in the arguments each rule by its coresponding default. This yields a graph in which each node contains a tree of defaults (which is essentially a pre-argument in the sense of Definition 19). Definition 19 requires each argument to be a set of pre-arguments, and this can be satisfied by for every node of the graph, replacing its pre-argument PA by $\{PA\}$. The resulting graph (call it $AF^{\text{tree}}_{T_P}=(Ar^{\text{tree}}_{T_P},att^{\text{tree}}_{T_P})$) is indeed an argumentation framework in the sense of Definition 19. This is because: (i) every node in the graph consists of a well-formed argument, (ii) every well-formed argument is included in the graph,

(iii) every arrow in the graph stands for an attack, (iv) every possible attack is represented by an arrow in the graph.

Since our starting point was a (normal) logic program (Definition 11) in which strong negation does not play a role, none of the resulting defaults will have negation in their consequents. This, together with the fact that we assume $\mathcal{W}_P$ to be the empty set, implies that every argument in AF_{T_P} is consistent. That is, for every $A \in Ar^{\text{tree}}_{T_P}$ it holds that $\bot \notin concs(A)$. Therefore, it holds that $(Ar^{\text{tree}}_{T_P}, att^{\text{tree}}_{T_P}) = (Ar^{\text{ctree}}_{T_P}, att^{\text{ctree}}_{T_P})$ (where the latter is defined by Definition 21).

Now, it should be noted that (Ar_P, att_P) $(=AF_P)$ is isomorphic to $(Ar^{\text{ctree}}_{T_P}, att^{\text{ctree}}_{T_P})$ $(=AF^{\text{ctree}}_{T_P})$. After all, the only thing the above-described translation process does is to change the contents of the arguments. Nothing is changed with respect to the actual structure of the graph (its nodes and arrows). Since semi-stable semantics, just like any mainstream argumentation semantics, satisfies *language independence* [5], the fact that AF_P and $AF^{\text{ctree}}_{T_P}$ are isomorphic means their extensions of arguments will correspond to each other.

Although the semi-stable extensions of AF_P are essentially the same as the semi-stable extensions of $AF^{\text{ctree}}_{T_P}$ on the abstract level, it does not necessarily follow that the conclusions yielded by these extensions of arguments are also the same. For instance, whereas an argument $A \in Ar_P$ only yields the head of its top-rule as a conclusion (Definition 12), an argument $A \in Ar^{\text{ctree}}_{T_P}$ yields a set not only containing the consequent of its top-default, but also the consequents of all other defaults in the argument, together with their logical consequences (Definition 19).

The first thing to notice is that although AF_P and $AF^{\text{ctree}}_{T_P}$ differ when it comes to the conclusions of the *individual* arguments, this difference becomes less if one looks at the conclusions yielded by the *extensions* of arguments. One particular feature that holds for both AF_P and $AF^{\text{ctree}}_{T_P}$ is that if a set of arguments $\mathcal{A}rgs$ defends an argument A, then it also defends all subarguments (subtrees) of A. It then follows that if a complete extension contains an argument A (meaning that it defends A) then it also contains all subarguments of A (as these are also defended). Since every semi-stable extension is also a complete extension, this property also holds for semi-stable extensions.

Let $\mathcal{A}rgs_P$ be a semi-stable extension of AF_P and $\mathcal{A}rgs^{\text{ctree}}_{T_P}$ be the associated semi-stable extensions of $AF^{\text{ctree}}_{T_P}$ (i.e. $\mathcal{A}rgs^{\text{ctree}}_{T_P}$ consists of those arguments from $\mathcal{A}rgs_P$ where the logic programming rules have been replaced by the corresponding defaults, using the procedure described earlier). It holds that every conclusion of $\mathcal{A}rgs_P$ ($c \in concs(\mathcal{A}rgs_P)$) is also a conclusion of $\mathcal{A}rgs^{\text{ctree}}_{T_P}$ ($c \in concs(\mathcal{A}rgs^{\text{ctree}}_{T_P})$). This is because $c \in concs(\mathcal{A}rgs_P)$ means that there exists an $A_P \in \mathcal{A}rgs_P$ with c as the head of its top-rule. Therefore, the associated $A^{\text{ctree}}_{T_P} \in \mathcal{A}rgs^{\text{ctree}}_{T_P}$ will have c as the consequent of its top-default. Therefore, $c \in concs(\mathcal{A}rgs^{\text{ctree}}_{T_P})$. Furthermore, it also holds that each *atomic* conclusion of $\mathcal{A}rgs^{\text{ctree}}_{T_P}$ is also a conclusion of $\mathcal{A}rgs_P$. This can be seen as follows. Let c be an atomic formula of $concs(\mathcal{A}rgs^{\text{ctree}}_{T_P})$. Then there exists an argument $A^{\text{ctree}}_{T_P}$ that has c as the consequent of one of its defaults. Let A_P be the associated argument from $\mathcal{A}rgs_P$. Then A_P has c as the head of one of its rules. Since, as we have observed earlier, $\mathcal{A}rgs_P$ is closed under subarguments, there will be an $A'_P \in \mathcal{A}rgs_P$ that has c as the head of its top-rule. Therefore, $c = Conc(A'_P)$ and hence, $c \in concs(\mathcal{A}rgs_P)$.

From the fact that each element of $concs(\mathcal{A}rgs_P)$ is also an element of $concs(\mathcal{A}rgs^{\text{ctree}}_{T_P})$ (and that these elements are atomic, because every logic program in the sense of Definition 11 has an atom as the consequent of each rule) and from the fact that each atomic element of $concs(\mathcal{A}rgs^{\text{ctree}}_{T_P})$ is also an element of $concs(\mathcal{A}rgs_P)$, it follows that $\mathcal{A}rgs_P$ and $\mathcal{A}rgs^{\text{ctree}}_{T_P}$ agree on the atomic conclusions. Since each default in T_P has an atomic conlusion as its consequent, it holds that every element of $\mathcal{A}rgs^{\text{ctree}}_{T_P}$ that is not an atomic conclusion has to be a logical consequence of the atomic conclusions. Hence, $\mathcal{A}rgs^{\text{ctree}}_{T_P} = Cn(\mathcal{A}rgs_P)$.

TABLE 1. Decision problems in AFs

Problem name	Instance	Question
Verification ($\text{VER}_{\mathcal{E}}$)	$\mathcal{H}=(Ar,att)$; $S\subseteq Ar$	Is $S\in\mathcal{E}(\mathcal{H})$?
Non-emptiness ($\text{EXISTS}_{\mathcal{E}}^{\neg\emptyset}$)	$\mathcal{H}=(Ar,att)$	Is there any $S\in\mathcal{E}(\mathcal{H})$ for which $S\neq\emptyset$?
Coincident ($\text{COIN}_{\mathcal{E},\mathcal{F}}$)	$\mathcal{H}=(Ar,att)$	Is $\mathcal{E}(\mathcal{H})=\mathcal{F}(\mathcal{H})$?
Credulous Acceptance ($\text{CA}_{\mathcal{E}}$)	$\mathcal{H}=(Ar,att)$; $x\in Ar$	Is there *any* $S\in\mathcal{E}(\mathcal{H})$ for which $x\in S$?
Sceptical Acceptance ($\text{SA}_{\mathcal{E}}$)	$\mathcal{H}=(Ar,att)$; $x\in Ar$	Is x a member of *every* $T\in\mathcal{E}(\mathcal{H})$?

6 Computational complexity

We assume the reader is familiar with the standard complexity classes P, NP, coNP together with classes in the so-called *Polynomial Hierarchy* (PH), in particular Σ_2^p and Π_2^p. We further assume some familiarity with the concept of polynomial-time many-one reducibility between decision problems. An accessible introduction to these may be found in Papadimitriou's text [41].

The class D^p is formed by decision problems L, whose positive instances are characterized as those belonging to $L_1\cap L_2$ where $L_1\in\text{NP}$ and $L_2\in\text{coNP}$. The problem SAT-UNSAT whose instances are pairs of 3-CNF formulae $\langle\varphi_1,\varphi_2\rangle$ accepted if φ_1 is satisfiable *and* φ_2 is unsatisfiable has been shown to be complete for this class [41, p. 413]. We may interpret D^p as those decision problems solvable by a (deterministic) polynomial time algorithm allowed to make at most two calls upon an NP *oracle*. More generally, the complexity class P^{NP} consists of decision problems that can be solved by a (deterministic) polynomial time algorithm provided with access to an NP *oracle* (calls upon which take a single step so that only polynomially many invocations are allowed). An important (presumed) subset of P^{NP} is defined by distinguishing whether oracle calls are *adaptive*—i.e. the exact formulation of the next oracle query may be dependent on the answers received to previous questions—or whether such queries are *non-adaptive*, i.e. the form of the questions to be put to the oracle is predetermined allowing all of these to be performed in parallel. The latter class has been denoted $\text{P}_{||}^{\text{NP}}$ and considered in Wagner [53, 54], Jenner and Toran [36].

Under the standard complexity-theoretic assumptions, it is conjectured that,

$$\text{P}\subset\left\{\begin{array}{c}\text{NP}\\ \text{coNP}\end{array}\right\}\subset\text{D}^p\subset\text{P}_{||}^{\text{NP}}\subset\text{P}^{\text{NP}}\subset\left\{\begin{array}{c}\Sigma_2^p\\ \Pi_2^p\end{array}\right\}$$

Given an argumentation framework $\mathcal{H}=(Ar,att)$, and a particular extension-based semantics $\mathcal{E}$, e.g. $\mathcal{E}$ could be any of SE (stable), PE (preferred) or SSE (semi-stable), Table 1 describes a number of general decision problems relative to $\mathcal{E}$. A number of natural problems concern the behaviour of frameworks regarding distinct extension semantics. In particular given extension semantics $\mathcal{E}$ and $\mathcal{F}$, the decision problem *Coincident* ($\text{COIN}_{\mathcal{E},\mathcal{F}}$) accepts an argumentation framework $\mathcal{H}=(Ar,att)$ if and only if $\mathcal{E}(\mathcal{H})=\mathcal{F}(\mathcal{H})$.

The results proved in this section are summarized in Table 2.

The results described in the first three lines of Table 2 are straightforward developments of constructions originally presented in [24] and [29]. The hardness results regarding credulous and

TABLE 2. Computational complexity w.r.t. semi-stable extensions

Problem	Lower bound	Upper bound	
VER_{SSE}	coNP-hard	coNP	Theorem 17
$\text{EXISTS}_{\text{SSE}}^{\neg\emptyset}$	NP-hard	NP	Corollary 1
$\text{COIN}_{\text{PE,SSE}}$	Π_2^p-hard	Π_2^p	Theorem 18
CA_{SSE}	$\text{P}_{\|\|}^{\text{NP}}$-hard	Σ_2^p	Theorem 19
SA_{SSE}	$\text{P}_{\|\|}^{\text{NP}}$-hard	Π_2^p	Theorem 20

sceptical acceptance under semi-stable semantics exploit a technical characterization of complete problems within $\text{P}_{||}^{\text{NP}}$ due to Chang and Kadin [19]. This introduces the concepts of a language having the properties OP_2 and OP_ω where OP is one of the Boolean operators {AND, OR}.

DEFINITION 24 ([19], pp. 175–76)
Let L be a language, i.e. a set of finite words over an alphabet. The languages, $\text{AND}_k(L)$ and $\text{OR}_k(L)$ $(k \geq 1)$ are

$$\begin{aligned} \text{AND}_k(L) &=_{\text{def}} \{\langle w_1, w_2, \ldots, w_k\rangle : \forall\, 1 \leq i \leq k\ w_i \in L\} \\ \text{OR}_k(L) &=_{\text{def}} \{\langle w_1, w_2, \ldots, w_k\rangle : \exists\, 1 \leq i \leq k\ w_i \in L\} \end{aligned}$$

The languages $\text{AND}_\omega(L)$ and $\text{OR}_\omega(L)$ are,

$$\text{AND}_\omega(L) =_{\text{def}} \bigcup_{k\geq 1} \text{AND}_k(L) \quad ; \quad \text{OR}_\omega(L) =_{\text{def}} \bigcup_{k\geq 1} \text{OR}_k(L)$$

A language, L, is said to have property OP_k (resp. OP_ω) if $\text{OP}_k(L) \leq_m^p L$ (respectively $\text{OP}_\omega(L) \leq_m^p L$).

The reason why these language operations are of interest is the following result.

FACT 1 ([19], Theorem 9, p. 182)
A language L is $\text{P}_{||}^{\text{NP}}$–complete (via $\leq_m^p$ reducibility) if and only if all of the following hold.

F1. $L \in \text{P}_{||}^{\text{NP}}$.
F2. L is NP–hard *and* L is coNP–hard.
F3. L has property AND_2.
F4. L has property OR_ω.

THEOREM 17
VER_{SSE} is coNP–complete.

PROOF. Given $\mathcal{H} = (Ar, att)$ and $S \subseteq Ar$, S defines a semi-stable extension of $\mathcal{H}$ if and only if, S is admissible and

$$\forall\, T \subseteq Ar \quad T \in \text{ADM}(\mathcal{H}) \ \Rightarrow\ \neg\,(S \cup S^+ \subset T \cup T^+)$$

a test that is easily accomplished by a coNP algorithm.

For coNP-hardness, it suffices to consider the special case $S = \emptyset$, i.e. the problem $\text{VER}_{\text{SSE}}(\mathcal{H}, \emptyset)$, which is the complement of $\text{EXISTS}_{\text{SSE}}^{\neg\emptyset}$. Given an instance of *unsatisfiability*—without loss of generality a 3-CNF formula $\varphi(Z_n) = \wedge_{j=1}^{m} C_j$—with each C_j a disjoint of literals from $\{z_1, \ldots, z_n,$

$\neg z_1, \ldots, \neg z_n\}$, the argumentation framework, $\mathcal{H}_\varphi = (Ar, att)$ has

$$\begin{array}{rcl} Ar & = & \{\varphi, C_1, \ldots, C_m\} \cup \{z_i,\ \neg z_i : 1 \leq i \leq n\} \\ att & = & \{\langle C_j, \varphi\rangle : 1 \leq j \leq m\} \cup \{\langle z_i, \neg z_i\rangle,\ \langle \neg z_i, z_i\rangle : 1 \leq i \leq n\} \cup \\ & & \{\langle z_i, C_j\rangle : z_i \text{ occurs in } C_j\} \cup \{\langle \neg z_i, C_j\rangle : \neg z_i \text{ occurs in } C_j\} \end{array}$$

As shown by [24], there is an admissible set containing the argument φ if and and only if $\varphi(Z_n)$ is satisfiable, i.e. $\neg \text{CA}_{\text{ADM}}(\mathcal{H}_\varphi, \varphi)$ if and only if $\varphi(Z_n)$ is unsatisfiable. Modify $\mathcal{H}_\varphi$ to the argumentation framework $\mathcal{K}_\varphi$ as follows: add a single new argument ψ to Ar together with $2n+1$ new attacks $\{\langle \psi, z_i\rangle : 1 \leq i \leq n\}$, $\{\langle \psi, \neg z_i\rangle : 1 \leq i \leq n\}$, and $\langle \varphi, \psi\rangle$. The argumentation framework $\mathcal{K}_\varphi$ has a non-empty *preferred* extension if and only if the CNF, φ is satisfiable. Hence, $\text{VER}_{\text{SSE}}(\mathcal{K}_\varphi, \emptyset)$ holds if and only if $\varphi(Z_n)$ is unsatisfiable. ■

COROLLARY 1
$\text{EXISTS}_{\text{SSE}}^{\neg\emptyset}$ is NP-complete.

PROOF. For membership in NP it suffices to test if $\text{EXISTS}_{\text{ADM}}^{\neg\emptyset}(\mathcal{H})$. The NP-hardness lower bound is immediate from the the proof of Theorem 17. ■

THEOREM 18
$\text{COIN}_{\text{PE,SSE}}$ is Π_2^p–complete.

PROOF. Given $\mathcal{H} = (Ar, att)$ every preferred extension of $\mathcal{H}$ is also a semi-stable extension if and only if,

$$\forall\, S \subseteq Ar \quad S \notin \text{PE}(\mathcal{H}) \ \vee \ S \in \text{SSE}(\mathcal{H})$$

This may be re-written as, $\forall\, S,\ T\ \exists\, U\, f(S, T, U)$ where $f(S, T, U)$ is the (polynomial time decidable) predicate

$$(S \notin \text{ADM}(\mathcal{H})) \bigvee (U \in \text{ADM}(\mathcal{H})) \wedge (S \subset U)) \bigvee$$
$$(\ (S \cup S^+ \subset T \cup T^+) \Rightarrow (T \notin \text{ADM}(\mathcal{H}))\ )$$

That is, 'for every subset (S), *either* S does not define a preferred extension of $\mathcal{H}$ (by reason of inadmissibility or containment in a larger admissible set, U) *or* (should S be a preferred extension), there is no (admissible) set (T) for which $S \cup S^+$ is strictly contained in $T \cup T^+$'.

The test described can be accomplished in Π_2^p.

To establish Π_2^p—hardness, we reduce to the complementary problem—i.e. that of deciding if a given $\mathcal{H}$ has a preferred extension that fails to be semi-stable, using the Σ_2^p-complete problem, QSAT_2^Σ instances of which comprise a CNF formula, $\varphi(Y_n, Z_n)$ over disjoint sets of propositional variables, that are accepted if there is some instantiation (α_Y) of Y_n for which every instantiation, (β_Z) of Z_n fails to satisfy $\varphi(Y_n, Z_n)$, i.e. $\exists\, \alpha_Y\ \forall\, \beta_Z\ \neg\varphi(\alpha_Y, \beta_Z)$. Given an instance, $\varphi(Y_n, Z_n)$, consider the argumentation framework $\mathcal{G}_\varphi(Ar', att')$ formed from the argumentation framework $\mathcal{H}_\varphi = (Ar, att)$ of Theorem 17, i.e. $Ar \subset Ar'$ and $att \subset att'$, so that

$$\begin{array}{rcl} Ar' & = & \{\varphi, C_1, \ldots, C_m\} \cup \{y_i, \neg y_i, z_i, \neg z_i : 1 \leq i \leq n\} \cup \{b_1, b_2, b_3\} \\ att' & = & \{\langle C_j, \varphi\rangle : 1 \leq j \leq m\} \cup \\ & & \{\langle y_i, \neg y_i\rangle, \langle \neg y_i, y_i\rangle, \langle z_i, \neg z_i\rangle, \langle \neg z_i, z_i\rangle : 1 \leq i \leq n\} \cup \\ & & \{\langle y_i, C_j\rangle : y_i \text{ occurs in } C_j\} \cup \{\langle \neg y_i, C_j\rangle : \neg y_i \text{ occurs in } C_j\} \cup \\ & & \{\langle z_i, C_j\rangle : z_i \text{ occurs in } C_j\} \cup \{\langle \neg z_i, C_j\rangle : \neg z_i \text{ occurs in } C_j\} \cup \\ & & \{\langle \varphi, b_1\rangle, \langle \varphi, b_2\rangle, \langle \varphi, b_3\rangle, \langle b_1, b_2\rangle, \langle b_2, b_3\rangle, \langle b_3, b_1\rangle\} \cup \\ & & \{\langle b_1, z_i\rangle, \langle b_1, \neg z_i\rangle : 1 \leq i \leq n\} \end{array}$$

From [29] this framework has a *non-stable* preferred extension if and only if $\varphi(Y_n, Z_n)$ is accepted as an instance of QSAT_2^{Σ}.[3] In particular, every satisfying instantiation of $\varphi(Y_n, Z_n)$ induces a corresponding *stable* extension of $\mathcal{G}_\varphi$. Now, noting that $\varphi(Y_n, Z_n)$ is accepted as instance of QSAT_2^{Σ} if and only if the CNF, $\psi(Y_n \cup \{u\}, Z_n)$ in which each clause of φ has a new variable u added to it, is also so accepted we claim that the argumentation framework $\mathcal{G}_\psi$ has a preferred (but not semi-stable) extension if and only if there is an instantiation, α of $Y_n \cup \{u\}$ under which $\psi(\alpha, Z_n)$ is unsatisfiable. First suppose $\text{SSE}(\mathcal{G}_\psi) \subset \text{PE}(\mathcal{G}_\psi)$. Since, $u = \top$ satisfies ψ, from the properties of $\mathcal{G}_\psi$ it follows that this has at least one stable extension, hence

$$\text{SE}(\mathcal{G}_\psi) = \text{SSE}(\mathcal{G}_\psi) \subset \text{PE}(\mathcal{G}_\psi)$$

and so $\mathcal{G}_\psi$ must contain a preferred extension which is not stable. From the analysis given in [29], we can construct an instantiation α_Y of Y_n which has $\psi(\alpha_Y, u = \bot, Z_n)$ unsatisfiable.

A similar analysis to that of [29] identifies a (non-stable) preferred extension for any instantation of α of $Y_n \cup \{u\}$ under which $\psi(\alpha, Z_n)$ is unsatisfiable, i.e. if ψ is accepted as an instance of QSAT_2^{Σ} then the set of preferred extensions of $\mathcal{G}_\psi$ does not coincide with its set of semi-stable extensions. ■

As a consequence of Fact 1, the lower bounds on CA_{SSE} and SA_{SSE} are derived using the following four part constructions.

S1. Prove that CA_{SSE} (respectively SA_{SSE}) is NP-hard.
S2. Prove that CA_{SSE} (respectively SA_{SSE}) is coNP-hard.
S3. Prove that CA_{SSE} (respectively SA_{SSE}) has property AND_2 (in fact, we will show both to have property AND_ω).
S4. Prove that CA_{SSE} (respectively SA_{SSE}) has property OR_ω.

THEOREM 19
a. CA_{SSE} is in Σ_2^p.
b. CA_{SSE} is $\text{P}_{||}^{\text{NP}}$–hard.

PROOF. For the upper bound in (a), x is a member of some semi-stable extension if and only if

$$\exists\, S \,\forall\, T \;\; (x \in S) \;\text{ and }\; (S \in \text{ADM}(\mathcal{H})) \text{ and}$$

$$(T \in \text{ADM}(\mathcal{H})) \Rightarrow \neg(\, (S \cup S^+ \subset T \cup T^+)\,)$$

Using the characterization of $\text{P}_{||}^{\text{NP}}$-complete languages described in Fact 1 the theorem follows given arguments that (S1)–(S4) all hold.

S1 CA_{SSE} is NP-hard.
Given an instance, $\varphi(Z_n)$ of SAT form an instance $\langle \mathcal{H}_\varphi, \varphi \rangle$ of CA_{SSE} in which $\mathcal{H}_\varphi$ is the AF described in the proof of Theorem 17. The argument φ is in a stable (hence semi-stable) extension of $\mathcal{H}_\varphi$ if and only if $\varphi(Z_n)$ is satisfiable. We deduce that CA_{SSE} is NP-hard as a result.

S2 CA_{SSE} is coNP-hard.
Given an instance $\varphi(Z_n)$ of UNSAT, first form the AF, $\mathcal{H}_\varphi$ as described in S1. Modify $\mathcal{H}_\varphi$ to a system $\mathcal{K}_\psi$ which has a new argument ψ added together with attacks $\langle \varphi, z_i \rangle$ and $\langle \varphi, \neg z_i \rangle$ for each $1 \le i \le n$ and $\{\langle \varphi, \psi \rangle\ \langle \psi, \varphi \rangle\}$. The instance is completed by choosing ψ as the argument of interest. The instance $\langle \mathcal{K}_\psi, \psi \rangle$ is accepted if there is a semi-stable extension containing ψ.

[3] Each witnessing non-stable but preferred extension is formed by a subset of $\{y_i, \neg y_i \; 1 \le i \le n\}$ for which the instantiation, α_Y of the corresponding literals in Y_n to $\top$ results in $\varphi(\alpha_Y, Z_n)$ being unsatisfiable.

If, however, there is a preferred extension containing φ, then this extension is also a stable extension which would preclude membership of ψ in a semi-stable set. Such a preferred extension exists if and only if $\varphi(Z_n)$ is satisfiable, so that $\langle \mathcal{K}_\psi, \psi \rangle$ is accepted as an instance of CA_{SSE} if and only if $\varphi(Z_n)$ is *unsatisfiable*.

S3. CA_{SSE} has property AND_ω.
Let $\langle\langle \mathcal{H}_1, x_1\rangle, \langle \mathcal{H}_2, x_2\rangle, \ldots, \langle \mathcal{H}_k, x_k\rangle\rangle$ define an instance of $\text{AND}_k(\text{CA}_{\text{SSE}})$. Form an instance $\langle \mathcal{H}, z\rangle$ of CA_{SSE} in which the k frameworks, $\mathcal{H}_i$ are extended by adding a set of k arguments $\{y_1, \ldots, y_k\}$, an argument z, and attacks $\{\langle y_i, z\rangle, \langle x_i, y_i\rangle\}$ for each $1 \leq i \leq k$. We claim that $\langle \mathcal{H}, z\rangle$ is accepted as an instance of CA_{SSE} if and only if each $\langle \mathcal{H}_i, x_i\rangle$ is accepted as such an instance. Suppose the latter is true and that S_i is a semi-stable extension in $\mathcal{H}_i$ that contains x_i. Then $S = \cup_{i=1}^{k} S_i \cup \{z\}$ is certainly admissible since each attack $\langle y_i, z\rangle$ is countered by the attack $\langle x_i, z\rangle$. Furthermore, S is a semi-stable extension as

$$S \cup S^+ = \bigcup_{i=1}^{k} S_i \cup S_i^+ \cup \{y_1, y_2, \ldots, y_k, z\}$$

so that all of the new arguments $\{y_1, \ldots, y_k, z\}$ occur within $S \cup S^+$. In total from semi-stable extensions S_i containing x_i we construct a semi-stable extension S containing z.
Conversely suppose S is a semi-stable extension of $\mathcal{H}$ and that $z \in S$. It is certainly the case that $\{x_1, \ldots, x_k\} \subset S$ since this is required in order to defend the attacks $\langle y_i, z\rangle$. Consider the set $S_i = S \cap Ar_i$ where Ar_i is the set of arguments in $\mathcal{H}_i$. Noting that $x_i \in S_i$ we claim that S_i is a semi-stable extension in $\mathcal{H}_i$. Suppose this were not the case so that some admissible subset, T of Ar_i satisfies $S_i \cup S_i^+ \subset T_i \cup T_i^+$. Without loss of generality, we may assume $x_i \notin T$ so that $x_i \in T^+$. Now consider the set $R = S \setminus (S_i \cup \{z\}) \cup T_i \cup \{y_i\}$. Observe that R is admissible: y_i being defended by the argument attacking x_i in T_i. In addition, however,

$$\begin{aligned} S \cup S^+ &= \cup_{j=1}^{k} S_j \cup S_j^+ \cup \{y_1, \ldots, y_k, z\} \\ &\subset \cup_{j \neq i}^{k} S_j \cup S_j^+ \; T_i \cup T_i^+ \cup \{y_1, \ldots, y_k, z\} \\ &= R \cup R^+ \end{aligned}$$

with R admissible and $z \notin R$: this contradicts the premise that S is semi-stable.

S4. CA_{SSE} has property OR_ω.
Let $\langle\langle \mathcal{H}_1, x_1\rangle, \langle \mathcal{H}_2, x_2\rangle, \ldots, \langle \mathcal{H}_k, x_k\rangle\rangle$ define an instance of $\text{OR}_k(\text{CA}_{\text{SSE}})$. Form an instance $\langle \mathcal{H}, z\rangle$ of CA_{SSE} in which the k frameworks, $\mathcal{H}_i$ are extended by adding arguments $\{y, z\}$ and attacks $\{\langle x_i, y\rangle : 1 \leq i \leq k\}$ and $\langle y, z\rangle$. First suppose, without loss of generality the $x_1 \in S_1$ a semi-stable extension of $\mathcal{H}_1$. Let $\langle S_2, \ldots, S_k\rangle$ be semi-stable extensions of $\mathcal{H}_i$ for $2 \leq i \leq k$. Then it easily follows that $S = \{z\} \cup \cup_{i=1}^{k} S_i$ is not only admissible but a semi-stable of $\mathcal{H}$ containing z. On the other hand suppose S with $x \in S$ is a semi-stable extension of $\mathcal{H}$. There must be at least one $\mathcal{H}_i$ for which $x_i \in S$ in order to defend the attack by y on z. Now considering the set $S \cap S_i$ gives a semi-stable extension in $\mathcal{H}_i$ containing x_i by a similar argument to that used in S3. ■

THEOREM 20

a. SA_{SSE} is in Π_2^p.
b. SA_{SSE} is $\text{P}_{||}^{\text{NP}}$-hard.

PROOF. We omit the easy upper bound proof, concentrating on (b). As before we obtain the result in four stages.

T1 SA_{SSE} is NP-hard.
Given an instance $\varphi(Z_n)$ of satisfiability, form the framework $\mathcal{K}_\varphi$ described in Theorem 17. The instance of SA_{SSE} is given by $\langle \mathcal{K}_\varphi, \varphi \rangle$. If $\varphi(Z_n)$ is satisfiable then the subset $\langle a_1, a_2, \ldots, a_n \rangle$ of $\{z_i, \neg z_i : 1 \leq i \leq n\}$ indicated by any satisfying assignment together with φ is a stable extension and hence also semi-stable. Hence, $\varphi(Z_n)$ satisfiable implies $\text{SA}_{\text{SSE}}(\mathcal{K}_\varphi, \varphi)$ holds. On the other hand, if $\varphi(Z_n)$ is unsatisfiable then (as argued in the proof of Theorem 17) $\mathcal{K}_\varphi$ has only the empty set as a semi-stable extension. We deduce that SA_{SSE} is NP-hard.

T2 SA_{SSE} is coNP-hard.
Given an instance, $\varphi(Z_n)$ of unsatisfiability, construct the framework $\mathcal{K}_\varphi$ described above but without the attacks $\langle \psi, z_i \rangle$ and $\langle \psi, \neg z_i \rangle$. The instance of SA_{SSE} is $\langle \mathcal{K}_\varphi, \psi \rangle$. If $\varphi(Z_n)$ is satisfiable then ψ cannot belong to the semi-stable extension induced by a satisfying instantiation (since this contains the argument φ). On the other hand, if $\varphi(Z_n)$ is unsatisfiable then every stable extension of $\mathcal{K}_\varphi$ has the form: exactly one of each of the pairs $\{z_i, \neg z_i\}$, the subset of clause arguments that are unattacked, and the argument ψ. Hence, ψ is a member of every semi-stable extension if and only if $\varphi(Z_n)$ is unsatisfiable.

T3. SA_{SSE} has property AND_ω. Similar to (S3).

T4. SA_{SSE} has property OR_ω. Similar to (S4). ■

We observe that the lower bound on CA_{SSE} shows that this decision problem is at least as hard as the analogous problem in the ideal semantics of [27] as shown in [28], while the upper bound for SA_{SSE} matches that of sceptical reasoning w.r.t. to preferred semantics [29]. Finally, our exact bounds for the verification problem, showing this to be coNP-complete, are identical to those already demonstrated for preferred semantics [24] and recognition of ideal *sets* [28].[4]

In Coste-Marquis *et al.* [21], it was shown that upper bounds on decision problems involving *single* arguments continue to apply when analogous formulations for *sets* of arguments are used. It is, of course, trivially the case that *lower* bounds hold by the simple expedient of treating a single argument, x say, as a single element set, $\{x\}$. It is not hard to see that the upper bounds on CA_{SSE} and SA_{SSE} also have this property. Formally, let $\text{CA}^{\{\}}_{\text{SSE}}$ and $\text{SA}^{\{\}}_{\text{SSE}}$ denote the credulous (resp. sceptical) acceptance problems when instances are an AF (Ar, att) and $S \subseteq Ar$ accepted if S is a subset of at least one (resp. every) semi-stable extension. Then,

THEOREM 21
a. $\text{CA}^{\{\}}_{\text{SSE}} \in \Sigma^p_2$.
b. $\text{SA}^{\{\}}_{\text{SSE}} \in \Pi^p_2$.

PROOF. For (a), S is a subset of some semi-stable extension if and only if

$$\exists\, T\ \forall\, U\ \ (S \subseteq T)\ \text{ and }\ (T \in \text{ADM}(\mathcal{H}))\ \text{and}$$

$$(U \in \text{ADM}(\mathcal{H})) \Rightarrow\ \neg(\ (T \cup T^+ \subset U \cup U^+)\ )$$

For (b), S is a subset of every semi-stable extension if and only if

$$\forall\, T\ (T \notin \text{SSE}(\mathcal{H})\quad \text{or } S \subseteq T$$

which is a Π^p_2 computation: the result of Theorem 17 showing that $T \notin \text{SSE}(\mathcal{H})$ can be decided in NP. ■

[4] Recently, optimal bounds on the complexity of sceptical and credulous acceptance have been obtained by Dvorak and Woltran [31]: these show credulous acceptance to be Σ^p_2-hard, and sceptical acceptance Π^p_2-hard, matching the upper bounds given in this article.

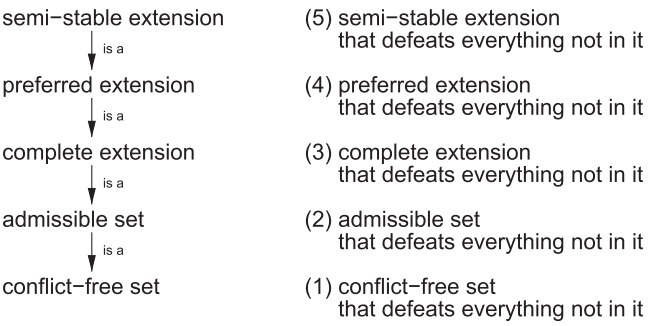

FIGURE 7. Hierarchy of argumentation related sets (left) and hierarchy of definitions of stable semantics (right).

7 Related Work: Semi-stable Semantics versus Stage Semantics

In this section, we treat an alternative approach that can be shown to satisfy the postulates of crash resistance, non-interference and backward compatibility, which was originally proposed by Verheij under the name of *stage extensions* [51]. Before treating Verheij's work, however, it can be worthwhile to sketch some context. Like was discussed before, every semi-stable extension is a preferred extension, every preferred extension is a complete extension, every complete extension is an admissible set, and every admissible set is a conflict-free set. This yields the picture at the left-hand side of Figure 7.

Similarly, there also exist various equivalent ways of defining stable semantics. These are shown at the right hand side of Figure 7. Their equivalence has for most part already been stated and proved at Proposition 3. The only thing that is still to be done is a proof that level (5) is equivalent to level (4), which is expressed in Proposition 7 below.

PROPOSITION 7
Let (Ar, att) be an argumentation framework and let $\mathcal{A}rgs \subseteq Ar$. The following statements are equivalent.

(5) $\mathcal{A}rgs$ is a semi-stable extension that attacks every argument in $Ar \backslash \mathcal{A}rgs$.
(4) $\mathcal{A}rgs$ is a preferred extension that attacks every argument in $Ar \backslash \mathcal{A}rgs$.

PROOF.
from 5 to 4: Trivial, since each semi-stable extension is also a preferred extension (Theorem 3).
from 4 to 5: Let $\mathcal{A}rgs$ be a preferred extension that attacks every argument in $Ar \backslash \mathcal{A}rgs$. From Proposition 3, it follows that $\mathcal{A}rgs$ is a stable extension. The fact that there exists a stable extension means, by Theorem 5, that the set of stable extensions is equal to the set of semi-stable extensions. This means that $\mathcal{A}rgs$ is also a semi-stable extension. ■

To some scholars, stable extensions appear as a sometimes unreachable ideal. If a condition is too strong to be fulfilled, then perhaps it makes sense to weaken it. In the case of stable semantics, the most obvious way of weakening would be to drop the condition that a stable extension attacks every argument not in it. The result, however, depends on the particular definition of stable semantics one starts with. For instance, if one weakens the level 4 definition (Figure 7) like this, then one ends up with the notion of preferred semantics. Likewise, if one applies this to the level 3 definition, then one ends up with the notion of complete semantics. The higher one goes, the stronger the resulting

semantics becomes. With semi-stable semantics, one tries to obtain an alternative for stable semantics that is still as strong as possible. A different approach would be not to go up, but to go down as much as possible. This would result in a definition that merely requires conflict-freeness, which forms the basis of the work of Verheij [51].

Although in [51] the concept of a stage extension was expressed in terms of pairs of sets of justified and defeated arguments, in this article, we will examine Verheij's basic idea in terms of the more established extensions approach (as has also been done in [14, 17]).

DEFINITION 25
Let (Ar, att) be an argumentation framework and $\mathcal{A}rgs \subseteq Ar$. $\mathcal{A}rgs$ is a *stage extension* iff $\mathcal{A}rgs$ is a conflict-free set where $\mathcal{A}rgs \cup \mathcal{A}rgs^+$ is maximal.

It is interesting to compare stage extensions with semi-stable extensions. Both semi-stable extensions and stage extensions have a maximal range. The difference is that a semi-stable extension has to be a complete extension, while a stage extension only has to satisfy the much weaker condition of being conflict-free.[5]

As an example of how stage extensions work, consider the argumentation framework consisting of an odd loop of three arguments A, B and C where A attacks B, B attacks C, and C attacks A. Here, there exist four conflict-free sets: $\emptyset$, $\{A\}$, $\{B\}$ and $\{C\}$, of which $\{A\}$, $\{B\}$ and $\{C\}$ are stage extensions.

In essence, what stage semantics does is to take the stable extensions of the maximal subframeworks that have at least one stable extension. That is, stage semantics tries to ignore minimal parts of the original argumentation framework that need to be ignored in order to obtain stable extensions. This is made formal in the following theorem from [14] where, given an argumentation framework $AF = (Ar, att)$ and a set $\mathcal{A}rgs \subseteq Ar$, $AF_{|\mathcal{A}rgs}$ stands for $(\mathcal{A}rgs, att \cap (\mathcal{A}rgs \times \mathcal{A}rgs))$.

THEOREM 22 ([14])
Let $AF = (Ar, att)$ be an argumentation framework and $\mathcal{A}rgs \subseteq Ar$. The following statements are equivalent:

(1) $\mathcal{A}rgs$ is a stage extension of AF.
(2) $\mathcal{A}rgs \cup \mathcal{A}rgs^+$ is a maximal subset of Ar such that $AF_{|\mathcal{A}rgs \cup \mathcal{A}rgs^+}$ has a stable extension, and $\mathcal{A}rgs$ is a stable extension of $AF_{\mathcal{A}rgs \cup \mathcal{A}rgs^+}$.

Using Theorem 22, it is not too difficult to see that stage semantics, just like semi-stable semantics, satisfies the postulates of crash resistance, non-interference and backward compatibility. It satisfies non-interference (and therefore also crash resistance), because unconnected components of the argumentation framework do not influence each other when selecting the maximal subframeworks that have stable extensions. It satisfies backward compatibility with stable semantics, because if the argumentation framework has a stable extension, then the maximal subframework that has a stable extension will be the entire argumentation framework itself.

[5] Verheij also studies what he calls *admissible stage extensions*, which can to some extent be described as admissible sets $\mathcal{A}rgs$ with maximal range ($\mathcal{A}rgs \cup \mathcal{A}rgs^+$). As was stated by Proposition 4, these correspond to semi-stable extensions. The similarity between admissible stage extensions and semi-stable extensions might not be immediately obvious, since Verheij chooses to express his ideas not in the standard way of argument extensions, but in the form of what he calls *stages*, which are pairs (J, D) where J is a set of *justified* arguments and D a set of *defeated* arguments [51].

FIGURE 8. Stage extensions do not always contain non-attacked arguments.

In order to understand the difference between stage semantics and semi-stable semantics, it is useful to make an analogy with classical logic. In the presence of a potentially inconsistent knowledge base, one could do two things:

(1) Take the maximal consistent subsets of the knowledge base, and examine what is entailed by all of these. That is, take the (classical) models of the maximal subsets of the knowledge base that have classical models.
(2) Define a new (paraconsistent) semantics such that the entire knowledge base will have models, without the need to ignore any of its contents.

Solution 1 (applying the original semantics to maximal subsets of the original problem description) is comparable to stage semantics, whereas solution 2 (redefining the semantics so that it can meaningfully be applied to a bigger class of knowledge bases) is comparable with semi-stable semantics.

With semi-stable and stage semantics, we have two approaches that both satisfy the postulates described in Section 2, and we cannot rule out that there exist more approaches that satisfy them. However, the postulates of crash resistance, non-interference and backward compatibility should be seen as *minimal* properties that one would like to obtain, and one can have good reasons for preferring some approaches that satisfy them above other approaches that also satisfy them. We now look at two such reasons, which are related to the status of attacked arguments, and the consistency of the argument-based conclusions.

7.1 On the status of unattacked arguments

To examine one of the fundamental differences between stage semantics and semi-stable semantics, it is illustrative to look at the argumentation framework of Figure 8

Here, there exist two stage extensions: $\{A\}$ and $\{B\}$. The first one is a stable extension of the subframework consisting of only A and B, the second one is a stable extension of the subframework consisting of only B and C. So although A is an argument without any attackers, it is not a member of every stage extension.[6] This can be seen as unusual, or even as undesirable, because in the argumentation framework of Figure 8 there seems to be no good reason for excluding A from any extension.

With semi-stable semantics, on the other hand, the argumentation framework of Figure 8 yields just a single extension: $\{A\}$. Furthermore, since each semi-stable extension is by definition also a complete extension, it follows that for *any* argumentation framework, each argument without any attackers is a member of each semi-stable extension. This puts semi-stable semantics in line with other mainstream semantics, like grounded, preferred, stable, complete, ideal and CF2, all of which satisfy the property that unattacked arguments are in every extension. The only mainstream semantics that violates this property is stage semantics.

[6]We would like to thank Pietro Baroni and Massimiliano Giacomin for this observation.

7.2 On the consistency of argument-based conclusions

Apart from the issue of how to treat unattacked arguments, there is also a second important difference between stage semantics and semi-stable semantics, which has to do with how one applies argumentation for inferring defeasible conclusions. Argumentation formalisms like [4, 35, 42, 46] assume the presence of a set of strict and defeasible rules (or reasons) which are used to construct arguments. As each argument has one (in some formalisms possibly more than one) conclusion, one can determine the conclusions associated to an extension, as well as the overall justified conclusions — usually based on the sceptical approach: an conclusion is overall justified iff it is entailed by every extension. Let us consider an argumentation formalism like ASPIC [4, 15, 45] or the argument-theoretic interpretation of OSCAR [43, 44]. Suppose the following nondefeasible information is present: $\{a, b, \neg(c \wedge d)\}$. Also suppose there are two defeasible rules: $a \Rightarrow c$ and $b \Rightarrow d$. One can now construct at least the following five arguments.

A: $(a) \Rightarrow c$
B: $(b) \Rightarrow d$
C: $((a) \Rightarrow c), \neg(c \wedge d) \rightarrow \neg d$
D: $((b) \Rightarrow d), \neg(c \wedge d) \rightarrow \neg c$
E: $\neg(c \wedge d)$

In the argument-theoretic version of OSCAR, as well as in [4], an argument can attack another argument by having a conclusion that is the opposite of the consequent of a defeasible rule in the other argument. Thus, in our example D attacks A and C, and C attacks B and D. The argument E does not have any attackers. The set $\{A, B, E\}$ is conflict-free but is not admissible, since it does not defend itself against C and D. Worse yet, the set $\{A, B, E\}$, even though it is conflict-free, has inconsistent conclusions (c, d and $\neg(c \wedge d)$). This illustrates that conflict-freeness does not imply consistency.

It is interesting to see what happens if one replaces the requirement of conflict-freeness by the requirement of admissibility. Is there still a problem with consistency for admissible sets of arguments? It turns out the answer is no. This can be seen as follows. Suppose an admissible set $\mathcal{A}rgs$ yields inconsistent conclusions. Then $\mathcal{A}rgs$ contains some minimal subset $\{A_1, \ldots, A_n\}$ such that $\{\mathtt{Conc}(A_1), \ldots, \mathtt{Conc}(A_n)\}$ is inconsistent.[7] Under the assumption that the nondefeasible information is consistent, this means that at least one of these arguments (say: A_i) contains at least one defeasible rule. As $\{A_1, \ldots, A_n\}$ is a *minimal* set of arguments yielding inconsistent conclusions, this means that $\{A_1, \ldots, A_{i-1}, A_{i+1}, \ldots, A_n\}$ yields conclusions that are not only consistent, but from which also the negation of the conclusion of A_i follows under classical logic. That is: $\{\mathtt{Conc}(A_1), \ldots, \mathtt{Conc}(A_{i-1}), \mathtt{Conc}(A_{i+1}), \ldots, \mathtt{Conc}(A_n)\} \models \neg\mathtt{Conc}(A_i)$. As in OSCAR, as well as in other approaches, strict rules coincide with classical entailment, there exists a strict rule of the form $\mathtt{Conc}(A_1), \ldots, \mathtt{Conc}(A_{i-1}), \mathtt{Conc}(A_{i+1}), \ldots, \mathtt{Conc}(A_n)\} \rightarrow \neg\mathtt{Conc}(A_i)$. This rule can then be used in an argument (say A') of the form $A_1, \ldots, A_{i-1}, A_{i+1}, \ldots, A_n \rightarrow \neg\mathtt{Conc}(A_i)$. This argument can then serve as a basis for constructing an argument (say A'') that attacks A_i, possibly by using parts of A_i itself. As $\mathcal{A}rgs$ is admissible, it should defend itself against A''. Therefore, it should contain an argument (say B) against the consequent of some defeasible rule in A''. But as every defeasible rule in A'' also occurs in some argument in $\mathcal{A}rgs$, this means that B attacks some argument in $\mathcal{A}rgs$. Therefore, $\mathcal{A}rgs$ is not conflict-free, which means it also cannot be an admissible set. Contradiction.

The key point here is that although weakening the requirement of admissibility to the requirement of merely conflict-freeness might seem reasonable when one takes into account a purely abstract view of argumentation in which arguments do not have an internal structure or even conclusions

[7] We write $\mathtt{Conc}(A_i)$ for the conclusion of argument A_i.

(like is done in [51]), it can lead to serious problems when one actually tries to apply this principle in a full blown argumentation formalism in which the emphasis is on defeasible entailment. For this, conflict-freeness is not enough; admissibility is really needed.

As an aside, one may argue that Verheij's approach of stage extensions is based not so much on conflict-free sets as such, but on conflict-free sets with a maximal *range*. Recall that a stage extension is a conflict-free set $\mathcal{A}rgs$ of which $\mathcal{A}rgs \cup \mathcal{A}rgs^+$ is maximal. In the above example, the set $\{A, B, E\}$ is not a stage extension since its range is $\{A, B, E\}$ and there exists a stage (for instance $\{A, C, E\}$) with a bigger range (in this case $\{A, B, C, D, E\}$). Although the approach of stage *extensions* thus properly deals with the above example, there exist other examples where this approach fails. Consider adding two new arguments F and G, where F is self-attacking and is attacked by A, and G is self-attacking and is attacked by B. Such arguments could for instance be created by the approach of using undercutters, as is done in [9]. In that case, the set $\{A, B, E\}$ is a stage extension. This is because its range ($\{A, B, E, F, G\}$) cannot be made larger. Thus, stage extensions do not necessarily produce consistent conclusions either. For Verheij's dialectical negation approach, where the concept of classical consistency does not exist in the first place, stage extensions work fine. For other approaches, it can cause some real problems.[8]

8 Discussion

In this article, we have stated three postulates (non-interference, crash resistance and backward compatibility) that aim to capture necessary properties for the notion of paraconsistency. That is, our aim is to describe what it means for a formalism to be a paraconsistent version of another formalism.[9] This makes it possible to meaningfully apply paraconsistency to a whole range of formalisms that are fundamentally different to classical logic, which has traditionally been the main focus of paraconsistency. To illustrate the applicability of these postulates outside of the domain of classical logic, we show how they can be satisfied with respect to three non-classical formalisms: abstract argumentation, logic programming and default logic.

It should be mentioned that obtaining the properties of non-interference and crash resistance is not just a matter of applying a particular semantics (such as semi-stable). Equally important is the issue of how arguments are constructed. This is in line with [15] in which a number of postulates is provided whose satisfaction depends on the argumentation semantics as well as on how the arguments are constructed. As an example, when applying semi-stable semantics to default logic (Section 5), one explicitly needs to rule out inconsistent arguments in order for non-interference to hold. This phenomenon is not necessarily related to semi-stable semantics. Pollock's OSCAR [43], for instance, implements preferred semantics [44] but, as is explained in [9], violates non-interference because it does not block the construction of inconsistent arguments. Our current work thus confirms the findings of [15] that argumentation semantics and argument construction cannot be studied purely in isolation. One needs to have a suitable combination of semantics and argument construction in order to obtain the kind of results that can be regarded as desirable.

[8]This also raises some tricky questions for other semantics that are not admissibility based, such as CF2 [6].

[9]When restricted to logics endowed with formal (model-theoretical) semantics, this idea has already been defended since 2005 by Alexandre Costa-Leite under the label 'paraconsistentization' [20]. More recently, Arieli *et al*. specified a systematic way of constructing what they call *ideal paraconsistent logics* [1, 2]. Their approach, however, is based on the notion of Tarskian consequence and assumes particular elements of classical logic, whereas our approach takes non-monotonic formalisms (like abstract argumentation, logic programming and default logic) as its starting point.

Acknowledgements

We would like to thank Bart Verheij, Philippe Besnard and Ofer Arieli for their useful comments and corrections.

References

[1] O. Arieli, A. Avron, and A. Zamansky. Ideal paraconsistent logics. *Studia Logica*, **99**, 31–60, 2011.

[2] O. Arieli, A. Avron, and A. Zamansky. What is an ideal logic for reasoning with inconsistency? In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, T. Walsh, ed., pp. 706–711, 2011.

[3] O. Arieli and A. Avron. The value of the four values. *Artificial Intelligence*, **102**, 97–141, 1998.

[4] ASPIC-consortium. Deliverable D2.5: Draft formal semantics for ASPIC system. 2005.

[5] P. Baroni and M. Giacomin. On principle-based evaluation of extension-based argumentation semantics. *Artificial Intelligence*, **171**, 675–700, 2007.

[6] P. Baroni, M. Giacomin, and G. Guida. SCC-recursiveness: a general schema for argumentation semantics. *Artificial Intelligence*, **168**, 165–210, 2005.

[7] A. Bondarenko, P. M. Dung, R. A. Kowalski, and F. Toni. An abstract, argumentation-theoretic approach to default reasoning. *Artificial Intelligence*, **93**, 63–101, 1997.

[8] G. Brewka and G. Gottlob. Well-founded semantics for default logic. *Fundamenta Informaticae*, **31**, 221–236, 1997.

[9] M. W. A. Caminada. Contamination in formal argumentation systems. In *Proceedings of the 17th Belgium-Netherlands Conference on Artificial Intelligence (BNAIC)*, K. Verbeeck, K. Tuyls, A. Nowé, B. Manderick, and B. Kuijpers, eds, pp. 59–65, 2005.

[10] M. W. A. Caminada. On the issue of reinstatement in argumentation. In *Logics in Artificial Intelligence; 10th European Conference, JELIA 2006*, Vol. 4160 of *LNAI*, M. Fischer, W. van der Hoek, B. Konev, and A. Lisitsa, eds, pp. 111–123. Springer, 2006.

[11] M. W. A. Caminada. Semi-stable semantics. In *Computational Models of Argument; Proceedings of COMMA 2006*, P. E. Dunne and T. J. M. Bench-Capon, eds, pp. 121–130. IOS Press, 2006.

[12] M. W. A. Caminada. An algorithm for computing semi-stable semantics. *Technical Report UU-CS-2007-010*, Utrecht University, 2007.

[13] M. W. A. Caminada. An algorithm for computing semi-stable semantics. In *Proceedings of the 9th European Conference on Symbolic and Quantitalive Approaches to Reasoning with Uncertainty (ECSQARU 2007)*, Vol. 4724 of *Springer Lecture Notes in AI*, pp. 222–234. Springer, 2007.

[14] M. W. A. Caminada. An algorithm for stage semantics. In *Proceedings of the Third International Conference on Computational Models of Argument (COMMA 2010)*, P. Baroni, F. Cerutti, M. Giacomin, and G. R. Simari, eds, pp. 147–158. IOS Press, 2010.

[15] M. W. A. Caminada and L. Amgoud. On the evaluation of argumentation formalisms. *Artificial Intelligence*, **171**, 286–310, 2007.

[16] M. W. A. Caminada and D. M. Gabbay. A logical account of formal argumentation. *Studia Logica*, **93**, 109–145, 2009. Special issue: new ideas in argumentation theory.

[17] M. W. A. Caminada and B. Verheij. On the existence of semi-stable extensions. In *Proceedings of the 22nd Benelux Conference on Artificial Intelligence*, G. Danoy, M. Seredynski, R. Booth, B. Gateau, I. Jars, and D. Khadraoui, eds, 2010.

[18] W. Carnielli, M. E. Coniglio, and J. Marcos. Logics of formal inconsistency. In *Handbook of Philosophical Logic*, 2nd edn. D. M. Gabbay and F. Guenthner, eds, Vol. 14, pp. 15–114. Springer, 2002.
[19] R. Chang and J. Kadin. On computing Boolean connectives of characteristic function. *Math. Syst. Theory*, **28**, 173–198, 1995.
[20] A. F. B. Costa-Leite. *Interactions of metaphysical and epistemic concepts*. PhD Thesis, Université de Neuchâtel, 2007.
[21] S. Coste-Marquis, C. Devred, and P. Marquis. Symmetric argumentation frameworks. In *Proceedings of the 8th European Conference on Symbolic and Quantitative Approaches to Reasoning With Uncertainty (ECSQARU)*, Vol. 3571 of *LNAI*, L. Godo, ed., pp. 317–328. Springer, 2005.
[22] N. C. A. da Costa. On the theory of inconsistent formal systems. *Notre Dame Journal of Formal Logic*, **15**, 497–510, 1974.
[23] C. V. Damásio and L. M. Pereira. A survey of paraconsistent semantics for logic programs. In *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, Ph. Besnard, A. Hunter, Dov M. Gabbay and Ph. Smets, eds, pp. 241–320. Kluwer Academic Publishers, 1998.
[24] Y. Dimopoulos and A. Torres. Graph theoretical structures in logic programs and default theories. *Theoretical Computer Science*, **170**, 209–244, 1996.
[25] S. Doutre and J. Mengin. An algorithm that computes the preferred extensions of argumentation frameworks. In *ECAI'2000, Third International Workshop on Computational Dialectics (CD'2000)*, G. Brewka, N. Jennings, and G. Vreeswijk, eds, pp. 55–62, 2000.
[26] P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and *n*-person games. *Artificial Intelligence*, **77**, 321–357, 1995.
[27] P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and *n*-person games. *Artificial Intelligence*, **77**, 321–357, 1995.
[28] P. E. Dunne. The computational complexity of ideal semantics. *Artificial Intelligence*, **173**, 1559–1591, 2009.
[29] P. E. Dunne and T. J. M. Bench-Capon. Coherence in finite argument systems. *Artificial Intelligence*, **141**, 187–203, 2002.
[30] P. E. Dunne and M.W.A. Caminada. Computational complexity of semi-stable semantics in abstract argumentation frameworks. In *Logics in Artificial Intelligence, 11th European Conference (JELIA 2008)*, Vol. 5293 of *Lecture Notes in Computer Science*, S. Hölldobler, C. Lutz, and H. Wansing, eds, pp. 153–165. Springer, 2008.
[31] W. Dvořák and S. Woltran. Complexity of semi-stable and stage semantics in argumentation frameworks. *Information Processing Letters*, **110**, 425–430, 2010.
[32] Th. Eiter, N. Leone, and D. Saccá. On the partial semantics for disjunctive deductive databases. *Annals of Mathematics and Artificial Intelligence*, **19**, 59–96, 1997.
[33] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Proceedings of the 5th International Conference/Symposium on Logic Programming*, R. A. Kowalski and K. Bowen, eds, pp. 1070–1080. MIT Press, 1988.
[34] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, **9**, 365–385, 1991.
[35] G. Governatori, M. J. Maher, G. Antoniou, and D. Billington. Argumentation semantics for defeasible logic. *Journal of Logic and Computation*, **14**, 675–702, 2004.

[36] B. Jenner and J. Toran. Computing functions with parallel queries to NP. *Theoretical Computer Science*, **141**, 175–193, 1995.
[37] N. Leone, G. Pfeifer, W. Faber, Th. Eiter, G. Gottlob, S. Perri, and F. Scarcello. The DLV system for knowledge representation and reasoning. *ACM Transactions on Computational Logic*, **7**, 499–562, 2006.
[38] W. Marek and M. Truszczynski. Stable semantics for logic programs and default theories. In *Proceedings of the North Americal Conference Logic Programming*, pp. 243–256. MIT Press, 1989.
[39] S. Modgil and M. W. A. Caminada. Proof theories and algorithms for abstract argumentation frameworks. In *Argumentation in Artificial Intelligence*, I. Rahwan and G. R. Simari, eds, pp. 105–129. Springer, 2009.
[40] I. Niemelä and P. Simons. Smodels - an implementation of the stable model and well-founded semantics for normal logic programs. In *Proceedings of the 4th International Conference on Logic Programming and Nonmonotonic Reasoning*, Vol. 1265 of *Lecture Notes in Artificial Intelligence*, J. Dix, U. Furbach, and A. Nerode, eds, pp. 420–429, 1997.
[41] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.
[42] J. L. Pollock. How to reason defeasibly. *Artificial Intelligence*, **57**, 1–42, 1992.
[43] J. L. Pollock. *Cognitive Carpentry. A Blueprint for How to Build a Person*. MIT Press, 1995.
[44] H. Prakken. Commonsense reasoning. *Technical report*, Institute of Information and Computing Sciences, Utrecht University, 2004.
[45] H. Prakken. An abstract framework for argumentation with structured arguments. *Argument and Computation*, **1**, 93–124, 2010.
[46] H. Prakken and G. Sartor. Argument-based extended logic programming with defeasible priorities. *Journal of Applied Non-Classical Logics*, **7**, 25–75, 1997.
[47] T. C. Przymusinski. The well-founded semantics coincides with the three-valued stable semantics. *Fundamenta Informaticae*, **13**, 445–463, 1990.
[48] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, **13**, 81–132, 1980.
[49] Ch. Sakama and K. Inoue. Paraconsistent stable semantics for extended disjunctive programs. *Journal of Logic and Computation*, **5**, 265–285, 1995.
[50] A. van Gelder, K. A. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *Journal of ACM*, **38**, 620–650, 1991.
[51] B. Verheij. Two approaches to dialectical argumentation: admissible sets and argumentation stages. In *Proceedings of the Eighth Dutch Conference on Artificial Intelligence (NAIC'96)*, J.-J. Ch. Meyer and L. C. van der Gaag, eds, pp. 357–368, 1996.
[52] J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1944.
[53] K. Wagner. Bounded query computations. In *Proceedings of the 3rd Conference on Structure in Complexity Theory*, pp. 260–277. IEEE Computer Society Press, 1988.
[54] K. Wagner. Bounded query classes. *SIAM Journal of Computation*, **19**, 833–846, 1990.
[55] Y. Wu, M. W. A. Caminada, and D. M. Gabbay. Complete extensions in argumentation coincide with 3-valued stable models in logic programming. *Studia Logica*, **93**, 383–403, 2009.
[56] J.-H. You and L.-Y. Yuan. A three-valued semantics for deductive databases and logic programs. *Journal of Computer System Sciences*, **49**, 334–361, 1994.

Received 18 March 2010