Instantiating Knowledge Bases in Abstract Dialectical Frameworks*

Hannes Strass

Computer Science Institute, Leipzig University

Abstract We present a translation from defeasible theory bases to abstract dialectical frameworks, a recent generalisation of abstract argumentation frameworks. Using several problematic examples from the literature, we first show how our translation addresses important issues of existing approaches. We then prove that the translated frameworks satisfy the rationality postulates closure and direct/indirect consistency. Furthermore, the frameworks can detect inconsistencies in the set of strict inference rules and cyclic (strict and defeasible) supports amongst literals. We also show that the translation involves at most a quadratic blowup and is therefore effectively and efficiently computable.

1 Introduction

Abstract argumentation frameworks (AFs) [1] are a formalism that is widely used in argumentation research. Such an AF consists of a set of arguments and an attack relation between these arguments. Their semantics determines which sets of arguments of a given AF can be accepted according to specific criteria. A common way to employ Dung's AFs is as abstraction formalism. In this view, expressive languages are used to model concrete argumentation scenarios, and translations into Dung AFs provide these original languages with semantics. The advantage of translating into an argumentation formalism is that the resulting semantics can be given a dialectical interpretation, which can be used to inform humans how a particular conclusion was inferred.

However, the approach is not without its problems. Caminada and Amgoud [2] reported some difficulties they encountered when defining an abstract argumentation-based semantics for defeasible theory bases. Defeasible theory bases are simple logic-inspired formalisms working with inference rules on a set of literals. Inference rules can be strict, in which case the conclusion of the inference (a literal) must necessarily hold whenever all antecedents (also literals) hold. Inference rules can also be defeasible, which means that the conclusion *usually* holds whenever the antecedents hold. Here, the word "usually" suggests that there could be exceptional cases where a defeasible rule has not been applied.

In response to the problems they encountered, Caminada and Amgoud [2] stated general rationality postulates for AFs based on defeasible theories. The intention of these postulates is to mathematically capture what humans perceive

* This research has been partially supported by DFG under project BR-1817/7-1.

as rational behaviour from the semantics of defeasible theory bases. First of all the *closure* postulate says that whatever model or extension the target formalism (the AF) produces, it must be closed under application of strict rules, meaning that all applicable strict rules have been applied. Direct and indirect *consistency* postulates express that any model or extension of the target formalism must be internally consistent with respect to the literals of the defeasible theory base (directly) and even with respect to application of strict rules (indirectly).

Later, Wyner et al. [3] criticised Caminada and Amgoud's definition of arguments on ontological grounds and gave an alternative translation. We are agnostic with respect to Wyner et al.'s criticism, but use their translation as a starting point for our own work. Such a further refinement is necessary since the translation of Wyner et al. [3] still yields unintuitive results on benchmark examples and does not satisfy the closure and indirect consistency postulates.

The basis of our solution to the aforementioned problems is a shift in the target language. While until now abstract argumentation frameworks were the formalism of choice, we will use the more general abstract *dialectical* frameworks (ADFs) [4]. Where AFs allow only attacks between arguments, ADFs can also represent support relations and many more. More specifically, in an AF an argument is accepted if none of its attackers is accepted. The same can be expressed in an ADF. But ADFs can also express that an argument is only accepted if all of its supporters are accepted, or the argument is accepted if *some* of its supporters are accepted, or it is accepted if some *attacker* is *not* accepted or ...

The expressiveness of ADFs in comparison to AFs – which we studied in [5,6] – enables us to give a direct and straightforward translation from defeasible theory bases to abstract dialectical frameworks. We will show that this translation – the main contribution of this paper – treats the benchmark examples right and satisfies the rationality postulates of Caminada and Amgoud [2]. We consider this further important evidence that abstract dialectical frameworks are useful tools for representing and reasoning about argumentation scenarios. We also perform a complexity analysis of our translation; this is significant in that we are not aware of complexity analyses of the mentioned previous approaches.

The availability of support in ADFs (in contrast to AFs) as a target formalism will be of fundamental importance to our translation. Among other things, it will allow us to resolve cyclic dependencies among literals in a defeasible theory base in a straightforward way. The treatment of such support cycles is built into ADF standard semantics, which can be considered a product of decades of research into nonmonotonic knowledge representation languages.

In the rest of the paper, we first recall the necessary background on defeasible theory bases, abstract argumentation frameworks and abstract dialectical frameworks. In Section 3 we look at the translations of Caminada and Amgoud [2] and Wyner et al. [3], discuss some problems of these, and introduce generalised versions of the rationality postulates. In Section 4 we then define our own translation. We show how it treats the problematic examples, prove that it satisfies the (generalised versions of the) rationality postulates and analyse its computational complexity. We conclude with a discussion of related and future work.

2 Background

Defeasible Theories. Following Caminada and Amgoud [2], we use a set *Lit* of literals that are built using syntactical negation $\neg\cdot$ and define a semantic negation function $\bar{\cdot}$ such that for an atom p we have $\overline{p} = \neg p$ and $\overline{\neg p} = p$. Throughout the paper, we assume that *Lit* is closed under negation in the sense that $\psi \in Lit$ implies $\overline{\psi} \in Lit$. A set $S \subseteq Lit$ of literals is *consistent* iff there is no literal $\psi \in Lit$ such that both $\psi \in S$ and $\neg\psi \in S$. For literals $\phi_1, \ldots, \phi_n, \psi \in Lit$, a *strict rule* over *Lit* is of the form $r : \phi_1, \ldots, \phi_n \to \psi$; a *defeasible rule* over *Lit* is of the form $r : \phi_1, \ldots, \phi_n \Rightarrow \psi$. (The only difference is the arrows.) Here r is the unique *rule name*, the literals $\phi_1, \ldots, \phi_n$ constitute the *rule body* and ψ is the *rule head* or *conclusion*. Intuitively, a strict rule says that the rule head is necessarily true whenever all body literals are true; a defeasible rule says that the head ψ is *usually* true whenever all body literals are true. In definitions, we use the symbol $\Rrightarrow$ as meta-level variable for $\to$ and $\Rightarrow$.

For a set $M \subseteq Lit$ of literals and a set *StrInf* of strict rules over *Lit*, we say that M is *closed under StrInf* iff $r : \phi_1, \ldots, \phi_n \to \psi \in StrInf$ and $\phi_1, \ldots, \phi_n \in M$ imply $\psi \in M$. Accordingly, the *closure of M under StrInf* is the smallest set $Cl_{StrInf}(M)$ that contains M and is closed under *StrInf*. A *defeasible theory* or *theory base* is a triple $(Lit, StrInf, DefInf)$ where *Lit* is a set of literals, *StrInf* is a set of strict rules over *Lit* and *DefInf* is a set of defeasible rules over *Lit*. The semantics of theory bases is usually defined via a translation to abstract argumentation frameworks, which will be introduced next.

Abstract Argumentation Frameworks. Dung [1] introduced argumentation frameworks as pairs $\Theta = (A, R)$ where A is a set and $R \subseteq A \times A$ a relation. The intended reading of an AF Θ is that the elements of A are arguments whose internal structure is abstracted away. The only information about the arguments is given by the relation R encoding a notion of attack: a pair $(a, b) \in R$ expresses that argument a attacks argument b in some sense.

The purpose of semantics for argumentation frameworks is to determine sets of arguments (called *extensions*) which are acceptable according to various standards. We will only be interested in so-called *stable* extensions, sets S of arguments that do not attack each other and attack all arguments not in the set. More formally, a set $S \subseteq A$ of arguments is *conflict-free* iff there are no $a, b \in S$ with $(a, b) \in R$. A set S is a *stable extension* for (A, R) iff it is conflict-free and for all $a \in A \setminus S$ there is a $b \in S$ with $(b, a) \in R$.

Abstract Dialectical Frameworks. Brewka and Woltran [4] introduced abstract dialectical frameworks as a powerful generalisation of Dung AFs that are able to capture not only attack and support, but also more general notions such as joint attack and joint support.

Definition 1. *An* abstract dialectical framework *is a triple $\Xi = (S, L, C)$ where*

- *S is a set of* statements*,*

– $L \subseteq S \times S$ *is a set of* links, *where* $par(s) \stackrel{\text{def}}{=} \{r \in S \mid (r, s) \in L\}$
– $C = \{C_s\}_{s \in S}$ *is a set of total functions* $C_s : 2^{par(s)} \to \{in, out\}$.

Intuitively, the function C_s for a statement s determines the acceptance status of s, which naturally depends on the status of its parent nodes. Any such function C_s can alternatively be represented by a propositional formula φ_s over the vocabulary $par(s)$. The understanding here is that for $M \subseteq par(s)$, $C_s(M) = in$ iff M is a model of φ_s (written $M \models \varphi_s$), where an interpretation is identified with the set of atoms that are evaluated to true.

Brewka and Woltran [4] introduced several semantical notions for ADFs. First, for an ADF $\Xi = (S, L, C)$ where C is given by a set of propositional formulas φ_s for each $s \in S$, a set $M \subseteq S$ is a *model for* Ξ iff for all statements s we have: $s \in M$ iff $M \models \varphi_s$.

Example 1 (Abstract dialectical framework). Consider the ADF $D = (S, L, C)$ with statements $S = \{a, b, c, d\}$, links $L = \{(a, c), (b, b), (b, c), (b, d)\}$ and acceptance functions given by the formulas $\varphi_a = \top$, $\varphi_b = b$, $\varphi_c = a \wedge b$ and $\varphi_d = \neg b$. Intuitively, these acceptance conditions express that (1) a is always accepted, (2) b supports itself, (3) c needs the joint support of a and b, and (4) d is attacked by b. The two models of D are $M_1 = \{a, b, c\}$ and $M_2 = \{a, d\}$.

In recent work [6], we redefined several standard ADF semantics and defined additional ones. In this paper, we are only interested in two-valued semantics, that is, models and stable models. The definition of the latter is based on the notion of a reduct and an operator originally introduced by Brewka and Woltran [4].

The operator Γ_Ξ takes two sets A, R of statements, where the intuition is that all statements in A are accepted and those in R are rejected. (So those in $S \setminus (A \cup R)$ are undecided.) According to these acceptance statuses, the operator evaluates all acceptance formulas and decides which statements can be definitely accepted or rejected.

The reduct implements the intuition that whatever is false in a stable model can be assumed false, but whatever is true in a stable model must be constructively provable. The next definition combines all of this.

Definition 2. *Let* $\Xi = (S, L, C)$ *be an abstract dialectical framework. Define an operator by* $\Gamma_\Xi(A, R) = (acc(A, R), rej(A, R))$ *for* $A, R \subseteq S$, *where*

$$acc(A, R) = \{s \in S \mid \text{for all } A \subseteq Z \subseteq (S \setminus R), \text{we have } Z \models \varphi_s\}$$
$$rej(A, R) = \{s \in S \mid \text{for all } A \subseteq Z \subseteq (S \setminus R), \text{we have } Z \not\models \varphi_s\}$$

For a set $M \subseteq S$, *define the reduced ADF* $\Xi_M = (M, L_M, C_M)$ *by the set of links* $L_M = L \cap M \times M$ *and for each* $s \in M$ *we set* $\varphi_{M,s} = \varphi_s[r/\bot : r \notin M]$.
A model M *for* Ξ *is a* stable model *of* Ξ *iff the least fixpoint of the operator* Γ_{Ξ_M} *is given by* (M, R) *for some* $R \subseteq S$.

Example 1 (Continued). Of the two models M_1, M_2 we have seen earlier, only M_2 is a stable model. Intuitively, the statement $b \in M_1$ cyclically supports itself.

It is clear that ADFs are a generalisation of AFs: for an argumentation framework $\Theta = (A, R)$, its *associated abstract dialectical framework* is $\Xi(\Theta) = (A, R, C)$ with $C_a(B) = in$ iff $B = \emptyset$ for each $a \in A$. But this is not just syntactical; Brewka and Woltran [4] showed that their semantical notions for ADFs are generalisations of Dung's respective AF notions; likewise, in [5,6] we proved correspondence results for all of the newly defined semantics. Brewka and Woltran [4] defined a particular subclass of ADFs called *bipolar*. Intuitively, in bipolar ADFs each link is supporting or attacking (or both). It will turn out that ADFs resulting from our automatic translation from defeasible theory bases are all bipolar.

3 Instantiations to Abstract Argumentation Frameworks

The general approach to provide a semantics for defeasible theories is to translate the defeasible theory into an argumentation formalism and then let the already existing semantics for that argumentation formalism determine the semantics of the defeasible theory. In the literature, the target formalism of choice are Dung's abstract argumentation frameworks. They abstract away from everything except arguments and attacks between them, so to define a translation to AFs one has to define arguments and attacks. We now review two particular such approaches.

3.1 The Approach of Caminada and Amgoud [2]

Caminada and Amgoud [2] define a translation from defeasible theories to argumentation frameworks. They create arguments in an inductive way by applying one or more inference rules. The internal structure of the arguments reflects how a particular conclusion was derived by applying an inference rule to the conclusions of subarguments, and allows arguments to be nested. So the base case of the induction takes into account rules with empty body, that is, rules of the form $\rightarrow \psi$ (or $\Rightarrow \psi$) for some literal ψ. Each such rule leads to an argument $A = [\rightarrow \psi]$ (or $[\Rightarrow \psi]$), and the conclusion of the rule becomes the conclusion of the argument. For the induction step, we assume there are arguments $A_1, \ldots, A_n$ with conclusions $\phi_1, \ldots, \phi_n$, respectively. If there is a strict rule $\phi_1, \ldots, \phi_n \rightarrow \psi$, we can build a new argument $A = [A_1, \ldots, A_n \rightarrow \psi]$ with conclusion ψ. (Likewise, from a defeasible rule $\phi_1, \ldots, \phi_n \Rightarrow \psi$ we can build a new argument $A = [A_1, \ldots, A_n \Rightarrow \psi]$.) Similar to rules, arguments can be strict or defeasible, where application of at least one defeasible rule makes the whole argument defeasible. (So strict arguments exclusively use strict rules.)

Caminada and Amgoud [2] then define two different kinds of attacks between arguments, rebuts and undercuts. An argument a rebuts another argument b if a subargument of a concludes some literal ψ, while there is a defeasible subargument of b that concludes $\overline{\psi}$. An argument a undercuts another argument b if the latter has a subargument that results from applying a defeasible rule and the applicability of that rule is disputed by a subargument of a.[1] In any case, we see that only defeasible arguments can be attacked.

[1] We will focus on rebuts in this paper since they are sufficient for our main points.

Example 2 (Married John, [2, Example 4]). Consider the following vocabulary with intended natural-language meaning: $w \ldots$ John wears something that looks like a wedding ring, $g \ldots$ John often goes out late with his friends, $m \ldots$ John is married, $b \ldots$ John is a bachelor, $h \ldots$ John has a wife. There are several relationships between these propositions, which are captured in the following theory base: the literals are $Lit = \{w, g, h, m, b, \neg w, \neg g, \neg h, \neg m, \neg b\}$, the strict rules are given by $StrInf = \{r_1 :\rightarrow w,\ r_2 :\rightarrow g,\ r_3 : b \rightarrow \neg h,\ r_4 : m \rightarrow h\}$ and the defeasible rules $DefInf = \{r_5 : w \Rightarrow m,\ r_6 : g \Rightarrow b\}$.

In the ASPIC system of Caminada and Amgoud [2], all the literals in the set $S = \{w, g, m, b\}$ are sceptical consequences of the constructed AF. Caminada and Amgoud observe that this is clearly unintended since the natural-language interpretation would be that John is a married bachelor. Moreover, the closure of S under $StrInf$ is $Cl_{StrInf}(S) = \{w, g, m, b, h, \neg h\}$, which is inconsistent. So not only are there applicable strict rules that have not been applied in S, but their application would lead to inconsistency.

To avoid anomalies such as the one just seen, Caminada and Amgoud [2] went on to define three natural rationality postulates for rule-based argumentation-based systems which are concerned with the interplay of consistency and strict rule application. Our formulation of them is slightly different for various reasons:

- We are concerned with argumentation frameworks as well as with abstract dialectical frameworks in this paper, so we made the postulates parametric in the target argumentation formalism.
- We removed the respective second condition on the sceptical conclusions with respect to all extensions/models. Propositions 4 and 5 in [2] show that they are redundant in their case.
- We are not constrained to formalisms and semantics where there are only finitely many extensions/models.
- For the sake of readability, we assume that the literals Lit of the defeasible theory are contained in the vocabulary of the target formalism.[2]

The first postulate requires that the set of conclusions for any extension should be closed under application of strict rules.

Postulate 1 (Closure) *Let $(Lit, StrInf, DefInf)$ be a defeasible theory. Its translation satisfies* closure *for semantics σ iff for any σ-model M, we find that $Cl_{StrInf}(Lit \cap M) \subseteq Lit \cap M$.*

Naturally, the notion of consistency is reduced to consistency of a set of literals of the underlying logical language. Note that consistency only concerns the local consistency of a given single model/extension of the target formalism. It may well be that the formalism is globally inconsistent in the sense of not allowing for any model with respect to a particular semantics. The latter behaviour can be desired, for example if the original theory base is inconsistent already.

[2] This is not a proper restriction since reconstruction of conclusions about the original defeasible theory is one of the goals of the whole enterprise and so there should be at least a translation function from argumentation models to theory models.

Postulate 2 (Direct Consistency) *Let* $(Lit, StrInf, DefInf)$ *be a defeasible theory with translation* X *and* σ *a semantics.* X *satisfies* direct consistency *iff for all* σ*-models* M *we have that* $Lit \cap M$ *is consistent.*

Caminada and Amgoud [2] remark that it is usually easy to satisfy direct consistency, but much harder to satisfy the stronger notion of indirect consistency. For this to hold, for each model its closure under strict rules must be consistent.

Postulate 3 (Indirect Consistency) *Let* $(Lit, StrInf, DefInf)$ *be a defeasible theory with translation* X *and* σ *a semantics.* X *satisfies* indirect consistency *iff for all* σ*-models* M *we have that* $Lit \cap Cl_{StrInf}(Lit \cap M)$ *is consistent.*

As a counterpart to Proposition 7 of Caminada and Amgoud [2], we can show that closure and direct consistency together imply indirect consistency.

Proposition 1. *Let* $(Lit, StrInf, DefInf)$ *be a defeasible theory with translation* X *and* σ *a semantics. If* X *satisfies closure and direct consistency, then it satisfies indirect consistency.*

Proof. Let X *satisfy closure and direct consistency, and let* M *be a* σ*-model for* X*. We have to show that* $Lit \cap Cl_{StrInf}(Lit \cap M)$ *is consistent. Since* X *satisfies closure,* $Cl_{StrInf}(Lit \cap M) \subseteq Lit \cap M$*. Thus* $Lit \cap Cl_{StrInf}(Lit \cap M) \subseteq Lit \cap M$*. Now since* X *satisfies direct consistency,* $Lit \cap M$ *is consistent. Hence its subset* $Lit \cap Cl_{StrInf}(Lit \cap M)$ *is consistent and* X *satisfies indirect consistency.* □

3.2 The Approach of Wyner et al. [3]

Wyner et al. [3] identified some problems of the approach of Caminada and Amgoud [2] and proposed an alternative translation from theory bases to argumentation frameworks. We do not necessarily support or reject their philosophical criticisms, but rather find the translation technically appealing. They create an argument for each literal in the theory base's language and additionally an argument for each rule. Intuitively, the literal arguments indicate that the literal holds, and the rule arguments indicate that the rule is applicable. Furthermore, the defined conflicts between these arguments are straightforward:

(1) opposite literals attack each other; (2) rules are attacked by the negations of their body literals; (3) defeasible rules are attacked by the negation of their head; (4) all rules attack the negation of their head.

Definition 3 ([3, Definitions 4,5]). *Let* $TB = (Lit, StrInf, DefInf)$ *be a defeasible theory. Define an argumentation framework* $\Theta(TB) = (A, R)$ *as follows.*

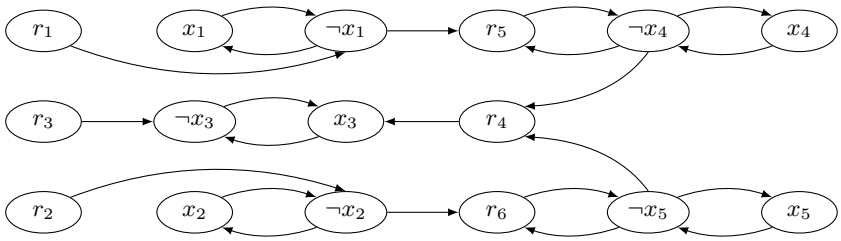
$$
\begin{aligned}
A = \quad & Lit \cup \{r \mid r : \phi_1, \ldots, \phi_n \Rrightarrow \psi \in StrInf \cup DefInf\} \\
R = \quad & \left\{(\psi, \overline{\psi}) \;\middle|\; \psi \in Lit\right\} \\
\cup \; & \left\{(\overline{\phi_i}, r) \;\middle|\; r : \phi_1, \ldots, \phi_n \Rrightarrow \psi \in StrInf \cup DefInf, 1 \leq i \leq n\right\} \\
\cup \; & \left\{(\overline{\psi}, r) \;\middle|\; r : \phi_1, \ldots, \phi_n \Rightarrow \psi \in DefInf\right\} \\
\cup \; & \left\{(r, \overline{\psi}) \;\middle|\; r : \phi_1, \ldots, \phi_n \Rrightarrow \psi \in StrInf \cup DefInf\right\}
\end{aligned}
$$

Let us look at one of their own examples which they adapted from [2].

Example 3 ([3, Example 5]). Consider the following theory base.

$$\begin{aligned} \mathit{Lit} &= \{x_1, x_2, x_3, x_4, x_5, \neg x_1, \neg x_2, \neg x_3, \neg x_4, \neg x_5\} \\ \mathit{StrInf} &= \{r_1 :\rightarrow x_1, \quad r_2 :\rightarrow x_2, \quad r_3 :\rightarrow x_3, \quad r_4 : x_4, x_5 \rightarrow \neg x_3\} \\ \mathit{DefInf} &= \{r_5 : x_1 \Rightarrow x_4, \quad r_6 : x_2 \Rightarrow x_5\} \end{aligned}$$

We can see that x_1, x_2, x_3 are strictly asserted and thus should be contained in any extension. The AF translation is depicted below.



The stable extensions of this AF are as follows:

$$S_1 = \{x_1, x_2, x_3, \neg x_4, \neg x_5, r_1, r_2, r_3\} \qquad S_2 = \{x_1, x_2, x_3, \neg x_4, x_5, r_1, r_2, r_3, r_6\}$$
$$S_3 = \{x_1, x_2, x_3, x_4, \neg x_5, r_1, r_2, r_3, r_5\} \quad S_4 = \{x_1, x_2, x_4, x_5, r_1, r_2, r_3, r_4, r_5, r_6\}$$

While the first three extensions can be considered intended, S_4 is not closed under strict rules and indirectly inconsistent: r_3 is applicable but x_3 does not hold, r_4 is applicable but $\neg x_3$ does not hold.

A similar observation can be made in Example 2: the AF translation according to Wyner et al. [3] has a stable extension $\{w, g, m, b, r_1, r_2, r_3, r_4, r_5, r_6\}$ where John is a married bachelor.

4 Instantiations to Abstract Dialectical Frameworks

In this section, we extend the theory base to AF translation of Wyner et al. [3] to ADFs. Due to the availability of support, this is straightforward. Indeed, support and attack are sufficient for our purposes.

4.1 From Theory Bases to ADFs

As in the approach of Wyner et al. [3], we directly use the literals from the theory base as statements that express whether the literal holds. We also use rule names as statements indicating that the rule is applicable. Additionally, for each rule r we create a statement -r indicating that the rule has not been applied. Not applying a rule is acceptable for defeasible rules, but unacceptable for strict rules since it would violate the closure postulate. This is enforced via

integrity constraints saying that it may not be the case in any model that the rule body holds but the head does not hold. Defeasible rules offer some degree of choice, whence we leave it to the semantics whether or not to apply them. This choice is modelled by a mutual attack cycle between r and $\text{-}r$. The remaining acceptance conditions are equally straightforward:

- Opposite literals attack each other.
- A literal is accepted whenever some rule deriving it is applicable, that is, all rules with head ψ support statement ψ.
- A strict rule is applicable whenever all of its body literals hold, that is, the body literals of r are exactly the supporters of r.
- Likewise, a defeasible rule is applicable whenever all of its body literals hold, and additionally the negation of its head literal must not hold.

In particular, literals cannot be accepted unless there is some rule deriving them.

Definition 4. *Let $TB = (Lit, StrInf, DefInf)$ be a theory base. Define an ADF $\Xi(TB) = (S, L, C)$ by $S = Lit \cup \{r, \text{-}r \mid r : \phi_1, \ldots, \phi_n \Rrightarrow \psi \in StrInf \cup DefInf\}$; the acceptance functions of statements s can be parsimoniously represented by propositional formulas φ_s.*[3] *For a literal $\psi \in Lit$, we define*

$$\varphi_\psi = \neg[\overline{\psi}] \wedge \bigvee_{r:\phi_1,\ldots,\phi_n \Rrightarrow \psi \in StrInf \cup DefInf} [r]$$

For a strict rule $r : \phi_1, \ldots, \phi_n \to \psi \in StrInf$, we define

$$\varphi_r = [\phi_1] \wedge \ldots \wedge [\phi_n] \qquad \textit{and} \qquad \varphi_{\text{-}r} = [\phi_1] \wedge \ldots \wedge [\phi_n] \wedge \neg[\psi] \wedge \neg[\text{-}r]$$

For a defeasible rule $r : \phi_1, \ldots, \phi_n \Rightarrow \psi \in DefInf$, we define

$$\varphi_r = [\phi_1] \wedge \ldots \wedge [\phi_n] \wedge \neg[\overline{\psi}] \wedge \neg[\text{-}r] \qquad \textit{and} \qquad \varphi_{\text{-}r} = \neg[r]$$

Finally, there is a link $(s', s) \in L$ iff $[s']$ occurs in the acceptance formula φ_s.

For strict rules with name r, the self-attack of $\text{-}r$ does not materialise whenever either the rule body is not satisfied or the rule head holds; otherwise the strict rule is applicable but has not been applied and the constraint $\text{-}r$ prevents this undesirable state of affairs from getting included in a model. (For the formulas defined above, the empty disjunction leads to $\bot$ – logical falsity – and the empty conjunction to $\top$ – logical truth.)

Let us see how our translation treats the examples seen earlier.

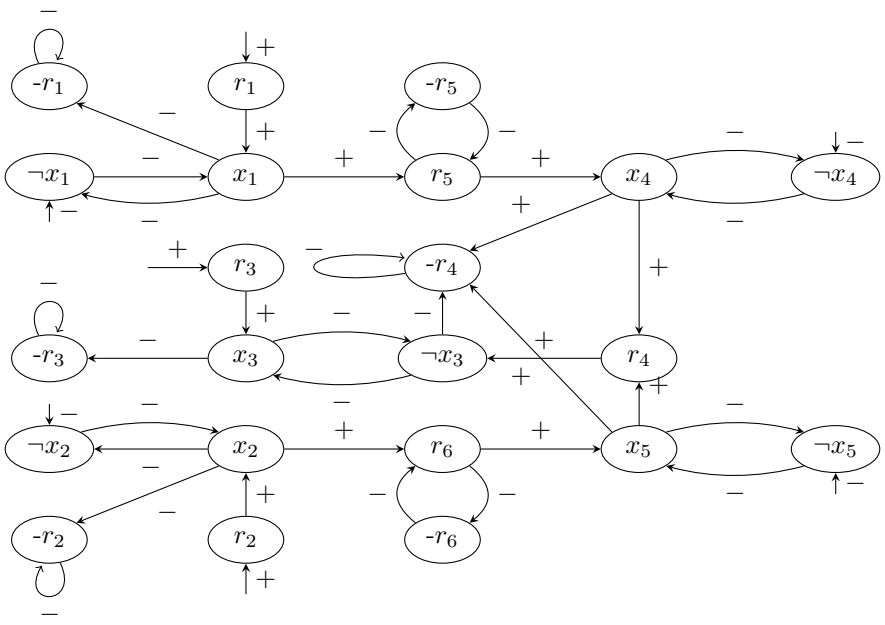
[3] In these formulas, we write ADF statements in brackets, to avoid confusion between negation being applied inside a statement name – as in $[\neg x]$ – and negation being applied in the formula outside of the statement's name – as in $\neg[\text{-}r]$. Thus $[\neg x]$ and $\neg[x]$ are syntactically different literals in the language of acceptance formulas; their meaning is intertwined via the semantics of ADFs.

Example 3 (Continued). Definition 4 yields the following acceptance formulas.

$$\begin{array}{llll}
\varphi_{x_1} = \neg[\neg x_1] \wedge [r_1] & \varphi_{x_2} = \neg[\neg x_2] \wedge [r_2] & \varphi_{x_3} = \neg[\neg x_3] \wedge [r_3] & \\
\varphi_{x_4} = \neg[\neg x_4] \wedge [r_5] & \varphi_{x_5} = \neg[\neg x_5] \wedge [r_6] & & \\
\varphi_{\neg x_1} = \bot \quad \varphi_{\neg x_2} = \bot & \varphi_{\neg x_3} = \neg[x_3] \wedge [r_4] & \varphi_{\neg x_4} = \bot \quad \varphi_{\neg x_5} = \bot & \\
\varphi_{r_1} = \top \quad \varphi_{r_2} = \top & \varphi_{r_3} = \top & \varphi_{r_4} = [x_4] \wedge [x_5] & \\
\varphi_{r_5} = [x_1] \wedge \neg[\neg x_4] \wedge \neg[\text{-}r_5] & \varphi_{r_6} = [x_2] \wedge \neg[\neg x_5] \wedge \neg[\text{-}r_6] & & \\
\varphi_{\text{-}r_1} = \neg[x_1] \wedge \neg[\text{-}r_1] & \varphi_{\text{-}r_2} = \neg[x_2] \wedge \neg[\text{-}r_2] & \varphi_{\text{-}r_3} = \neg[x_3] \wedge \neg[\text{-}r_3] & \\
\varphi_{\text{-}r_4} = [x_4] \wedge [x_5] \wedge \neg[\neg x_3] \wedge \neg[\text{-}r_4] & & \varphi_{\text{-}r_5} = \neg[r_5] \quad \varphi_{\text{-}r_6} = \neg[r_6] &
\end{array}$$

Statements with an acceptance condition of the form $\neg p_1 \wedge \ldots \wedge \neg p_n$ behave like AF arguments. So in particular r_1, r_2, r_3 are always *in* since these rules have an empty body. Similarly, $\text{-}r_1, \text{-}r_2, \text{-}r_3$ are self-attacking arguments. The statements $\neg x_1, \neg x_2, \neg x_4, \neg x_5$ are always *out* since there are no rules deriving these literals. The remaining acceptance conditions are clear from the definitions: literals are supported by the rules deriving them and rules in turn are supported by their body literals.

For illustration, we also provide the ADF in form of a labelled graph, where the labels + and − indicate supporting and attacking links. Several statements have constant truth values as acceptance conditions, in the picture this is indicated via a link from the surroundings.[4]



[4] This is inspired by conventions from automata theory, where initial states are indicated likewise.

For this ADF, models and stable models coincide, and there are three of them:

$$M_1 = \{x_1, x_2, x_3, r_1, r_2, r_3, \text{-}r_5, \text{-}r_6\} \qquad M_2 = \{x_1, x_2, x_3, x_4, r_1, r_2, r_3, r_5, \text{-}r_6\}$$
$$M_3 = \{x_1, x_2, x_3, x_5, r_1, r_2, r_3, \text{-}r_5, r_6\}$$

Roughly, in M_1 none of the defeasible rules r_5, r_6 has been applied – indicated by $\text{-}r_5$ and $\text{-}r_6$ –, while in M_2 and M_3 either one of them has been applied. As intended, there is no model where both defeasible rules have been applied, as this would lead to a set that contains both x_4 and x_5; this in turn would make rule r_4 applicable, allowing to conclude $\neg x_3$ in contradiction to x_3 being strictly true according to rule r_3.

We can furthermore see that all of the models are closed under strict rule application (they contain x_1, x_2, x_3 and no other strict rule is applicable) and directly consistent, thus also indirectly consistent.

A similar observation can be made for John (not) being married (Example 2); our ADF translation has three (stable) models: $M_1 = \{w, g, r_1, r_2, \text{-}r_5, \text{-}r_6\}$, $M_2 = \{w, g, h, m, r_1, r_2, r_4, r_5, \text{-}r_6\}$ and $M_3 = \{w, g, b, \neg h, r_1, r_2, r_3, \text{-}r_5, r_6\}$. Again, the argumentation translation of the theory base satisfies closure and direct and indirect consistency. We will later prove that the satisfaction of the postulates is not a coincidence in our approach. But first of all let us consider another problem which often arises in knowledge representation and reasoning.

4.2 Support Cycles in Theory Bases

When logical, rule-based approaches are used for knowledge representation, a recurring issue is that of cyclic dependencies between propositions of the knowledge base. If such support cycles are carelessly overlooked or otherwise not treated in an adequate way, they can lead to counterintuitive conclusions. Consider this famous example from Denecker et al. [7].

Example 4 (Gear Wheels [7]). There are two interlocked gear wheels x and y that can be separately turned and stopped. Let x_0 and y_0 denote whether x (resp. y) turns at time point 0, and likewise for a successive time point 1. At any one time point, whenever the first wheel turns (resp. stops), it causes the second one to turn (resp. stop), and vice versa. This is expressed by strict rules r_1 to r_8. Without a cause for change, things usually stay the way they are from one time point to the next, which is expressed by the defeasible rules r_a to r_d.

$$\begin{aligned}
\mathit{Lit} &= \{x_0, y_0, x_1, y_1, \neg x_0, \neg y_0, \neg x_1, \neg y_1\} \\
\mathit{StrInf} &= \{r_1 : x_0 \to y_0, \quad r_2 : y_0 \to x_0, \quad r_3 : \neg x_0 \to \neg y_0, \quad r_4 : \neg y_0 \to \neg x_0, \\
&\qquad r_5 : x_1 \to y_1, \quad r_6 : y_1 \to x_1, \quad r_7 : \neg x_1 \to \neg y_1, \quad r_8 : \neg y_1 \to \neg x_1\} \\
\mathit{DefInf} &= \{r_a : x_0 \Rightarrow x_1, \quad r_b : \neg x_0 \Rightarrow \neg x_1, \quad r_c : y_0 \Rightarrow y_1, \quad r_d : \neg y_0 \Rightarrow \neg y_1\}
\end{aligned}$$

For later reference, we denote this theory base by $TB_{GW} = (\mathit{Lit}, \mathit{StrInf}, \mathit{DefInf})$. To model a concrete scenario, we add the rules $\mathit{StrInf}' = \{r_i :\to \neg x_0, r_j :\to \neg y_0\}$

expressing that both wheels initially stand still. We denote the augmented theory base for this concrete scenario by $TB'_{GW} = (Lit, StrInf \cup StrInf', DefInf)$. It is clearly unintended that there is some model for TB'_{GW} where the gear wheels magically start turning with one being the cause for the other and vice versa.

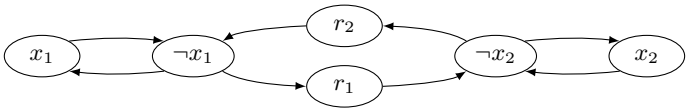
Example 5. Consider the following defeasible rules saying that *rain* and *wet* grass usually go hand in hand: $Lit = \{rain, wet, \neg rain, \neg wet\}$, $StrInf = \emptyset$ and $DefInf = \{r_1 : rain \Rightarrow wet, r_2 : wet \Rightarrow rain\}$. The intended meaning is that one is usually accompanied by the other, not that both may appear out of thin air.

To see how argumentation translations of theory bases treat such cycles, let us look at a simplified version of the gear wheels example.

Example 6. Consider a theory base with two literals mutually supporting each other through strict rules: $Lit = \{x_1, x_2, \neg x_1, \neg x_2\}$, the strict rules are given by $StrInf = \{r_1 : x_1 \to x_2, \quad r_2 : x_2 \to x_1\}$ and $DefInf = \emptyset$. Our ADF translation of this example yields the acceptance formulas

$$\varphi_{x_1} = [r_2] \qquad \varphi_{\neg x_1} = \bot \qquad \varphi_{r_1} = [x_1] \qquad \varphi_{\text{-}r_1} = [x_1] \wedge \neg[x_2] \wedge \neg[\text{-}r_1]$$
$$\varphi_{x_2} = [r_1] \qquad \varphi_{\neg x_2} = \bot \qquad \varphi_{r_2} = [x_2] \qquad \varphi_{\text{-}r_2} = [x_2] \wedge \neg[x_1] \wedge \neg[\text{-}r_2]$$

The ADF has two models, $M_1 = \{x_1, x_2, r_1, r_2\}$ and $M_2 = \emptyset$. Only M_2 is a stable model due to the cyclic self-support of the statements in M_1. Note that not only do x_1 and x_2 not hold in M_2, neither do $\neg x_1$ and $\neg x_2$ (there are no rules possibly deriving them). In contrast, the translation of Wyner et al. [3] yields the AF



with two stable extensions $S_1 = \{x_1, r_1, x_2, r_2\}$ and $S_2 = \{\neg x_1, \neg x_2\}$. In S_1, x_1 and x_2 hold due to self-support while in S_2 they are "guessed" to be false.

In our view, this is problematic since it is not made clear to the user that these different extensions arise due to self-support. Even if we grant that for some application domains, cyclic self-support of literals might be intended or at least not unintended, the user should be able to distinguish whether different models/extensions arise due to present or absent self-support on the one hand, or due to conflicts between defeasible conclusions on the other hand. ADFs provide this important distinction, since cycles are allowed in models and disallowed in stable models, while both semantics are identical in their treatment of conflicts between defeasible conclusions.

In the approach of Caminada and Amgoud [2], treatment of cycles is built into the definition of the set of arguments in the resulting argumentation framework. The arguments are created using structural induction, where rules with empty bodies form the induction base and all other rules form the induction step. For the general gear wheel domain TB_{GW} of Example 4, and for Examples 5 and

6, their translation would not create any arguments (there are no assertions in the theory bases), and the approach could not draw any conclusions about these examples. The concrete scenario of the interlocked gear wheel domain TB'_{GW} in Example 4, where both wheels initially stand still, would be treated correctly by the approach of Caminada and Amgoud [2]. But note that the well-foundedness of the treatment of cyclic dependencies is built into the syntax of the resulting argumentation framework – there are no arguments that could conclude that any of the wheels is turning, although there are (strict and defeasible) rules with such conclusions. Consequently, a part of the semantics of the theory base is already fixed by the translation, irrespective of the argumentation semantics that is used later on.

4.3 Inconsistent Theory Bases

Example 7 (Inconsistent Theory Base). Consider the following (obviously inconsistent) theory base in which both a literal and its negation are strictly asserted: $\mathit{Lit} = \{x, \neg x\}$, $\mathit{StrInf} = \{r_1 :\rightarrow x,\ r_2 :\rightarrow \neg x\}$ and $\mathit{DefInf} = \emptyset$. Our ADF translation yields the acceptance formulas

$$\begin{array}{lll} \varphi_x = \neg[\neg x] \wedge [r_1] & \varphi_{r_1} = \top & \varphi_{\text{-}r_1} = \neg[x] \wedge \neg[\text{-}r_1] \\ \varphi_{\neg x} = \neg[x] \wedge [r_2] & \varphi_{r_2} = \top & \varphi_{\text{-}r_2} = \neg[\neg x] \wedge \neg[\text{-}r_2] \end{array}$$

This ADF has no models, and so the theory base's inconsistency is detected.

On the other hand, the associated argumentation framework due to Wyner et al. [3] is given by the set of arguments $A = \{x, \neg x, r_1, r_2\}$ and the attacks $R = \{(x, \neg x), (\neg x, x), (r_1, \neg x), (r_2, x)\}$. In the only stable extension $\{r_1, r_2\}$ both rules are applicable but none of the head literals hold due to immanent conflict.

In the approach of Caminada and Amgoud [2], we can construct two strict arguments that conclude x and $\neg x$, respectively. There are no attacks between these arguments since they are both strict. The resulting AF has a stable extension from which both x and $\neg x$ can be concluded, which detects the inconsistency.

4.4 Properties of the Translation

In this section, we analyse some theoretical properties of our translation. First we show that it satisfies (our reformulations of) the rationality postulates of Caminada and Amgoud [2]. Then we analyse the computational complexity of translating a given theory base and show that the blowup is at most quadratic.

Postulates. It is elementary to show that the ADFs resulting from our translation satisfy direct consistency. This is because the statements ψ and $\overline{\psi}$ mutually attack each other.

Proposition 2. *For any theory base $TB = (\mathit{Lit}, \mathit{StrInf}, \mathit{DefInf})$, its associated ADF $\Xi(TB)$ satisfies direct consistency with respect to the model semantics.*

Proof. *Let M be a model for $\Xi(TB)$ and assume to the contrary that $M \cap Lit$ is inconsistent. Then there is a $\psi \in Lit$ such that $\psi \in M$ and $\neg\psi \in M$. Since $\neg\psi \in M$, the acceptance condition of $\neg\psi$ yields $\psi \notin M$. Contradiction.* □

We can also prove that they satisfy closure: by construction, the (acceptance conditions of) statements -r for strict rules r guarantee that the rule head is contained in any model that contains the rule body.

Proposition 3. *For any theory base $TB = (Lit, StrInf, DefInf)$, its associated ADF $\Xi(TB)$ satisfies closure with respect to the model semantics.*

Proof. *Let M be a model of $\Xi(TB)$ and $r : \phi_1, \ldots, \phi_n \rightarrow \psi \in StrInf$ such that we find $\phi_1, \ldots, \phi_n \in M$. We have to show $\psi \in M$. By definition, $\Xi(TB)$ has a statement -r with parents $par(\text{-}r) = \{\phi_1, \ldots, \phi_n, \psi, \text{-}r\}$. We next show that -$r \notin M$: assume to the contrary that -$r \in M$. Then by the acceptance condition of -r we get -$r \notin M$, contradiction. Thus -$r \notin M$. Now the acceptance condition of -r yields $\phi_1 \notin M$ or $\ldots$ or $\phi_n \notin M$ or $\psi \in M$ or -$r \in M$. By assumption, we have $\phi_1, \ldots, \phi_n \in M$ and -$r \notin M$, thus we get $\psi \in M$.* □

By Proposition 1 the translation satisfies indirect consistency.

Corollary 1. *For any theory base $TB = (Lit, StrInf, DefInf)$, its associated ADF $\Xi(TB)$ satisfies indirect consistency with respect to the model semantics.*

Since any stable model is a model, our translation also satisfies the postulates for the stable model semantics.

Corollary 2. *For any theory base $TB = (Lit, StrInf, DefInf)$, its associated ADF $\Xi(TB)$ satisfies closure and direct and indirect consistency with respect to the stable model semantics.*

It should be noted that defeasible rules may or may not be applied – the approach is not eager to apply defeasible rules.

Complexity. For a theory base $TB = (Lit, StrInf, DefInf)$, we define the size of its constituents as follows. Quite straightforwardly, the size of a set of literals is just its cardinality, the size of a rule is the number of literals in it, the size of a set of rules is the sum of the sizes of its elements and the size of a theory base is the sum of the sizes of its components.

We want to analyse the size of its ADF translation $\Xi(TB) = (S, L, C)$ according to Definition 4. Clearly, the number of statements is linear in the size of the theory base, since we have one statement for each literal and two statements for each rule: $|S| = |Lit| + 2 \cdot (|StrInf| + |DefInf|)$. Since $L \subseteq S \times S$, the number of links in L is at most quadratic in the cardinality of S: $|L| \leq |S|^2$. Finally, we have seen in Definition 4 that the acceptance conditions of statements can be parsimoniously represented by propositional formulas. It can be checked that the size of each one of these formulas is at most linear in the size of the theory base. Since there are linearly many statements with one acceptance formula each, the

acceptance conditions can be represented in quadratic space. So overall, the resulting ADF $\Xi(TB) = (S, L, C)$ can be represented in space which is at most quadratic in the size of the original theory base. In particular, in our approach a finite theory base always yields a finite argumentation translation. This is in contrast to the definition of Caminada and Amgoud [2], where the strict rule set $StrInf = \{r_0 :\rightarrow a, r_1 : a \rightarrow b, r_2 : b \rightarrow a\}$ allows to construct infinitely many arguments $A_1 = [\rightarrow a], A_2 = [A_1 \rightarrow b], A_3 = [A_2 \rightarrow a], A_4 = [A_3 \rightarrow b], \ldots$[5]

5 Conclusion

We presented a translation from theory bases to abstract dialectical frameworks. The translated frameworks satisfy the rationality postulates closure and direct/indirect consistency, which we generalised to make them independent of a specific target formalism. Furthermore, the translated frameworks can detect inconsistencies in the rule base and cyclic supports amongst literals. We also showed that the translation involves at most a quadratic blowup and is therefore effectively computable. Furthermore, our translation produces a number of statements which is linear in the size of the theory base and can be considered efficient in this regard. (In the approach of [2] the number of produced arguments is unbounded in general.) In terms of desired behaviour, we compared our translation to previous approaches from the literature [2,3] and demonstrated how we avoid common problems.

In earlier work, Brewka and Gordon [8] translated Carneades [9] argument evaluation structures (directly) to ADFs. They extended the original Carneades formalism by allowing cyclic dependencies among arguments. Meanwhile, Van Gijzel and Prakken [10] also translated Carneades into AFs (via ASPIC+ [11], that extends and generalises the definitions of Caminada and Amgoud [2]). They can deal with cycles, but there is only one unique grounded, preferred, complete, stable extension. Thus the semantic richness of abstract argumentation is not used, and the user cannot choose whether they want to accept or reject circular justifications of arguments. In contrast, in the approach of Brewka and Gordon [8] the user can decide whether cyclic justifications should be allowed or disallowed, by choosing models or *stable* models as ADF semantics.

We regard this work as another piece of evidence that abstract dialectical frameworks are well-suited as target formalisms for translations from less directly accessible languages such as theory bases. A natural next step would be to consider as input the specification language of ASPIC+ [11]. A recent approach to preferences between statements [6] might be a good starting point for this. Further work could also encompass the study of additional ADF semantics, like complete or preferred models [6], and whether the approach can be modified

[5] Even if we exclude cycles in rules, there are rule sets which allow for exponentially many arguments: Set $D_0 = \{\Rightarrow p_0, \Rightarrow \neg p_0\}$, $D_1 = D_0 \cup \{p_0 \Rightarrow p_1, \neg p_0 \Rightarrow p_1\}$ and for $i \geq 1$, $D_{i+1} = D_i \cup \{p_0, p_i \Rightarrow p_{i+1}, \neg p_0, p_i \Rightarrow p_{i+1}\}$. For any $n \in \mathbb{N}$, the size of D_n is linear in n and D_n leads to 2^{n+1} arguments, among them 2^n arguments for p_n.

such that it is eager to apply defeasible rules. Finally, we can compare existing approaches to cycles in AFs [12,13] with the treatment of cycles in ADFs.

Acknowledgements. The author is grateful to Gerhard Brewka for informative discussions. He also thanks Leila Amgoud and Martin Caminada for clarifying some aspects of the ASPIC framework.

References

1. Dung, P.M.: On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-Person Games. Artificial Intelligence **77** (1995) 321–358
2. Caminada, M., Amgoud, L.: On the evaluation of argumentation formalisms. Artificial Intelligence **171**(5–6) (2007) 286–310
3. Wyner, A., Bench-Capon, T., Dunne, P.: Instantiating knowledge bases in abstract argumentation frameworks. In: Proceedings of the AAAI Fall Symposium – The Uses of Computational Argumentation. (2009)
4. Brewka, G., Woltran, S.: Abstract Dialectical Frameworks. In: Proceedings of the Twelfth International Conference on the Principles of Knowledge Representation and Reasoning (KR). (2010) 102–111
5. Strass, H.: Approximating operators and semantics for Abstract Dialectical Frameworks. Technical Report 1, Institute of Computer Science, Leipzig University (January 2013)
6. Brewka, G., Ellmauthaler, S., Strass, H., Wallner, J.P., Woltran, S.: Abstract Dialectical Frameworks Revisited. In: Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, AAAI Press (August 2013) To appear.
7. Denecker, M., Theseider-Dupré, D., Van Belleghem, K.: An Inductive Definition Approach to Ramifications. Linköping Electronic Articles in Computer and Information Science **3**(7) (January 1998) 1–43
8. Brewka, G., Gordon, T.F.: Carneades and Abstract Dialectical Frameworks: A Reconstruction. In: Computational Models of Argument: Proceedings of COMMA 2010, Desenzano del Garda, Italy, September 8-10, 2010. Volume 216 of Frontiers in Artificial Intelligence and Applications., IOS Press (2010) 3–12
9. Gordon, T.F., Prakken, H., Walton, D.: The Carneades model of argument and burden of proof. Artificial Intelligence **171**(10–15) (2007) 875–896
10. Van Gijzel, B., Prakken, H.: Relating Carneades with Abstract Argumentation. In: Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence – Volume Two, AAAI Press (2011) 1113–1119
11. Prakken, H.: An abstract framework for argumentation with structured arguments. Argument & Computation **1**(2) (2010) 93–124
12. Baroni, P., Giacomin, M., Guida, G.: SCC-recursiveness: A general schema for argumentation semantics. Artificial Intelligence **168**(1–2) (2005) 162–210
13. Baroni, P., Dunne, P.E., Giacomin, M.: On the resolution-based family of abstract argumentation semantics and its grounded instance. Artificial Intelligence **175**(3–4) (2011) 791–813