TLP **15** (4–5): 434–448, 2015. © Cambridge University Press 2015
doi:10.1017/S1471068415000149

Improved answer-set programming encodings for abstract argumentation

SARAH A. GAGGL and NORBERT MANTHEY
Technische Universität Dresden, Germany

ALESSANDRO RONCA
La Sapienza, University of Rome

JOHANNES P. WALLNER
University of Helsinki, Department of Computer Science, HIIT

STEFAN WOLTRAN
Vienna University of Technology, Austria

submitted 29 April 2015; revised 3 July 2015; accepted 15 July 2015

Abstract

The design of efficient solutions for abstract argumentation problems is a crucial step towards advanced argumentation systems. One of the most prominent approaches in the literature is to use Answer-Set Programming (ASP) for this endeavor. In this paper, we present new encodings for three prominent argumentation semantics using the concept of conditional literals in disjunctions as provided by the ASP-system clingo. Our new encodings are not only more succinct than previous versions, but also outperform them on standard benchmarks.

KEYWORDS: Answer-Set Programming, Abstract Argumentation, Implementation, ASPARTIX

1 Introduction

Abstract Argumentation (Dung 1995; Rahwan and Simari 2009) is at the heart of many advanced argumentation systems (Besnard and Hunter 2008; Caminada and Amgoud 2007) and is concerned with finding jointly acceptable arguments by taking only their inter-relationships into account. Efficient solvers for abstract argumentation are thus an important development, a fact that is also witnessed by a new competition which takes place in 2015 for the first time (Cerutti *et al.* 2014)[1].

[1] See `http://argumentationcompetition.org` for further information.

https://doi.org/10.1017/S1471068415000149 Published online by Cambridge University Press

To date, several approaches for implementing abstract argumentation exist, many of them following the so-called reduction-based (see (Charwat *et al.* 2015)) paradigm: hereby, existing efficient software which has originally been developed for other purposes is used. Prominent examples for this approach are (i) the CSP-based system ConArg (Bistarelli and Santini 2011), (ii) SAT-based approaches (e.g. (Cerutti *et al.* 2014; Dvořák *et al.* 2014)) and (iii) systems which rely on Answer-Set Programming (ASP); see (Toni and Sergot 2011) for a comprehensive survey. In fact, ASP (Brewka *et al.* 2011) is particularly well-suited since ASP systems by default enumerate all solutions of a given program, thus enabling the enumeration of extensions of an abstract argumentation framework in an easy manner. Moreover, disjunctive ASP is capable of expressing problems being even complete for the 2nd level of the polynomial hierarchy. In fact, several semantics for abstract argumentation like preferred, semi-stable (Caminada *et al.* 2012), or stage (Verheij 1996) are of this high complexity (Dunne and Bench-Capon 2002; Dvořák and Woltran 2010).

One particular candidate for an ASP reduction-based system is ASPARTIX (Egly *et al.* 2010; Dvořák *et al.* 2013). Here, a fixed program for each semantics is provided and the argumentation framework under consideration is just added as an input-database. The program together with the input-database is then handed over to an ASP system of choice in order to calculate the extensions. This makes the ASPARTIX approach easy to adapt and an appealing rapid-prototyping method. The proposed encodings in ASPARTIX for the high-complexity semantics mentioned above come, however, with a certain caveat. This stems from the fact that encodings for such complex programs have to follow a certain saturation pattern, where restricted use of cyclic negation has to be taken care of (we refer to (Egly *et al.* 2010) for a detailed discussion). The original encodings followed the definition of the semantics quite closely and thus resulted in quite complex and tricky loop-techniques which are a known feature for ASP experts, but hard to follow for ASP laymen. Moreover, experiments in other domains indicated that such loops also potentially lead to performance bottlenecks.

In this work, we thus aim for new and simpler encodings for the three semantics of preferred, semi-stable, and stage extensions. To this end, we provide some alternative characterizations for these semantics and design our new encodings along these characterizations in such a way that costly loops are avoided. Instead we make use of the ASP language feature of conditional literals in disjunction (Syrjänen 2009; Gebser *et al.* 2015). Moreover, we perform exhaustive experimental evaluation against the original ASPARTIX-encodings, the ConArg system, and another ASP-variant (Dvořák *et al.* 2013) which makes use of the ASP front-end *metasp* (Gebser *et al.* 2011), where the required maximization is handled via meta-programming. Our results show that the new ASP encodings not only outperform the previous variants, but also makes ASPARTIX more powerful than ConArg.

The novel encodings together with the benchmark instances are available under `http://dbai.tuwien.ac.at/research/project/argumentation/systempage/#conditional`.

https://doi.org/10.1017/S1471068415000149 Published online by Cambridge University Press

2 Background

2.1 Abstract argumentation

First, we recall the main formal ingredients for argumentation frameworks (Dung 1995; Baroni *et al.* 2011) and survey relevant complexity results (see also (Dunne and Wooldridge 2009)).

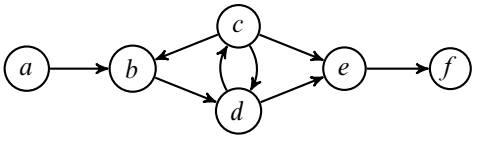
Definition 1
An *argumentation framework (AF)* is a pair $F = (A, R)$ where A is a set of arguments and $R \subseteq A \times A$ is the attack relation. The pair $(a, b) \in R$ means that a attacks b. An argument $a \in A$ is *defended* by a set $S \subseteq A$ if, for each $b \in A$ such that $(b, a) \in R$, there exists a $c \in S$ such that $(c, b) \in R$. We define the *range of* S (w.r.t. R) as $S_R^+ = S \cup \{x \mid \exists y \in S \text{ such that } (y, x) \in R\}$.

Semantics for argumentation frameworks are given via a function σ which assigns to each AF $F = (A, R)$ a set $\sigma(F) \subseteq 2^A$ of extensions. We shall consider here for σ the functions *stb*, *adm*, *prf*, *stage*, and *sem* which stand for stable, admissible, preferred, stage, and semi-stable semantics respectively.

Definition 2
Let $F = (A, R)$ be an AF. A set $S \subseteq A$ is *conflict-free (in F)*, if there are no $a, b \in S$, such that $(a, b) \in R$. $cf(F)$ denotes the collection of conflict-free sets of F. For a conflict-free set $S \in cf(F)$, it holds that

- $S \in stb(F)$, if $S_R^+ = A$;
- $S \in adm(F)$, if each $s \in S$ is defended by S;
- $S \in prf(F)$, if $S \in adm(F)$ and there is no $T \in adm(F)$ with $T \supset S$;
- $S \in sem(F)$, if $S \in adm(F)$ and there is no $T \in adm(F)$ with $T_R^+ \supset S_R^+$;
- $S \in stage(F)$, if there is no $T \in cf(F)$ in F, such that $T_R^+ \supset S_R^+$.

Example 1
Consider the AF $F = (A, R)$ with $A = \{a, b, c, d, e, f\}$ and $R = \{(a, b), (b, d), (c, b), (c, d), (c, e), (d, c), (d, e), (e, f)\}$, and the graph representation of F:



We have $stb(F) = stage(F) = sem(F) = \{\{a, d, f\}, \{a, c, f\}\}$. The admissible sets of F are $\emptyset$, $\{a\}$, $\{c\}$, $\{a, c\}$, $\{a, d\}$, $\{c, f\}$, $\{a, c, f\}$, $\{a, d, f\}$, and $prf(F) = \{\{a, c, f\}, \{a, d, f\}\}$.

We recall that each AF F possesses at least one preferred, semi-stable, and stage extension, while $stb(F)$ might be empty. However, it is well known that $stb(F) \neq \emptyset$ implies $stb(F) = stage(F) = sem(F)$ as also seen in the above example.

Next, we provide some alternative characterisations for the semantics of our interest. They will serve as the basis of our encodings.

The alternative characterisation for preferred extensions relies on the following idea. An admissible set S is preferred, if each other admissible set E (which is not a subset of S) is in conflict with S.

https://doi.org/10.1017/S1471068415000149 Published online by Cambridge University Press

Proposition 1
Let $F = (A, R)$ be an AF and $S \subseteq A$ be admissible in F. Then, $S \in prf(F)$ if and only if, for each $E \in adm(F)$ such that $E \not\subseteq S$, $E \cup S \notin cf(F)$.

Proof
Let $S \in adm(F)$ and assume there exists an admissible (in F) set $E \not\subseteq S$, such that $E \cup S \in cf(F)$. It is well known (see, e.g. (Dunne *et al.* 2014), Lemma 1) that if two sets E_1, E_2 defend themselves in an AF F, then also $E_1 \cup E_2$ defends itself in F. It follows that $E \cup S \in adm(F)$ and by assumption $S \subset E \cup S$. Thus, $S \notin prf(F)$. For the other direction, let $S \in adm(F)$ but $S \notin prf(F)$. Hence, there exists an $S' \supset S$ such that $S' \in adm(F)$. Clearly, $S' \not\subseteq S$ but $S' = (S \cup S') \in cf(F)$. □

We turn to semi-stable and stage semantics. In order to verify whether a candidate extension S is a stage (resp. semi-stable) extension of an AF F, we check whether for any set S' such that $S' \supset S_R^+$ there is no conflict-free (resp. admissible) set E such that $S' \subseteq E_R^+$. We also show that is sufficient to check this for minimal such sets S'. Observe that the above check is trivially true if S is already stable, mirroring the observation that $stb(F) = stage(F) = sem(F)$ whenever $stb(F) \neq \emptyset$.

Definition 3
Let $F = (A, R)$ be an AF and $S \subseteq A$. A *cover* of S in F is any $E \subseteq A$ such that $S \subseteq E_R^+$. The set of covers of S in F is denoted by $\Gamma_F(S)$.

Proposition 2
Let $F = (A, R)$ be an AF and $S \in cf(F)$ (resp. $S \in adm(F)$). The following propositions are equivalent: (1) S is a stage (resp. semi-stable) extension of F; (2) for each $a \in A \setminus S_R^+$, there is no $E \in \Gamma_F(S_R^+ \cup \{a\})$ such that $E \in cf(F)$ (resp. $E \in adm(F)$); (3) for each S' with $S_R^+ \subset S' \subseteq A$, there is no $E \in \Gamma_F(S')$, such that $E \in cf(F)$ (resp. $E \in adm(F)$).

Proof
We give the proof for stage extensions. The result for semi-stable proceeds analogously. (1)⇒(3): Suppose there is an S' with $S_R^+ \subset S' \subseteq A$, such that some $E \in \Gamma_F(S')$ is conflict-free in F. By definition, $S_R^+ \subset S' \subseteq E_R^+$. Hence, $S \notin stage(F)$. (2)⇒(1): Suppose $S \notin stage(F)$. Thus there exists $T \in cf(F)$ with $S_R^+ \subset T_R^+$. Let $a \in T_R^+ \setminus S_R^+$. It follows that $T \in \Gamma_F(S \cup \{a\})$. (3)⇒(2) is clear. □

Finally, we turn to the complexity of reasoning in AFs for two major decision problems. For a given AF $F = (A, R)$ and an argument $a \in A$, credulous reasoning under σ denotes the problem of deciding whether there exists an $E \in \sigma(F)$ s.t. $a \in E$. Skeptical Acceptance under σ is the problem of deciding whether for all $E \in \sigma(F)$ it holds that $a \in E$. Credulous reasoning for preferred semantics is NP-complete, while credulous reasoning for semi-stable and stage semantics is Σ_2^P-complete. For preferred, semi-stable, and stage semantics skeptical reasoning is Π_2^P-complete (Dung 1995; Dimopoulos and Torres 1996; Dunne and Bench-Capon 2002; Dunne and Caminada 2008; Dvořák and Woltran 2010).

https://doi.org/10.1017/S1471068415000149 Published online by Cambridge University Press

2.2 Answer-set programming

We give an overview of the syntax and semantics of disjunctive logic programs under the answer-sets semantics (Gelfond and Lifschitz 1991).

We fix a countable set $\mathcal{U}$ of *(domain) elements*, also called *constants*; and suppose a total order $<$ over the domain elements. An *atom* is an expression $p(t_1,\ldots,t_n)$, where p is a *predicate* of arity $n \geqslant 0$ and each t_i is either a variable or an element from $\mathcal{U}$. An atom is *ground* if it is free of variables. $B_{\mathcal{U}}$ denotes the set of all ground atoms over $\mathcal{U}$. A *(disjunctive) rule* r is of the form

$$a_1 \mid \cdots \mid a_n \leftarrow b_1,\ldots,b_k,\ not\ b_{k+1},\ldots,\ not\ b_m \tag{1}$$

with $n \geqslant 0$, $m \geqslant k \geqslant 0$, $n+m>0$, where $a_1,\ldots,a_n,b_1,\ldots,b_m$ are literals, and "*not*" stands for *default negation*. The *head* of r is the set $H(r)=\{a_1,\ldots,a_n\}$ and the *body* of r is $B(r)=\{b_1,\ldots,b_k,\ not\ b_{k+1},\ldots,\ not\ b_m\}$. Furthermore, $B^+(r)=\{b_1,\ldots,b_k\}$ and $B^-(r)=\{b_{k+1},\ldots,b_m\}$. A rule r is *normal* if $n \leqslant 1$ and a *constraint* if $n=0$. A rule r is *safe* if each variable in r occurs in $B^+(r)$. A rule r is *ground* if no variable occurs in r. A *fact* is a ground rule without disjunction and empty body. An *(input) database* is a set of facts. A program is a finite set of disjunctive rules. For a program π and an input database D, we often write $\pi(D)$ instead of $D \cup \pi$. If each rule in a program is normal (resp. ground), we call the program normal (resp. ground).

For any program π, let U_π be the set of all constants appearing in π. $Gr(\pi)$ is the set of rules $r\sigma$ obtained by applying, to each rule $r \in \pi$, all possible substitutions σ from the variables in r to elements of U_π. An *interpretation* $I \subseteq B_{\mathcal{U}}$ *satisfies* a ground rule r iff $H(r) \cap I \neq \emptyset$ whenever $B^+(r) \subseteq I$ and $B^-(r) \cap I = \emptyset$. I satisfies a ground program π, if each $r \in \pi$ is satisfied by I. A non-ground rule r (resp., a program π) is satisfied by an interpretation I iff I satisfies all groundings of r (resp., $Gr(\pi)$). $I \subseteq B_{\mathcal{U}}$ is an *answer set* of π iff it is a subset-minimal set satisfying the *Gelfond-Lifschitz reduct* $\pi^I = \{H(r) \leftarrow B^+(r) \mid I \cap B^-(r) = \emptyset, r \in Gr(\pi)\}$. For a program π, we denote the set of its answer sets by $\mathcal{AS}(\pi)$.

Modern ASP solvers offer additional language features. Among them we make use of the *conditional literal* (Syrjänen 2009; Gebser *et al.* 2015). In the head of a disjunctive rule literals may have conditions, e.g. consider the head of rule "$\mathbf{p}(X) : \mathbf{q}(X) \leftarrow$". Intuitively, this represents a head of disjunctions of atoms $\mathbf{p}(a)$ where also $\mathbf{q}(a)$ is true.

2.3 ASP encodings for AFs

For our novel encodings we utilize basic encodings for AFs, conflict-free sets, and admissible sets from (Egly *et al.* 2010). An AF is represented as a set of facts.

Definition 4
Let $F=(A,R)$ be an AF. We define $\hat{F} = \{\mathbf{arg}(a) \mid a \in A\} \cup \{\mathbf{att}(a,b) \mid (a,b) \in R\}$.

In the following definition we first formalize the correspondence between an extension, as subset of arguments, and an answer set of an ASP encoding; then we extend it to the one between sets of extensions and answer sets respectively.

https://doi.org/10.1017/S1471068415000149 Published online by Cambridge University Press

Listing 1. Module π_{cf}

```
in(X) ← arg(X), not out(X).                    1
out(X) ← arg(X), not in(X).                    2
← att(X,Y), in(X), in(Y).                      3
```

Listing 2. Module π_{def}

```
defeated(X) ← in(Y), att(Y,X).                 1
undefended(X) ← att(Y,X), not defeated(Y).     2
← in(X), undefended(X).                        3
```

Definition 5
Let $\mathscr{S} \subseteq 2^{\mathscr{U}}$ be a collection of sets of domain elements and let $\mathscr{I} \subseteq 2^{B_{\mathscr{U}}}$ be a collection of sets of ground atoms. We say that $S \in \mathscr{S}$ and $I \in \mathscr{I}$ correspond to each other, in symbols $S \cong I$, iff $S = \{a \mid \mathbf{in}(a) \in I\}$. We say that $\mathscr{S}$ and $\mathscr{I}$ correspond to each other, in symbols $\mathscr{S} \cong \mathscr{I}$, iff (i) for each $S \in \mathscr{S}$, there exists an $I \in \mathscr{I}$, such that $I \cong S$; and (ii) for each $I \in \mathscr{I}$, there exists an $S \in \mathscr{S}$, such that $S \cong I$.

It will be convenient to use the following notation and result later in Section 3.

Definition 6
Let $I, J \in 2^{B_{\mathscr{U}}}$ be sets of ground atoms. We say that I and J are equivalent, in symbols $I \equiv J$, iff $\{\mathbf{in}(a) \mid \mathbf{in}(a) \in I\} = \{\mathbf{in}(a) \mid \mathbf{in}(a) \in J\}$.

Lemma 3
Let $I, J \in 2^{B_{\mathscr{U}}}$, and $S \in 2^{\mathscr{U}}$. If $I \equiv J$ and $I \cong S$, then $J \cong S$.

In Listing 1 we see the ASP encoding for conflict-free sets, while Listing 2 shows defense of arguments. The encoding for admissible sets is given by $\pi_{adm} = \pi_{cf} \cup \pi_{def}$. The following has been proven in (Egly *et al.* 2010, Proposition 3.2).

Proposition 4
For any AF $F = (A, R)$, and any $I \in \mathscr{AS}(\pi_{cf}(\hat{F}))$, $\mathscr{P} = \{\{a \mid \mathbf{in}(a) \in I\}, \{a \mid \mathbf{out}(a) \in I\}\}$ is a partition of A.

Correctness of the encodings π_{cf} and π_{adm} was proven in (Egly *et al.* 2010).

Proposition 5
For any AF F, we have (i) $cf(F) \cong \mathscr{AS}(\pi_{cf}(\hat{F}))$, and (ii) $adm(F) \cong \mathscr{AS}(\pi_{adm}(\hat{F}))$.

Next, we characterize the encoding π_{range} (Listing 3), which, given a module computing some extension S (via **in**) of an AF (A, R), returns its range S_R^+ (via **range**) and also collects the arguments not contained in the range. We indicate via **unstable** that S is not stable, i.e. $S_R^+ \subset A$.

https://doi.org/10.1017/S1471068415000149 Published online by Cambridge University Press

Listing 3. Module π_{range}

```
range(X) ← in(X).                                  1
range(Y) ← in(X), att(X,Y).                        2
out_of_range(X) ← not range(X), arg(X).            3
unstable ← out_of_range(X), arg(X).                4
```

Listing 4. Module π_{eq}

```
eq_upto(Y) ← inf(Y), in(Y), inN(Y).                1
eq_upto(Y) ← inf(Y), out(Y), outN(Y).              2
eq_upto(Y) ← succ(Z,Y), in(Y), inN(Y), eq_upto(Z). 3
eq_upto(Y) ← succ(Z,Y), out(Y), outN(Y), eq_upto(Z). 4
eq ← sup(Y), eq_upto(Y).                           5
```

Lemma 6

Let $F = (A, R)$ be an AF, and π be a program not containing the predicates **range**$(\cdot)$, **out_of_range**$(\cdot)$ and **unstable**. Let $I \subseteq B_A$ and $S \subseteq A$ s.t. $I \cong S$. Furthermore let $\pi^+ = \pi \cup \pi_{range}$ and

$$I^+ = I \cup \{\textbf{range}(a) \mid a \in S_R^+\} \cup \{\textbf{out_of_range}(a) \mid a \in A \setminus S_R^+\} \cup \{\textbf{unstable} \mid S_R^+ \subset A\}. \tag{2}$$

Then, $I \in \mathscr{AS}(\pi(\hat{F}))$, if and only if $I^+ \in \mathscr{AS}(\pi^+(\hat{F}))$.

The preferred, semi-stable (Egly *et al.* 2010) and stage semantics (Dvořák *et al.* 2013) utilize the so-called *saturation technique*. We sketch here the basic ideas. Intuitively, in the saturation technique encoding for preferred semantics we make a first guess for a set of arguments in the framework, and then we verify if this set is admissible (via module π_{adm}). To verify if this set is also subset maximal admissible, a second guess is carried out via a disjunctive rule. If this second guess corresponds to an admissible set that is a proper superset of the first one, then the first one cannot be a preferred extension. Using the saturation technique now ensures that if all second guesses "fail" to be a strictly larger admissible set of the first guess, then there is one answer-set corresponding to this preferred extension. Usage of default negation within the saturation technique for the second guess is restricted, and thus a loop-style encoding is employed that checks if the second guess is admissible and a proper superset of the first guess.

Roughly, a loop construct in ASP checks a certain property for the least element in a set (here we use the predicate **inf**$(\cdot)$), and then checks this property "iteratively" for each (immediate) successor (via predicate **succ**$(\cdot,\cdot)$). If the property holds for the greatest element (**sup**$(\cdot)$), it holds for all elements. In Listing 4 we illustrate loop encodings, where we see a partial ASP encoding used for preferred semantics in (Egly *et al.* 2010) that derives **eq** if the first and second guesses are equal, i.e. the predicates corresponding to the guesses via **in**$(\cdot)$, resp. **out**$(\cdot)$, and **inN**$(\cdot)$, resp. **outN**$(\cdot)$, are true for the same constants.

Another variant of ASP encodings for preferred, semi-stable and stage semantics is developed by (Dvořák *et al.* 2013). There so-called meta-asp encodings are used,

https://doi.org/10.1017/S1471068415000149 Published online by Cambridge University Press

Listing 5. Module π_{satprf^2}

```
nontrivial ← out(X).                                          1
witness(X):out(X) ← nontrivial.                               2
spoil | witness(Z):att(Z,Y) ← witness(X), att(Y,X).           3
spoil ← witness(X), witness(Y), att(X,Y).                     4
spoil ← in(X), witness(Y), att(X,Y).                          5
witness(X) ← spoil, arg(X).                                   6
← not spoil, nontrivial.                                      7
```

which allow for minimizing statements w.r.t. subset inclusion directly in the ASP language (Gebser *et al.* 2011). For instance, π_{adm} can then be augmented with a minimizing statement on the predicate **out**, to achieve an encoding of preferred semantics.

3 Encodings

Here we present our new encodings for preferred, semi-stable, and stage semantics via the novel characterizations.

3.1 Encoding for preferred semantics

The encoding for preferred semantics is given by $\pi_{prf^2} = \pi_{adm} \cup \pi_{satprf^2}$, where π_{satprf^2} is provided in Listing 5. We first give the intuition of the program. A candidate S for being preferred in an AF $F = (A, R)$ is computed by the program π_{adm} via the **in**$(\cdot)$ predicate, and is already known admissible. If all arguments in A are contained in S we are done[2]. Otherwise, the remainder of the program π_{satprf^2} (Lines 2 and 7) is used to check whether there exists a set $E \in adm(F)$ such that $E \not\subseteq S$ and not in conflict with S. We start to build E by guessing some argument not contained in S (Line 2) and then in Line 3 we repeatedly add further arguments to E unless the set defends itself (otherwise we eventually derive **spoil**). Then, we check whether E is conflict-free (Line 4) and E is not in conflict with S (Line 5). If we are able to reach this point without deriving **spoil**, then the candidate S cannot be an answer-set (Line 7). This is in line with Proposition 1, which states that in this case S is not preferred.

By inspecting Listing 5 we also see important differences w.r.t. the encodings for preferred semantics of (Egly *et al.* 2010). In our new encodings, the "second guess" via predicate **witness**$(\cdot)$ is constructed through conditional disjunction instead of simple disjunction. Usage of the former allows to construct the witness set already with defense of arguments in mind. Furthermore loops, such as the one shown in Listing 4 that checks if the second guess is equal to first one or a loop construct that checks if every argument is defended, can be avoided, since these checks are partially incorporated into Line 2 of Listing 5 and into simpler further checks.

Correctness of this new encoding is stated and proved in the following proposition.

[2] Note, this is only the case when there are no attacks in F.

https://doi.org/10.1017/S1471068415000149 Published online by Cambridge University Press

Proposition 7

For any AF F, we have $prf(F) \cong \mathscr{AS}(\pi_{prf^2}(\hat{F}))$.

Proof

According to Definition 5, we have to prove (i) and (ii). With line numbers we refer here to the ASP encoding shown in Listing 5. We employ the *splitting theorem* (Lifschitz and Turner 1994) in order to get a characterisation of $\mathscr{AS}(\pi_{prf^2}(\hat{F}))$, in which the sub-programs π_{satprf^2} and π_{adm} are considered separately. The splitting set is $C_{prf^2} = \{\mathbf{arg}(\cdot), \mathbf{att}(\cdot,\cdot), \mathbf{in}(\cdot), \mathbf{out}(\cdot), \mathbf{defeated}(\cdot), \mathbf{undefended}(\cdot)\}$, and we obtain

$$\mathscr{AS}(\pi_{prf^2}(\hat{F})) = \bigcup_{J \in \mathscr{AS}(\pi_{adm}(\hat{F}))} \mathscr{AS}(J \cup \pi_{satprf^2}). \tag{3}$$

Proof (i). We prove that each preferred extension $S \in prf(F)$ has a corresponding answer-set $I \in \mathscr{AS}(\pi_{prf^2}(\hat{F}))$. From Equation (3) we know that $I \in \mathscr{AS}(\pi_{prf^2}(\hat{F}))$ if $I \in \mathscr{AS}(J \cup \pi_{satprf^2})$, for some $J \in \mathscr{AS}(\pi_{adm}(\hat{F}))$. Moreover $S \in prf(F)$ implies $S \in adm(F)$, hence by Proposition 5 there is $J \in \mathscr{AS}(\pi_{adm}(\hat{F}))$ s.t. $J \cong S$. In the following we distinguish between two complementary cases.

In case $R = \emptyset$, the set $S = A$ is the only preferred one, since it is trivially admissible and it cannot be contained in another set of arguments. We show $I = J$ is a subset-minimal model of $(J \cup \pi_{satprf^2})^J$. The subset-minimality is evident. Then, $\mathbf{out}(a) \notin J$ for any $a \in A$ by Proposition 4, hence J satisfies the rule at Line 1. Since $\mathbf{nontrivial} \notin J$, J satisfies the rules at Lines 2, and 7. Every other rule is satisfied because $\mathbf{att}(a,b) \notin \hat{F}$ for any $a, b \in A$.

In case $R \neq \emptyset$ we can build an interpretation I and prove that I is an answer-set by contraposition, i.e. if there is an $L \subset I$ which satisfies $(J \cup \pi_{satprf^2})^I$, then $S \notin prf(F)$. We define $I = J \cup \{\mathbf{spoil}, \mathbf{nontrivial}\} \cup \{\mathbf{witness}(a) \mid a \in A\}$. We have $I \cong S$ since $I \equiv J$. The set I satisfies $(J \cup \pi_{satprf^2})^I$ (got from $Gr(J \cup \pi_{satprf^2})$ by just removing the rule at Line 7), as $J \subseteq I$ and I contains all the heads of the rules in $(J \cup \pi_{satprf^2})^I$. Notice that $R \neq \emptyset$ guarantees that the head of the rule at Line 2 is non-empty.

Now we describe the necessary shape of L, in order to prove the main assertion next. L must contain **nontrivial** because of the rule at Line 1. Indeed $\mathbf{out}(c) \in L$ for some $c \in A \setminus S$, since $J \subseteq L$ with $J \cong S$ and $S \in cf(F)$ (since $S \in adm(F)$), which implies the existence of $c \in A \setminus S$ (we cannot have simultaneously $R \neq \emptyset$, $S \in cf(F)$ and $S = A$), which implies $\mathbf{out}(c) \in J$ by Proposition 4. We have $\mathbf{spoil} \notin L$, otherwise also $\{\mathbf{witness}(a) \mid a \in A\}$ would be in L (because of the rule at Line 6), making L equal to I, but they are different by assumption.

Now we show that, given L, it is possible to find a set $U \in adm(F)$ s.t. $U \not\subseteq S$ and $U \cup S \in cf(F)$, which implies $S \notin prf(F)$ by Proposition 1. We define $U = \{a \mid \mathbf{witness}(a) \in L\}$, and we show all the required properties:

$U \in cf(F)$, otherwise we would have two arguments a, b attacking each other, meaning $\{\mathbf{witness}(a), \mathbf{witness}(b), \mathbf{att}(a,b)\} \subseteq L$, which implies $B(r) \subseteq L$ and $H(r) \notin L$ for some rule r in the grounding of the rule at Line 3, since $\mathbf{spoil} \notin L$.

https://doi.org/10.1017/S1471068415000149 Published online by Cambridge University Press

Each $a \in U$ is defended by U, otherwise it would be possible to find two atoms **witness**$(a) \in L$ $[a \in U]$[3] and **att**$(b,a) \in L$ $[(b,a) \in R]$ for which there is no **witness**$(c) \in L$ $[c \in U]$ s.t. **att**$(c,b) \in L$ $[(c,b) \in R]$, thus violating the rule at Line 4, since **spoil** $\notin L$.

$U \not\subseteq S$. Indeed if we assume $U \subseteq S$, then for every **witness**$(a) \in L$ we have $a \in S$ (by definition of U), which corresponds to **in**$(a) \in J$ ($S \cong J$), implying **out**$(a) \notin J$ (by Proposition 4), making it impossible for L to satisfy the rule at Line 2, since **nontrivial** $\in L$.

$\{U \cup S\} \in cf(F)$. The sets U and S are conflict-free, so we have to show that there cannot be attack relations between the two sets: an argument $a \in S$ cannot attack an argument $b \in U$, otherwise we would have $\{$**witness**(b), **in**(a), **att**$(a,b)\} \subseteq L$, which implies $B(r) \subseteq L$ and $H(r) \notin L$ for some rule in the grounding of the rule at Line 5, since **spoil** $\notin L$; an argument $b \in U$ cannot attack an argument $a \in S$, otherwise an argument $c \in S$ should attack b by admissibility of S, thus violating the previous point.

Proof (ii). We prove that each $I \in \mathscr{AS}(\pi_{prf^2}(\hat{F}))$ corresponds to an $S \in prf(F)$. From Equation (3) we see that $I \in \mathscr{AS}(\pi_{prf^2}(\hat{F}))$ only if $I \in \mathscr{AS}(J \cup \pi_{satprf^2})$ for some $J \in \mathscr{AS}(\pi_{adm}(\hat{F}))$. We have $I \equiv J$, because $J \subseteq I$, and I does not have any additional ground atom **in**(a), since **in**$(\cdot)$ does not appear in the head of any rule of π_{satprf^2}. By Proposition 5 there exists $S \in adm(F)$ s.t. $S \cong J$, hence $S \cong I$ by Lemma 3. We show that S is also preferred in F, by distinguishing between two complementary cases.

nontrivial $\notin I$: we have **out**$(a) \notin I$ for any $a \in A$, otherwise the rule at Line 1 would be violated. By Proposition4 this implies **in**$(a) \in I$ for every $a \in A$, and the same is true for J ($J \cong I$), which we know to be admissible. Hence, $S = A$ and $S \in prf(F)$.

nontrivial $\in I$: we prove that S is preferred by contraposition, i.e. if $S \notin prf(F)$ then I is not a subset-minimal model of $(J \cup \pi_{satprf^2})^I$. We have that I must have a clear shape in order to satisfy $(J \cup \pi_{satprf^2})^I$. In particular $J \subseteq I$. Then **spoil** $\in I$ because of the rule at Line 7 hence, **witness**$(a) \in I$ for each **arg**$(a) \in I$ because of the rule at Line 6. Summing up we have $J \cup \{$**nontrivial**, **spoil**$\} \cup \{$**witness**$(a) \mid a \in A\} \subseteq I$. Finally we show that $I \notin \mathscr{AS}(J \cup \pi_{satprf^2})$, since we are able to build an interpretation $L \subset I$ satisfying the reduct $(J \cup \pi_{satprf^2})^I$. We remind that $S \notin prf(F)$ means that there exists $T \in prf(F)$ s.t. $S \subset T$. We use T to build the interpretation $L = J \cup \{$**nontrivial**$\} \cup \{$**witness**$(a) \mid a \in T\}$. We have $L \subset I$, because it does not contain **spoil** and $T \subseteq A$. In the following we show that L is a model of the reduct, because it contains J and it satisfies each rule in $Gr(\pi_{satprf^2})$.

L satisfies the rule at Line 2, because there exists **witness**$(a) \in L$ s.t. **out**$(a) \in L$, for some $a \in T \setminus S$ (the element a exists because T is a proper superset of S). [4]

[3] In this proof, the square brackets are used to point out an immediate implication of the statement preceding them. Usually the statement is about the framework F and the implication about an interpretation, or the other way around.

[4] If $a \in T \setminus S$, then $a \notin S$, then **in**$(a) \notin I$ ($S \cong I$), then **in**$(a) \notin J$ ($I \equiv J$), then **out**$(a) \in J$ (by Proposition 4), then **out**$(a) \in L$ ($J \subset L$). Summing up, if $a \in T \setminus S$, then **out**$(a) \in L$, and **witness**$(a) \in L$ by definition.

https://doi.org/10.1017/S1471068415000149 Published online by Cambridge University Press

Listing 6. Module π_{satsem^2}

```
larger_range(X) : out_of_range(X) ← unstable.                          1
larger_range(X) ← range(X), unstable.                                  2
witness(X) | witness(Z) : att(Z,X) ← larger_range(X), unstable.        3
spoil ← witness(X), witness(Y), att(X,Y), unstable.                    4
spoil | witness(Z) : att(Z,Y) ← witness(X), att(Y,X), unstable.        5
witness(X) ← spoil, arg(X), unstable.                                  6
larger_range(X) ← spoil, arg(X), unstable.                             7
← not spoil, unstable.                                                 8
```

Since T is admissible, for each $a \in T$ [**witness**$(a) \in L$] attacked by $b \in A$ [**att**$(b,a) \in \hat{F}$] there exists $c \in T$ [**witness**$(c) \in L$] attacking b [**att**$(c,b) \in \hat{F}$]. Hence L satisfies the rule at Line 3, even though **spoil** $\notin L$.

L does not contain the body of any rule in the grounding of the rule at Line 4, otherwise T would not be conflict free. L does not contain the body of any rule in the grounding of the rule at Line 5, otherwise T would not be conflict free, since $S \subset T$. L does not contain the body of any rule in the grounding of the rule at Line 6, because it does not contain **spoil**. □

3.2 *Encodings for semi-stable and stage semantics*

Semi-stable semantics The encoding for semi-stable semantics is given by $\pi_{sem^2} = \pi_{adm} \cup \pi_{range} \cup \pi_{satsem^2}$, with π_{satsem^2} shown in Listing 6. We first give the intuition. A candidate S for being semi-stable is computed by the program $\pi^{+}_{adm} = \pi_{adm} \cup \pi_{range}$ via the **in**$(\cdot)$ predicate and is known admissible. The module π_{range} computes the range and derives **unstable** iff the extension is not stable. If S is stable, we are done. Otherwise the remainder of the program π_{satsem^2} is used to check whether an admissible cover E of a superset of the range S^{+}_{R} exists. Starting from S^{+}_{R} (Line 2), a superset is achieved by adding at least one element out of it (Line 1). Then a cover is found (Line 3), which is admissible (Lines 3 and 4). If we are able to reach this point without deriving **spoil** (that is always a possibility for satisfying the constraints), then the candidate S cannot be an answer-set (Line 3). This is in line with Proposition 2, which states that in this case S is not semi-stable. Here we state the correctness of the encoding, a full proof is given in the online appendix (Appendix A).

Proposition 8
For any AF $F = (A, R)$, we have $sem(F) \cong \mathscr{AS}(\pi_{sem^2}(\hat{F}))$.

Stage semantics The encoding for stage semantics is given by $\pi_{stage^2} = \pi_{cf} \cup \pi_{range} \cup \pi_{satsem^2} \setminus \{r_{admcov}\}$, where r_{admcov} is the rule at Line 5 of Listing 6. The only differences w.r.t. the encoding for semi-stable semantics are: (i) it employs π_{cf} instead of π_{adm}, thus the candidate sets are only conflict-free; and (ii) it lacks the rule at Line 5, hence it considers all the conflict-free covers of the candidate set, which is still in line with Proposition 2. A proof sketch for the forthcoming correctness result is given in the online appendix (Appendix A).

Proposition 9
For any AF $F = (A, R)$, we have $stage(F) \cong \mathscr{AS}(\pi_{stage^2}(\hat{F}))$.

https://doi.org/10.1017/S1471068415000149 Published online by Cambridge University Press

Table 1. *Summary of test results*

PR	usc	solved	med	SST	usc	solved	med	STG	usc	solved	med
ConArg	60	2814	43.65	ConArg	50	3509	1.03	ConArg	–	606	600.00
Original	–	3425	180.36	Original	–	3386	211.96	Original	–	2185	600.00
Meta	1	4626	20.83	Meta	2	4830	17.70	Meta	5	2419	600.00
New	101	4765	5.77	New	13	4879	3.30	New	82	2501	384.92

4 Evaluation

We tested the novel encodings (NEW) extensively and compared them to the original (ORIGINAL) and metasp (META) encodings as well as to the system ConArg (Bistarelli and Santini 2011). For the novel and original encodings we used *Clingo 4.4* and for the metasp encodings we used *gringo3.0.5/clasp3.1.1* all from the Potassco group[5]. As benchmarks, we considered a collection of frameworks which have been used by different colleagues for testing before consisting of structured and randomly generated AFs, resulting in 4972 frameworks. In particular we used parts of the instances Federico Cerutti provided to us which have been generated towards an increasing number of SCCs (Vallati *et al.* 2014). Further benchmarks were used to test the system *dynpartix* and we included the instances provided by the ICCMA 2015 organizers. The full set is available at `http://dbai.tuwien.ac.at/research/project/argumentation/systempage/#conditional`.

For each framework the task is to enumerate all solutions. The computation has been performed on an Intel Xeon E5-2670 running at 2.6 GHz. From the 16 available cores we used only every fourth core to allow a better utilization of the CPU's cache. We applied a 10 minutes timeout, allowing to use at most 6.5 GB of main memory.

It turns out that for each semantics the new encodings significantly outperform the original ones as well as the system ConArg. Furthermore, there is a clear improvement to the metasp encodings, as illustrated in Fig. 1 which shows the cactus plots of the required runtime to solve frameworks (x-axis) with the respective timeout (y-axis) for the three discussed semantics. While for preferred and semi-stable semantics the novel encodings are able to solve more than 4700 instances (out of 4972), one can observe a different trend for stage semantics. There, the new encodings return the best result with 2501 solved instances. Table 1 gives a summary of the test results, where *usc* denotes the unique solver contribution, i.e. the number of AFs which could only be solved by the particular solver, *solved* gives the number of solved instances by the solver, and *med* is the median of the computation time of the solver. Interestingly, ConArg is able to solve 60 (resp. 50) instances for preferred (resp. semi-stable) semantics which are not solvable by the other systems. However, the novel encodings are able to uniquely solve 101 (resp. 82) instances for preferred (resp. stage) semantics. The original encodings have no *unique solver contribution* for all of the considered semantics, thus it is save to replace them with the new

[5] `http://potassco.sourceforge.net`

https://doi.org/10.1017/S1471068415000149 Published online by Cambridge University Press

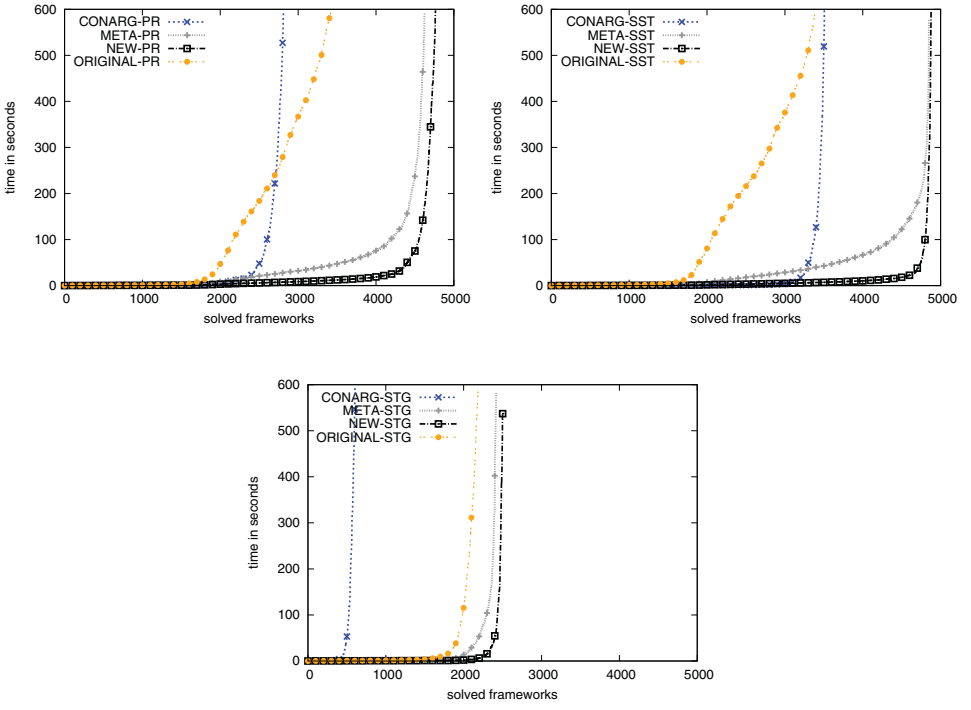


Fig. 1. Runtimes for preferred (PR), semi-stable (SST) and stage (STG) semantics.

encodings. The entries for the median also show that all the novel encodings perform much faster than the other systems, except for semi-stable where ConArg has the lowest median. However, here ConArg is able to solve about 1300 instances less than the novel encodings.

Another interesting observation is that the grounding size of all new encodings is significantly smaller than of both the original and the metasp encodings.

5 Conclusion

In this work, we have developed novel ASP encodings for computationally challenging problems arising in abstract argumentation. Our new encodings for preferred, semi-stable, and stage semantics avoid complicated loop constructs present in previous encodings. In addition to being more succinct, our empirical evaluation showed that a significant performance boost was achieved compared to the earlier ASP encodings, and that our encodings outperform the state-of-the-art system ConArg. From an ASP perspective, our results indicate that loops in saturation encodings (as used in the previous encodings in (Egly *et al.* 2010)) are a severe performance bottleneck which should be avoided.

In future work, we plan to compare our results also with the systems CEGARTIX (Dvořák *et al.* 2014) and ArgSemSAT (Cerutti *et al.* 2014). Furthermore, we also aim for finding better ASP encodings for the ideal (Dung *et al.* 2007) and eager semantics (Caminada 2007).

https://doi.org/10.1017/S1471068415000149 Published online by Cambridge University Press

Acknowledgements

This work has been funded by the Austrian Science Fund (FWF) through projects Y698 and I1102, by the German Research Foundation (DFG) through project HO 1294/11-1, and by Academy of Finland through grants 251170 COIN and 284591.

References

Baroni, P., Caminada, M. W. A. and Giacomin, M. 2011. An Introduction to Argumentation Semantics. *The Knowledge Engineering Review 26*, 4, 365–410.

Besnard, P. and Hunter, A. 2008. *Elements of Argumentation*. MIT Press.

Bistarelli, S. and Santini, F. 2011. ConArg: A constraint-based computational framework for argumentation systems. In *Proceedings of the 23rd IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2011)*, T. M. Khoshgoftaar and X. H. Zhu, Eds. IEEE Computer Society Press, 605–612.

Brewka, G., Eiter, T. and Truszczyński, M. 2011. Answer set programming at a glance. *Communications of the ACM 54*, 12, 92–103.

Caminada, M. W. A. 2007. Comparing Two Unique Extension Semantics for Formal Argumentation: Ideal and Eager. In *Proceedings of the 19th Belgian-Dutch Conference on Artificial Intelligence, BNAIC 2007*. 81–87.

Caminada, M. W. A. and Amgoud, L. 2007. On the Evaluation of Argumentation Formalisms. *Artificial Intelligence 171,* 5-6, 286–310.

Caminada, M. W. A., Carnielli, W. A. and Dunne, P. E. 2012. Semi-Stable Semantics. *Journal of Logic and Computation 22*, 5, 1207–1254.

Cerutti, F., Giacomin, M. and Vallati, M. 2014. ArgSemSAT: solving argumentation problems using SAT. In *Proceedings of the 5th International Conference on Computational Models of Argument (COMMA 2014)*, S. Parsons, N. Oren, C. Reed, and F. Cerutti, Eds. FAIA, vol. 266. IOS Press, 455–456.

Cerutti, F., Oren, N., Strass, H., Thimm, M. and Vallati, M. 2014. A benchmark framework for a computational argumentation competition. In *Proceedings of the 5th International Conference on Computational Models of Argument (COMMA 2014)*, S. Parsons, N. Oren, C. Reed, and F. Cerutti, Eds. FAIA, vol. 266. IOS Press, 459–460.

Charwat, G., Dvorák, W., Gaggl, S. A., Wallner, J. P. and Woltran, S. 2015. Methods for solving reasoning problems in abstract argumentation - A survey. *Artificial Intelligence 220*, 28–63.

Dimopoulos, Y. and Torres, A. 1996. Graph Theoretical Structures in Logic Programs and Default Theories. *Theoretical Computer Science 170,* 1-2, 209–244.

Dung, P. M. 1995. On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-Person Games. *Artificial Intelligence 77,* 2, 321–358.

Dung, P. M., Mancarella, P. and Toni, F. 2007. Computing Ideal Sceptical Argumentation. *Artificial Intelligence 171,* 10-15, 642–674.

Dunne, P. E. and Bench-Capon, T. J. M. 2002. Coherence in finite argument systems. *Artificial Intelligence 141,* 1/2, 187–203.

Dunne, P. E. and Caminada, M. W. A. 2008. Computational Complexity of Semi-Stable Semantics in Abstract Argumentation Frameworks. In *Proceedings of the 11th European Conference on Logics in Artificial Intelligence (JELIA 2008)*, S. Hölldobler, C. Lutz, and H. Wansing, Eds. LNCS, vol. 5293. Springer, 153–165.

https://doi.org/10.1017/S1471068415000149 Published online by Cambridge University Press

Dunne, P. E., Dvořák, W., Linsbichler, T. and Woltran, S. 2014. Characteristics of multiple viewpoints in abstract argumentation. In *Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR 2014)*, C. Baral, G. De Giacomo, and T. Eiter, Eds. AAAI Press, 72–81.

Dunne, P. E. and Wooldridge, M. 2009. Complexity of Abstract Argumentation. In *Argumentation in Artificial Intelligence*, G. Simari and I. Rahwan, Eds. Springer US, 85–104.

Dvořák, W. and Woltran, S. 2010. Complexity of semi-stable and stage semantics in argumentation frameworks. *Information Processing Letters 110*, 11, 425–430.

Dvořák, W., Gaggl, S. A., Wallner, J. P. and Woltran, S. 2013. Making use of advances in answer-set programming for abstract argumentation systems. In *Proceedings of the 19th International Conference on Applications of Declarative Programming and Knowledge Management (INAP 2011), Revised Selected Papers*, H. Tompits, S. Abreu, J. Oetsch, J. Pührer, D. Seipel, M. Umeda, and A. Wolf, Eds. LNAI, vol. 7773. Springer, 114–133.

Dvořák, W., Järvisalo, M., Wallner, J. P. and Woltran, S. 2014. Complexity-sensitive decision procedures for abstract argumentation. *Artificial Intelligence 206*, 53–78.

Egly, U., Gaggl, S. A. and Woltran, S. 2010. Answer-set programming encodings for argumentation frameworks. *Argument & Computation 1*, 2, 147–177.

Gebser, M., Kaminski, R., Kaufmann, B., Lindauer, M., Ostrowski, M., Romero, J., Schaub, T. and Thiele, S. 2015. *Potassco User Guide*, Second edition ed.

Gebser, M., Kaminski, R. and Schaub, T. 2011. Complex optimization in answer set programming. *Theory and Practice of Logic Programming 11*, 4-5, 821–839.

Gelfond, M. and Lifschitz, V. 1991. Classical negation in logic programs and disjunctive databases. *New Generation Computing 9*, 3/4, 365–386.

Lifschitz, V. and Turner, H. 1994. Splitting a logic program. In *Proceedings of the 11th International Conference on Logic Programming (ICLP 1994)*, P. V. Hentenryck, Ed. MIT Press, 23–37.

Rahwan, I. and Simari, G. R., Eds. 2009. *Argumentation in Artificial Intelligence*. Springer.

Syrjänen, T. 2009. Logic programs and cardinality constraints: Theory and practice. Ph.D. thesis, Aalto University.

Toni, F. and Sergot, M. 2011. Argumentation and answer set programming. In *Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning: Essays in Honor of Michael Gelfond*, M. Balduccini and T. C. Son, Eds. LNCS, vol. 6565. Springer, 164–180.

Vallati, M., Cerutti, F. and Giacomin, M. 2014. Argumentation frameworks features: an initial study. In *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI 2014)*, T. Schaub, G. Friedrich, and B. O'Sullivan, Eds. FAIA, vol. 263. IOS Press, 1117–1118.

Verheij, B. 1996. Two Approaches to Dialectical Argumentation: Admissible Sets and Argumentation Stages. In *Proceedings of the Eighth Dutch Conference on Artificial Intelligence (NAIC'96)*, J.-J. C. Meyer and L. van der Gaag, Eds. 357–368.

https://doi.org/10.1017/S1471068415000149 Published online by Cambridge University Press