The ingredients of the argumentation reasoner pyglaf: python, circumscription, and glucose to taste

Mario Alviano

Department of Mathematics and Computer Science, University of Calabria, Italy
alviano@mat.unical.it

Abstract. The fundamental mechanism that humans use in argumentation can be formalized in abstract argumentation frameworks. Many semantics are associated with abstract argumentation frameworks, each one consisting of a set of extensions, that is, a set of sets of arguments. Some of these semantics are based on preference relations that essentially impose to maximize or minimize some property. This paper presents the argumentation reasoner PYGLAF, which provides a uniform view of many semantics for abstract argumentation frameworks in terms of circumscription. Specifically, several computational problems of abstract argumentation frameworks are reduced to circumscription by means of linear encodings, and a few others are solved by means of a sequence of calls to an oracle for circumscription. Python is used to build the encodings and to control the execution of the external circumscription solver, which is based on the SAT solver GLUCOSE.

Keywords: abstract argumentation frameworks, propositional circumscription, minimal model enumeration, incremental solving

1 Introduction

Abstract argumentation frameworks [21] are a declarative paradigm to represent and reason about the fundamental mechanism humans use in argumentation. Abstract argumentation frameworks are represented by directed graphs: nodes represent abstract arguments, and arcs encode an attack relation between abstract arguments. The term abstract refers to the fact that the content of the arguments is not further analyzed. What is analyzed instead is the attack relation. A simple and well-known example is the Nixon diamond, whose arguments "Nixon is anti-pacifist since he is a republican" and "Nixon is a pacifist since he is a quaker" attack each other. Several semantics have been defined in the literature to capture different aspects of the processed abstract argumentation frameworks, where a semantics associates each AF with a set of extensions, and an extension is a set of arguments satisfying an accepting condition. Prominent examples are *complete extensions* and *stable extensions* [15].

Many semantics of abstract argumentation frameworks are based on a preference relation over the extensions characterizing other semantics. Such preference relations essentially amount to inclusion relationships either on the set of accepted arguments, or on some other property of the extensions. Specifically, *grounded extensions* [22] are subset-minimal with respect to the accepted arguments, and conversely *preferred extensions* [15] and *ideal extensions* [22] are subset-maximal with respect to the accepted arguments. Other semantics of this kind are *semi-stable extensions* [15] and *stage extensions* [27], which require to maximize the set of arguments in the range of the accepted arguments, that is, those accepted and those attacked by some accepted argument.

Circumscription [26] is a nonmonotonic logic formalizing common sense reasoning by means of a second order semantics, which essentially enforces to minimize the extension of some predicates. In the special case of propositional theories, the simplest form of circumscription essentially selects subset minimal models, while in the form introduced by Lifschitz [25] some atoms are used to group interpretations, and other atoms are subject to minimization. With a little abuse on the definition of circumscription, the minimization can be imposed on a set of literals, so that a set of negative literals can be used to encode a maximization objective function. Therefore, circumscription is a good candidate as target language to solve computational problems of abstract argumentation frameworks.

This paper presents the argumentation reasoner PYGLAF, which materializes the above intuition. The reasoner is implemented in Python, and its interface is fully compliant with the specification[1] given for the Second International Competition on Computational Models of Argumentation (ICCMA'17). PYGLAF implements linear reductions from many computational problems of abstract argumentation framework to circumscription. The circumscribed theories are encoded in the input format of CIRCUMSCRIPTINO [1], a circumscription solver extending the SAT solver GLUCOSE [8] with the unsatisfiable core algorithm ONE [7] enhanced by *reiterated progression shrinking* [3, 4], native support for cardinality constraints as in WASP [5, 6, 20], and polyspace model enumeration [2]. The input format of CIRCUMSCRIPTINO is very similar to the DIMACS format, and described on its web page (`http://alviano.com/software/circumscriptino/`). The reductions are presented in Section 3, and cover all semantics mentioned in this introduction but the ideal extension. Eventually, a linear reduction for ideal semantics is obtained after computing the union of all admissible extensions of the input graph; such a set is computed by means of iterative calls to the external circumscription solver.

An additional use case is given by the Dung's Triathlon, a special track of ICCMA'17 where the argumentation reasoners are asked to compute the grounded, stable and preferred extensions of the input graph, in a single run of computation. PYGLAF addresses the triathlon by taking advantage of the following observations [21]: the grounded extension is contained in all preferred extensions, and all stable extensions are also preferred extensions. Hence, PYGLAF first computes the grounded extension, which is then used to simplify the circumscribed theory associated with the enumeration of preferred extensions. As soon as a preferred

[1] `http://www.dbai.tuwien.ac.at/iccma17/SolverRequirements.pdf`

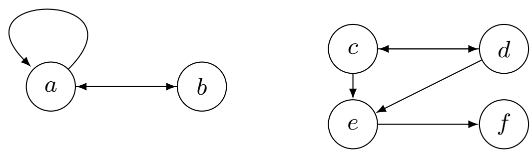

Fig. 1. The graph used as running example.

extension is returned by CIRCUMSCRIPTINO, its stability is checked by means of a linear time procedure implemented in Python.

The implemented prototype is tested empirically on the instances of the First International Competition on Computational Models of Argumentation (ICCMA'15). For the enumeration of complete, stable, grounded, and preferred extensions, the performance of PYGLAF is almost aligned with the most efficient argumentation reasoners of ICCMA'15. Semi-stable, stage, and ideal extensions were not part of ICCMA'15, but they are supported by the reasoner CONARG2 [12, 10, 11], which is therefore compared with PYGLAF: for the enumeration of these extensions the performance of PYGLAF is superior in terms of solved instances as well as average running time. Finally, concerning the triathlon, the performance of the proposed strategy is compared with the sequential addressing of the three subproblems, showing a minimal gain in terms of solved instances and average execution time.

2 Background

This section introduces the required background on abstract argumentation frameworks [21, 22, 15, 27, 19] and on circumscription [26, 25].

2.1 Abstract argumentation framework

An *abstract argumentation framework* (AF) is a directed graph G whose nodes are arguments, and whose arcs represent an attack relation. Let $arg(G)$ and $att(G)$ denote respectively the set of nodes and arcs in G. The *size* of G, denoted $|G|$, is defined as the number of its nodes and arcs, that is, $|G| := |arg(G)| + |att(G)|$. An *extension* E is a set of arguments. The *range* of E in G is $E^+_G := E \cup \{x \mid \exists yx \in att(G) \text{ with } y \in E\}$. (When G is clear from the context, the range of E in G is simply denoted E^+.)

Example 1. Let G be the graph reported in Figure 1. Hence, $arg(G) = \{a, b, c, d, e, f\}$, $att(G) = \{aa, ab, ba, cd, dc, ce, de, ef\}$, and $|G| = 6+8 = 14$. The following are some extensions of G that will be used later in the paper: $E_1 = \emptyset$, $E_2 = \{d, f\}$, $E_3 = \{b, d, f\}$, $E_4 = \{b\}$, $E_5 = \{b, c, f\}$, and $E_6 = \{c, f\}$. The associated ranges are the following: $E_1^+ = \emptyset$, $E_2^+ = \{c, d, e, f\}$, $E_3^+ = \{a, b, c, d, e, f\}$, $E_4^+ = \{a, b\}$, $E_5^+ = \{a, b, c, d, e, f\}$, and $E_6^+ = \{c, d, e, f\}$. ∎

Let G be an AF, and $E \subseteq arg(G)$. The following semantics are of interest for this paper:

– E is *conflict-free* if there are no $x, y \in E$ with $xy \in att(G)$. Let $\mathbf{CF}(G)$ denote the set of conflict-free extensions of G.
– E is *admissible* if $E \in \mathbf{CF}(G)$, and for all $yx \in att(G)$ such that $x \in E$, there is $zy \in att(G)$ such that $z \in E$. Let $\mathbf{ADM}(G)$ denote the set of admissible extensions of G.
– E is *complete* if $E \in \mathbf{ADM}(G)$, and satisfies the following property: if $x \in arg(G)$ is such that for all $yx \in att(G)$ there is $zy \in att(G)$ with $z \in E$, then $x \in E$. Let $\mathbf{CO}(G)$ denote the set of complete extensions of G.
– E is *stable* if $E \in \mathbf{CO}(G)$, and $E_G^+ = arg(G)$. Let $\mathbf{ST}(G)$ denote the set of stable extensions of G.
– E is *grounded* if $E \in \mathbf{CO}(G)$, and there is no $E' \in \mathbf{CO}(G)$ such that $E' \subset E$. Let $\mathbf{GR}(G)$ denote the set of grounded extensions of G.
– E is *preferred* if $E \in \mathbf{CO}(G)$, and there is no $E' \in \mathbf{CO}(G)$ such that $E' \supset E$. Let $\mathbf{PR}(G)$ denote the set of preferred extensions of G.
– E is *semi-stable* if $E \in \mathbf{CO}(G)$, and there is no $E' \in \mathbf{CO}(G)$ such that ${E'}_G^+ \supset E_G^+$. Let $\mathbf{SST}(G)$ denote the set of semi-stable extensions of G.
– E is *stage* if $E \in \mathbf{CF}(G)$, and there is no $E' \in \mathbf{CF}(G)$ such that ${E'}_G^+ \supset E_G^+$. Let $\mathbf{STG}(G)$ denote the set of stage extensions of G.
– E is *ideal* if $E \in \mathbf{ADM}(G)$, $E \subseteq \bigcap \mathbf{PR}(G)$, and there is no $E' \in \mathbf{ADM}(G)$ such that $E' \subseteq \bigcap \mathbf{PR}(G)$ and $E' \supset E$. Let $\mathbf{ID}(G)$ denote the set of ideal extensions of G.

In particular, the last seven semantics above are those considered in the Second International Competition on Computational Models of Argumentation (ICCMA'17).

Example 2 (Continuing Example 1). The set of complete extensions of G is $\mathbf{CO}(G) = \{E_1, \ldots, E_6\}$. E_3 and E_5 are also the stable extensions of G because $E_3^+ = E_5^+ = arg(G)$. These are also the preferred, semi-stable, and stage extensions of G. E_1 is evidently the grounded extension. Finally, the ideal extension is $\{b\}$. ∎

For an argument $x \in arg(G)$, and a set S of extensions, x is *credulously accepted* in S, denoted $S \models_c x$, if there is $E \in S$ such that $x \in E$. Argument x is *skeptically accepted* in S, denoted $S \models_s x$, if $x \in E$ for all $E \in S$.

Example 3 (Continuing Example 2). $S \models_s b$ and $S \models_s f$ holds for S being $\mathbf{ST}(G)$, $\mathbf{PR}(G)$, $\mathbf{SST}(G)$, $\mathbf{STG}(G)$. For the same S, $S \models_c d$ and $S \models_c c$, but $S \not\models_s d$ and $S \not\models_s c$. ∎

2.2 Circumscription

Let $\mathcal{A}$ be a fixed, countable set of *atoms* including $\bot$. A *literal* is an atom possibly preceded by the connective $\neg$. For a literal ℓ, let $\overline{\ell}$ denote its *complementary literal*, that is, $\overline{p} = \neg p$ and $\overline{\neg p} = p$ for all $p \in \mathcal{A}$; for a set L of literals, let $\overline{L}$ be $\{\overline{\ell} \mid \ell \in L\}$. Moreover, for a set L of literals and a set A of atoms, the *restriction* of L to symbols in A is $L|_A := L \cap (A \cup \overline{A})$.

Formulas are defined as usual by combining atoms and the connectives $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$. A *theory* is a set T of formulas including $\neg\bot$; the set of atoms occurring in T is denoted by $atoms(T)$. The *size* of formulas and theories is defined as the number of occurring literals; formally: for $p \in \mathcal{A}$, $|p| := 1$; for ϕ and ψ formulas, and $\circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$, $|\neg\phi| := |\phi|$, and $|\phi \circ \psi| := |\phi| + |\psi|$; for a theory T, $|T| := \sum_{\phi \in T} |\phi|$.

An *assignment* is a set A of literals such that $A \cap \overline{A} = \emptyset$. An *interpretation* for a theory T is an assignment I such that $(I \cup \overline{I}) \cap \mathcal{A} = atoms(T)$. Relation $\models$ is defined as usual: for $p \in \mathcal{A}$, $I \models p$ if $p \in I$; for ϕ and ψ formulas, $I \models \neg\phi$ if $I \not\models \phi$, $I \models \phi \wedge \psi$ if $I \models \phi$ and $I \models \psi$, $I \models \phi \vee \psi$ if $I \models \phi$ or $I \models \psi$, and $I \models \phi \rightarrow \psi$ if $I \models \psi$ whenever $I \models \phi$; $I \models \phi \leftrightarrow \psi$ if $I \models \psi$ whenever $I \models \phi$, and vice versa; for a theory T, $I \models T$ if $I \models \phi$ for all $\phi \in T$. I is a *model* of a theory T if $I \models T$. Let $models(T)$ denote the set of models of T. (Models will be also represented by the set of their atoms, as their negative literals are implicit.)

Example 4. Let T_1 be the theory $\{\neg\bot,\ a \vee b,\ c \vee d \rightarrow \neg a \wedge b\}$. The size of T_1 is $|T_1| = 1 + 2 + 4 = 7$. The models of T_1 are the following: $\{a\}$, $\{b\}$, $\{a, b\}$, $\{b, c\}$, $\{b, d\}$, and $\{b, c, d\}$. ■

Circumscription applies to a theory T, a set P of literals, and a set Z of atoms; literals in P are subject to minimization, while atoms in Z are irrelevant. Formally, relation $\leq^{PZ}$ is defined as follows: for I, J interpretations of T, $I \leq^{PZ} J$ if both $I|_{\mathcal{A} \setminus (P \cup \overline{P} \cup Z)} = J|_{\mathcal{A} \setminus (P \cup \overline{P} \cup Z)}$ and $I \cap P \subseteq J \cap P$. $I \in models(T)$ is a *preferred model* of T with respect to $\leq^{PZ}$ if there is no $J \in models(T)$ such that $I \not\leq^{PZ} J$ and $J \leq^{PZ} I$. Let $CIRC(T, P, Z)$ denote the set of preferred models of T with respect to $\leq^{PZ}$.

Example 5 (Continuing Example 4). $CIRC(T_1, \{a, b, c, d\}, \emptyset)$ contains the minimal models of T_1: $\{a\}$, and $\{b\}$. Minimization can be restricted to few atoms: $CIRC(T_1, \{b\}, \{a, c, d\})$ contains only $\{a\}$. Atoms not in P nor in Z are used to group interpretations: $CIRC(T_1, \{b\}, \{c, d\})$ contains $\{a\}$, but also $\{b\}$, $\{b, c\}$, $\{b, d\}$, and $\{b, c, d\}$. Finally, $CIRC(T_1, \{\neg a, \neg b\}, \{d\})$ contains the following models: $\{a, b\}$, $\{b, c\}$, and $\{b, c, d\}$. ■

3 Linear reductions

The constructions presented in this section associate arguments of the input AF G with atoms of a theory T, so to have a trivial one-to-one mapping between extensions of G and models of T. In particular, a set S of extensions of G and a set S' of models of T are said equivalent with respect to G, denoted $S \equiv_G S'$, if $S = \{I \cap \mathit{arg}(G) \mid I \in S'\}$.

In order to maintain the theory compact, additional atoms are possibly introduced. Specifically, for each argument x, an atom a_x is possibly introduced to represent that x is attacked by some argument that belongs to the computed extension:

$$\mathit{attacked}(G) := \left\{ a_x \leftrightarrow \bigvee_{yx \in \mathit{att}(G)} y \;\middle|\; x \in \mathit{arg}(G) \right\} \tag{1}$$

Note that if there is no $yx \in \mathit{att}(G)$, then $\bigvee_{yx \in \mathit{att}(G)} y$ is essentially $\bot$, that is, atom a_x is constrained to be false. Other atoms possibly introduced by our constructions are of the form r_x, to enforce that argument x belongs to the range E_G^+ of the computed extension E:

$$\mathit{range}(G) := \left\{ r_x \rightarrow x \vee \bigvee_{yx \in \mathit{att}(G)} y \;\middle|\; x \in \mathit{arg}(G) \right\} \tag{2}$$

Example 6 (Continuing Example 2). For the graph G in Figure 1, the following formulas are possibly introduced:

$$S_1 := \{a_a \leftrightarrow a \vee b, a_b \leftrightarrow a, a_c \leftrightarrow d, a_d \leftrightarrow c, a_e \leftrightarrow c \vee d, a_f \leftrightarrow e\};$$
$$S_2 := \{r_a \rightarrow a \vee b, r_b \rightarrow b \vee a, r_c \rightarrow c \vee d, r_d \rightarrow d \vee c, r_e \rightarrow e \vee c \vee d, r_f \rightarrow f \vee e\}.$$

In fact, note that S_1 is *attacked*(G), and S_2 is *range*(G). ■

The notion of conflict-free, admissible, complete, and stable extensions are encoded by the following sets of formulas:

$$\mathit{conflict\text{-}free}(G) := \{\neg\bot\} \cup \{\neg x \vee \neg y \mid xy \in \mathit{att}(G)\} \tag{3}$$
$$\mathit{admissible}(G) := \mathit{conflict\text{-}free}(G) \cup \mathit{attacked}(G) \cup \{x \rightarrow a_y \mid yx \in \mathit{att}(G)\} \tag{4}$$
$$\mathit{complete}(G) := \mathit{admissible}(G) \cup \left\{ \left(\bigwedge_{yx \in \mathit{att}(G)} a_y \right) \rightarrow x \;\middle|\; x \in \mathit{arg}(G) \right\} \tag{5}$$
$$\mathit{stable}(G) := \mathit{complete}(G) \cup \mathit{range}(G) \cup \{r_x \mid x \in \mathit{arg}(G)\} \tag{6}$$

In particular, note that in (4) truth of an argument x implies that all arguments attacking x are actually attacked by some true argument. In (5), instead, whenever all attackers of an argument x are attacked by some true argument, argument x is forced to be true. Finally, in (6) all atoms of the form r_x are forced to be true, so that the range of the computed extension has to cover all arguments.

Example 7 (Continuing Example 6). Consider the following sets of formulas:

$$S_3 := \{\neg\bot, \neg a, \neg a \vee \neg b, \neg c \vee \neg d, \neg c \vee \neg e, \neg d \vee \neg e, \neg e \vee \neg f\};$$
$$S_4 := \{a \rightarrow a_a, a \rightarrow a_b, b \rightarrow a_a, c \rightarrow a_d, e \rightarrow a_c, d \rightarrow a_c, e \rightarrow a_d, f \rightarrow a_e\};$$
$$S_5 := \{a_a \wedge a_b \rightarrow a, a_a \rightarrow b, a_d \rightarrow c, a_c \rightarrow d, a_c \wedge a_d \rightarrow e, a_e \rightarrow f\}.$$

Note that S_3 is *conflict-free*(G), $S_3 \cup S_1 \cup S_4$ is *admissible*(G), $S_3 \cup S_1 \cup S_4 \cup S_5$ is *complete*(G), and $S_3 \cup S_1 \cup S_4 \cup S_5 \cup S_2 \cup \{r_a, r_b, r_c, r_d, r_e, r_f\}$ is *stable*(G). ■

The following circumscribed theories capture the complete, stable, grounded, preferred, semi-stable, and stage semantics of G:

$$co(G) := CIRC(complete(G), \emptyset, Z) \tag{7}$$
$$st(G) := CIRC(stable(G), \emptyset, Z) \tag{8}$$
$$gr(G) := CIRC(complete(G), arg(G), \{a_x \mid x \in arg(G)\}) \tag{9}$$
$$pr(G) := CIRC(complete(G), \overline{arg(G)}, \{a_x \mid x \in arg(G)\}) \tag{10}$$
$$sst(G) := CIRC(complete(G) \cup range(G), \{\neg r_x \mid x \in arg(G)\}, Z) \tag{11}$$
$$stg(G) := CIRC(\mathit{conflict\text{-}free}(G) \cup range(G), \{\neg r_x \mid x \in arg(G)\}, Z) \tag{12}$$

where Z is $arg(G) \cup \{a_x \mid x \in arg(G)\}$. Note that the notion of complete and stable extensions does not really involve any preference relation, and therefore the set of literals to be minimized is empty in (7) and (8). Grounded and preferred extensions are instead obtained by imposing complementary objective literals: true arguments are minimized in (9) to capture grounded extensions, and false arguments are minimized in (10) to capture preferred extensions. Finally, in (11) and (12) false atoms in the range of the computed extensions are minimized, where computed extensions are respectively complete extensions and conflict-free extensions.

Example 8 (Continuing Example 7). Let Z be $\{a, b, c, d, e, f, a_a, a_b, a_c, a_d, a_e, a_f\}$. It can be checked that $CIRC(S_3 \cup S_1 \cup S_4 \cup S_5, \emptyset, Z)$ contains, for $i \in [1..6]$, the model $E_i \cup \{a_x \mid x \in E_i^+\}$, and only these models. Similar for the other circumscribed theories resulting by applying (8)–(12) to graph G in Figure 1. ■

The ideal semantics cannot be captured by reductions similar to those above. However, every AF G is associated with a unique ideal extension, which can be equivalently defined as follows (Proposition 3.6 by Caminada [14]): Let X be the set of admissible extensions of G that are not attacked by any admissible extensions, that is, $X := \{E \in \mathbf{ADM}(G) \mid \nexists E' \in \mathbf{ADM}(G)$ such that $yx \in att(G), x \in E, y \in E'\}$. E is the ideal extension of G if $E \in X$, and there is no $E' \in X$ such that $E' \supsetneq E$. Hence, assuming that the union U of all admissible extensions of G is provided in input, the following linear construction can be defined:

$$id(G, U) := CIRC(admissible(G) \cup \overline{arg(G) \setminus Y},\ \overline{Y},\ \emptyset) \tag{13}$$

where Y is $U \setminus \{x \mid \exists yx \in att(G),\ y \in U\}$.

Example 9 (Continuing Example 7). The union of all admissible extensions of graph G in Figure 1 is $U = \{b, c, d, f\}$. Hence, $Y = \{b, f\}$, and $id(G, U)$ is $CIRC(S_3 \cup S_1 \cup S_4 \cup \{\neg a, \neg c, \neg d, \neg e\}, \{\neg b, \neg f\}, \emptyset) = \{\{b\}\}$. ■

3.1 Properties

Correctness of the reductions presented in this section is a direct consequence of the fact that equations (3)–(6) encode the definitions of conflict-free, admissible, complete, and stable extensions in propositional logic.

Theorem 1 (Correctness). *Let G be an AF. The following equivalences hold:* $\mathbf{CO}(G) \equiv_G co(G)$, $\mathbf{ST}(G) \equiv_G st(G)$, $\mathbf{GR}(G) \equiv_G gr(G)$, $\mathbf{PR}(G) \equiv_G pr(G)$, $\mathbf{SST}(G) \equiv_G sst(G)$, $\mathbf{STG}(G) \equiv_G stg(G)$, *and* $\mathbf{ID}(G) \equiv_G id(G, \bigcup_{E \in \mathbf{ADM}(G)} E)$.

Compactness of the reductions follows from the fact that equations (3)–(6) have linear size with respect to the size of G.

Theorem 2 (Compactness). *Let G be an AF. The size of the theories in (7)–(13) is linear in $|G|$.*

Credulous acceptance can be also reduced to circumscription for complete, stable, and preferred extensions.

Theorem 3 (Credulous). *Let G be an AF, and x be an argument. The following properties hold:*

- $\mathbf{CO}(G) \models_c x$ *iff* $CIRC(complete(G) \cup \{x\}, \emptyset, arg(G) \cup \{a_y \mid y \in arg(G)\}) \neq \emptyset$;
- $\mathbf{ST}(G) \models_c x$ *iff* $CIRC(stable(G) \cup \{x\}, \emptyset, arg(G) \cup \{a_y \mid y \in arg(G)\}) \neq \emptyset$;
- $\mathbf{PR}(G) \models_c x$ *iff* $CIRC(complete(G) \cup \{x\}, \overline{arg(G)}, \{a_x \mid x \in arg(G)\}) \neq \emptyset$.

Similarly, skeptical acceptance can be reduced to circumscription for complete, and stable extensions.

Theorem 4 (Skeptical). *Let G be an AF, and x be an argument. The following properties hold:*

- $\mathbf{CO}(G) \models_s x$ *iff* $CIRC(complete(G) \cup \{\neg x\}, \emptyset, arg(G) \cup \{a_y \mid y \in arg(G)\}) = \emptyset$;
- $\mathbf{ST}(G) \models_s x$ *iff* $CIRC(stable(G) \cup \{\neg x\}, \emptyset, arg(G) \cup \{a_y \mid y \in arg(G)\}) = \emptyset$.

4 Implementation

The reductions presented in the previous section have been implemented in the reasoner PYGLAF (`http://alviano.com/software/pyglaf/`). The underlying programming language is Python, and the external circumscription solver is CIRCUMSCRIPTINO. The communication between PYGLAF and CIRCUMSCRIPTINO is handled in the simplest possible way, that is, via stream processing. This design choice is principally motivated by the fact that the communication is often minimal, limited to a single invocation of the circumscription solver.

The interface of PYGLAF is fully compliant with the specification given for the Second International Competition on Computational Models of Argumentation (ICCMA'17). Abstract argumentation frameworks can be encoded in trivial graph format (TGF) as well as in aspartix format (APX). The following data

Algorithm 1: Compute the union of admissible extensions of an AF G

$T := admissible(G)$; `// fix the underlying theory`
$Z := arg(G) \cup \{a_x \mid x \in arg(G)\}$; `// fix the set of irrelevant atoms`
$U := \emptyset$; `// initialize the union of all admissible extensions`
repeat
  Compute $I \in CIRC(T, \overline{arg(G) \setminus U}, Z)$; `// prefer arguments not in U`
  $I := I \cap arg(G)$; `// restrict to arguments of G`
  $U := U \cup I$; `// possibly extend the union`
until $I \subseteq U$; `// terminate when no argument is added to U`

structures are populated during the parsing of the input graph G: a list `arg` of the arguments in $arg(G)$; a dictionary `argToIdx`, mapping each argument x to its position in `arg`; a dictionary `att`, mapping each argument x to the set $\{y \mid xy \in att(G)\}$; a dictionary `attR`, mapping each argument x to the set $\{y \mid yx \in att(G)\}$. The first element of the list `arg`, in position 0, is not used to simplify the alignment between arguments of G and variables of the produced circumscribed theory. Within these data structures, the theories in (7)–(13) can be constructed in amortized linear time. Actually, their CNF representation is easily built without introducing any additional atom, as all formulas in the previous section are either implications of the form $\bigwedge A \rightarrow \bigvee B$, or equivalences of the form $p \leftrightarrow A$, where p is an atom and A, B are sets of literals.

Ideal extension. The linear reduction for the computation of the ideal extension requires the union of all admissible extensions as an input parameter. Such a set U is computed by means of Algorithm 1. Initially, U is empty, and CIRCUMSCRIPTINO is asked to compute a first admissible extension that maximize the accepted arguments. The accepted arguments are then added to U, another admissible extension is computed by CIRCUMSCRIPTINO. However, this time the maximization only involves the arguments that do not belong to U, so to expand U as much as possible. This process is repeated until the admissible extension returned by CIRCUMSCRIPTINO is covered by set U, as such extension witnesses that all admissible extensions are a subset of U.

Credulous and skeptical acceptance. Credulous acceptance is addressed according to Theorem 3 for complete, stable, and preferred extensions. Similarly, skeptical acceptance is addressed according to Theorem 4 for complete, and stable extensions. Grounded and ideal extensions are unique, and therefore credulous (and skeptical) acceptance are addressed by checking the presence of the query argument in the computed extension. Actually, for the ideal extension, a negative answer is possibly produced already if the query argument is not part of the union of all admissible extensions. The remaining acceptance problems are addressed naively by extension enumeration.

Dung's Triathlon. The triathlon is addressed by Algorithm 2 based on the following observations: the grounded extension is contained in every preferred extension (Theorem 25 by Dung [21]), and every stable extension is a preferred

Algorithm 2: Grounded, stable and preferred extensions of G

Compute $I_{gr} \in co(G)$; `// first of all, compute the grounded extension`
$stable := \emptyset$; `// initialize the set of stable extensions`
$preferred := \emptyset$; `// initialize the set of preferred extensions`
`// simplification:` I_{gr} `is contained in all preferred extensions`
$T := complete(G) \cup \{x \in arg(G) \mid x \in I_{gr}\}$; $P := \overline{arg(G)}$; $Z := \{a_x \mid x \in arg(G)\}$;
for $I \in CIRC(T, P, Z)$ **do** `// enumerate preferred extensions`
  $preferred := preferred \cup \{I\}$; `// found new preferred extension`
  if *for all* $x \in arg(G) \setminus I$ *there is* $yx \in att(G)$ *such that* $y \in I$ **then**
    $stable := stable \cup \{I\}$; `// the preferred extension is also stable`
return $(I_{gr}, stable, preferred)$;

extension (Lemma 15 by Dung [21]). Accordingly, the algorithm starts by computing the unique grounded extension I_{gr} of the input graph. After that, a theory whose models are complete extensions is built, and simplified by enforcing truth of all arguments in I_{gr}. The objective literals are the negation of all arguments, so that preferred extensions will be computed by CIRCUMSCRIPTINO. Every preferred extension returned by CIRCUMSCRIPTINO is finally checked for stability by means of a linear time Python function.

5 Experiments

In order to assess empirically the implemented reasoner, instances from the First International Competition on Computational Models of Argumentation (ICCMA'15) were tested on the enumeration of all extensions for several semantics. The performance of PYGLAF is compared with the following reasoners that participated in the competition: ArgSemSAT (version 1.0rc3; [18]) and LabSATSolver (version 0.1; [9]), based on SAT encodings of Caminada's labeling approach [13, 16]; ASPARTIX-D (version ICCMA'15; [24]), based on reductions to answer set programming; ConArg2 (version 1.0; [11]), based on reductions to constraint satisfaction; PREFMaxSAT (version 0.1rc3; [17]), based on iterative calls to a MaxSAT solver; PREFASP (version ICCMA'15; [23]), similar to PREFMaxSAT, but based on answer set programming. The experiments were run on an Intel Xeon 2.4 GHz with 16 GB of memory, and time and memory were limited to 10 minutes and 15 GB, respectively.

Complete and stable extensions. For these two semantics the set of objective literals is empty. Aggregated results are reported on Table 1. The performance of PYGLAF is essentially aligned to the performance of ArgSemSAT and LabSATSolver: they run out of time on the same instance of 4 ST SMALL. All instances are solved by ASPARTIX-D, while ConArg2 collected several timeouts. Similar results are obtained for the enumeration of stable extensions, which was completed for all instances by all tested reasoners but ConArg2.

Grounded and preferred extensions. For the computation of grounded and preferred extensions the underlying propositional theory has models representing complete extensions of the input graph. Complementary objective literals are used: accepted arguments are minimized to obtain the grounded extension, while nonaccepted arguments are minimized to obtain preferred extensions. Aggregated results are reported on Table 2. The grounded extension is unique, and computed very efficiently by PYGLAF, ARGSEMSAT, and LABSATSOLVER. Several timeouts are instead collected by ASPARTIX-D and CONARG2. As for preferred extensions, a very good performance is exhibited by PYGLAF, ARGSEMSAT, and LABSATSOLVER, running out of time only on one instance (the same instance for which they cannot enumerate all complete extensions). Preferred extensions can be also enumerated by the reasoners PREFMAXSAT and PREFASP. The first reasoner collected several timeouts, while PREFASP resulted very efficient and solved all testcases.

Other extensions and Dung's Triathlon. The enumeration of semi-stable, stage and ideal extensions is supported by CONARG2, but not by the other reasoners (at least in the versions found on the web). Hence, for these two semantics the comparison is restricted to PYGLAF and CONARG2. Aggregated results are reported on Table 3. The performance of PYGLAF is in general better than that of CONARG2, especially on the enumeration of semi-stable and ideal extensions. Several timeouts are collected by PYGLAF on the enumeration of stage extensions due to their large number. It is interesting to observe that for the stage semantics PYGLAF requires around one order of magnitude less memory than CONARG2. Concerning the triathlon, the improvement on the naive approach of executing

Table 1. Enumeration of complete and stable extensions: solved instances, average execution time (seconds) on solved instances, and average memory consumption (MB).

COMPLETE		PYGLAF			CONARG2			ARGSEMSAT			LABSATSOL.			ASPARTIX-D		
Benchmark	#	sol	time	mem	sol	time	mem	sol	time	mem	sol	time	mem	sol	time	mem
1 gr small	24	24	1.3	50	24	1.4	54	24	0.8	75	24	1.7	140	24	0.6	43
2 gr med.	24	24	3.6	84	24	3.8	126	24	2.5	147	24	2.2	206	24	1.4	76
3 gr large	24	24	18.4	237	24	17.1	489	24	14.5	457	24	4.0	503	24	5.2	221
4 st small	24	23	49.2	36	11	167.1	14	23	101.9	27	23	68.8	58	24	30.1	11
5 st med.	24	24	65.9	42	0	—	18	24	115.2	31	24	92.0	64	24	28.7	15
7 scc small	24	24	0.1	14	24	0.5	4	23	15.4	21	24	1.0	58	24	0.0	1
8 scc med.	24	24	0.5	25	21	23.7	76	21	1.0	38	24	1.6	95	24	0.1	15
9 scc large	24	24	2.7	32	22	9.7	290	23	7.9	57	24	2.2	113	24	0.4	26
Total	192	191	17.6	65	150	20.6	134	186	32.8	107	191	21.4	155	192	8.3	51
STABLE		PYGLAF			CONARG2			ARGSEMSAT			LABSATSOL.			ASPARTIX-D		
Benchmark	#	sol	time	mem	sol	time	mem	sol	time	mem	sol	time	mem	sol	time	mem
1 gr small	24	24	1.4	51	24	0.8	35	24	0.8	76	24	1.7	140	24	0.6	42
2 gr med.	24	24	3.7	84	24	1.8	69	24	2.5	148	24	2.2	205	24	1.3	74
3 gr large	24	24	18.7	237	24	6.3	220	24	14.5	458	24	4.0	504	24	4.8	216
4 st small	24	24	35.0	31	23	163.1	7	23	42.3	21	24	41.7	57	24	14.3	11
5 st med.	24	24	22.6	34	17	301.3	9	24	41.5	25	24	31.7	62	24	11.9	13
7 scc small	24	24	0.1	14	24	0.1	1	24	0.0	1	24	0.9	56	24	0.0	0
8 scc med.	24	24	0.6	25	20	13.4	17	24	0.3	21	24	1.3	90	24	0.1	12
9 scc large	24	24	2.7	30	21	34.2	29	24	4.7	43	24	1.7	109	24	0.3	20
Total	192	192	10.6	63	177	56.9	49	191	13.2	99	192	10.6	153	192	4.2	48

the three computational tasks sequentially is minimal: the gain is limited to one solved instance, and few seconds of computation on average.

6 Related work

There are many argumentation reasoners; the closest to the present work are ArgSemSAT [18] and LabSATSolver [9], which are based on iterative calls to an external SAT solver. These two solvers encode in propositional logic the Caminada's labeling approach [13, 16], which maps each argument to a label among `in`, `out`, and `undec`. A similar strategy is implemented by prefMaxSAT [17] and prefASP [23], which encode Caminada's labeling respectively in MaxSAT and answer set programming. pyglaf instead encodes the several semantics of abstract argumentation framework in the language of circumscription. Actually, equations (3)–(6) can be seen as linear representations of the propositional theories introduced by Wallner et al. [28].

The definition of circumscription given in this paper is slightly different from the one originally introduced by Mc Carthy [26] and by Lifschitz [25], since minimization is enforced on a set of literals instead of a set of atoms. Actually, it is not difficult to transform the reductions presented in this paper to the original formalism of circumscription: replace each negative literal $\neg p$ in P with a fresh atom f_p, and add to the theory the formula $f_p \leftrightarrow \neg p$. However, it is remarked here that circumscriptino can handle negative objective literals without introducing any additional atom.

Table 2. Enumeration of grounded and preferred extensions: solved instances, average execution time (seconds) on solved instances, and average memory consumption (MB).

grounded		pyglaf			ConArg2			ArgSemSAT			LabSATSol.			aspartix-d		
Benchmark	#	sol	time	mem	sol	time	mem	sol	time	mem	sol	time	mem	sol	time	mem
1 gr small	24	24	1.3	51	24	1.4	54	24	0.8	75	24	1.4	128	24	118.7	802
2 gr med.	24	24	3.6	85	24	3.8	126	24	2.5	147	24	1.7	187	15	159.3	1660
3 gr large	24	24	18.4	237	24	17.1	490	24	14.5	456	24	3.0	431	0	—	4506
4 st small	24	24	0.1	17	11	166.6	16	24	5.8	14	24	0.6	50	24	0.9	49
5 st med.	24	24	0.1	20	0	—	21	24	8.9	19	24	0.8	55	24	1.5	71
7 scc small	24	24	0.1	15	24	0.5	3	24	0.0	2	24	0.7	51	8	0.7	59
8 scc med.	24	24	0.5	25	22	13.1	75	24	0.3	25	24	1.0	79	2	0.8	213
9 scc large	24	24	2.3	30	22	9.5	294	24	1.5	41	24	1.0	92	7	0.9	397
Total	192	192	3.3	60	151	19.1	135	192	4.3	97	192	1.3	134	104	51.1	970

preferred		pyglaf			ConArg2			ArgSemSAT			LabSATSol.			aspartix-d			prefASP			prefMaxSAT		
Bench.	#	sol	time	mem	sol	time	mem	sol	time	mem	sol	time	mem	sol	time	mem	sol	time	mem	sol	time	mem
1 gr s.	24	24	1.3	51	24	1.4	54	24	0.9	86	24	1.7	141	18	149.3	1157	24	3.1	43	24	6.6	77
2 gr m.	24	24	3.7	85	24	3.8	126	24	2.8	170	24	2.2	209	10	173.3	2287	24	6.9	72	24	21.6	175
3 gr l.	24	24	18.6	238	24	17.1	490	24	15.3	535	24	4.2	515	0	—	4885	24	23.8	192	24	117.9	717
4 st s.	24	23	72.6	36	11	178.8	11	23	95.1	27	23	117.6	60	24	56.7	59	24	51.2	14	14	99.9	57
5 st m.	24	24	102.6	45	0	—	14	24	120.4	32	24	178.0	66	24	56.9	86	24	39.6	15	4	185.2	77
7 scc s.	24	24	0.1	16	24	0.8	3	24	0.1	9	24	1.1	58	24	1.8	76	24	0.1	12	24	0.1	12
8 scc m.	24	24	0.6	26	21	32.1	39	24	0.6	30	24	1.9	94	24	17.9	302	24	1.7	19	24	1.5	53
9 scc l.	24	24	2.8	33	22	13.0	110	24	5.1	51	24	3.6	115	22	60.2	575	24	9.6	30	24	5.9	131
Total	192	191	25.0	66	150	23.2	106	191	29.7	117	191	38.4	157	146	61.3	1179	192	17.0	50	162	36.0	162

7 Conclusion

Many semantics of abstract argumentation frameworks are naturally encoded in the language of circumscription. Based on the comparison with some of the best performant reasoners of ICCMA'15, the linear encodings implemented in PYGLAF appear to be a reasonable solution to the enumeration problems of abstract argumentation frameworks. Clearly, this is subject to the availability of an efficient solver for circumscription. Currently, CIRCUMSCRIPTINO is used, but PYGLAF can easily accommodate different solvers. Some of the acceptance problems are currently handled naively. This condition can be significantly improved by extending CIRCUMSCRIPTINO with native support for query answering, which we plan to address in the near future.

References

1. Alviano, M.: Model enumeration in propositional circumscription via unsatisfiable core analysis. TPLP 17, 1–18 (2017)
2. Alviano, M., Dodaro, C.: Answer set enumeration via assumption literals. In: Adorni, G., Cagnoni, S., Gori, M., Maratea, M. (eds.) AI*IA 2016, Genova, Italy, November 29 - December 1, 2016, Proceedings. LNCS, vol. 10037, pp. 149–163. Springer (2016), https://doi.org/10.1007/978-3-319-49130-1_12

Table 3. Enumeration of other extensions and Dung's Triathlon: solved instances, average execution time (seconds) on solved instances, and memory consumption (MB).

		SEMI-STABLE						STAGE					
		PYGLAF			CONARG2			PYGLAF			CONARG2		
Benchmark	#	sol	time	mem	sol	time	mem	sol	time	mem	sol	time	mem
1 gr small	24	24	1.3	50	24	1.5	55	24	0.6	42	24	26.2	970
2 gr med.	24	24	3.6	84	24	3.9	128	24	1.8	69	24	113.2	3527
3 gr large	24	24	18.3	238	24	17.5	496	24	10.0	189	23	111.6	8743
4 st small	24	22	103.8	56	11	128.3	35	17	78.8	72	17	173.0	19
5 st med.	24	21	75.9	66	3	483.9	51	20	44.8	81	15	269.6	33
7 scc small	24	24	0.1	8	17	38.5	23	1	510.2	114	0	—	56
8 scc med.	24	24	0.8	25	7	29.3	364	3	19.9	117	2	2.0	83
9 scc large	24	24	6.8	36	10	6.5	1469	7	0.0	117	6	2.1	169
Total	192	187	24.7	70	120	36.1	328	120	25.9	100	111	116.3	1133
		IDEAL						DUNG'S TRIATHLON					
		PYGLAF			CONARG2			PYGLAF			PYGLAF NAIVE		
Benchmark	#	sol	time	mem	sol	time	mem	sol	time	mem	sol	time	mem
1 gr small	24	24	3.1	52	24	2.5	88	24	2.5	50	24	3.9	52
2 gr med.	24	24	9.2	85	24	6.6	210	24	7.0	84	24	10.8	86
3 gr large	24	24	50.0	238	24	29.9	832	40	21.7	142	40	33.3	149
4 st small	24	22	74.6	37	6	65.7	15	23	72.2	36	22	69.1	39
5 st med.	24	24	148.7	42	0	—	18	24	103.6	44	24	127.4	46
7 scc small	24	24	0.1	20	24	0.5	5	24	0.1	16	24	0.2	18
8 scc med.	24	24	0.7	27	21	11.0	69	24	0.9	25	24	1.7	28
9 scc large	24	24	2.2	34	22	12.6	303	8	0.3	21	8	0.4	21
Total	192	190	35.7	67	145	12.8	193	191	27.6	62	190	33.2	65

3. Alviano, M., Dodaro, C.: Anytime answer set optimization via unsatisfiable core shrinking. TPLP 16(5-6), 533–551 (2016)
4. Alviano, M., Dodaro, C.: Unsatisfiable core shrinking for anytime answer set optimization. In: Sierra, C. (ed.) IJCAI 2017, Melbourne, Australia, August 19-25, 2017. pp. 4781–4785. ijcai.org (2017)
5. Alviano, M., Dodaro, C., Faber, W., Leone, N., Ricca, F.: WASP: A native ASP solver based on constraint learning. In: Cabalar, P., Son, T.C. (eds.) LPNMR 2013, Corunna, Spain, September 15-19, 2013. Proceedings. LNCS, vol. 8148, pp. 54–66. Springer (2013)
6. Alviano, M., Dodaro, C., Leone, N., Ricca, F.: Advances in WASP. In: Calimeri, F., Ianni, G., Truszczynski, M. (eds.) Logic Programming and Nonmonotonic Reasoning - 13th International Conference, LPNMR 2015, Lexington, KY, USA, September 27-30, 2015. Proceedings. LNCS, vol. 9345, pp. 40–54. Springer (2015)
7. Alviano, M., Dodaro, C., Ricca, F.: A maxsat algorithm using cardinality constraints of bounded size. In: Yang, Q., Wooldridge, M. (eds.) IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015. pp. 2677–2683. AAAI Press (2015)
8. Audemard, G., Simon, L.: Predicting learnt clauses quality in modern SAT solvers. In: Boutilier, C. (ed.) IJCAI 2009, Pasadena, California, USA, July 11-17, 2009. pp. 399–404 (2009), `http://ijcai.org/Proceedings/09/Papers/074.pdf`
9. Beierle, C., Brons, F., Potyka, N.: A software system using a SAT solver for reasoning under complete, stable, preferred, and grounded argumentation semantics. In: Hölldobler, S., Krötzsch, M., Peñaloza, R., Rudolph, S. (eds.) KI 2015: Advances in Artificial Intelligence - 38th Annual German Conference on AI, Dresden, Germany, September 21-25, 2015, Proceedings. LNCS, vol. 9324, pp. 241–248. Springer (2015), `http://dx.doi.org/10.1007/978-3-319-24489-1_19`
10. Bistarelli, S., Rossi, F., Santini, F.: Enumerating extensions on random abstract-afs with argtools, aspartix, conarg2, and dung-o-matic. In: Bulling, N., van der Torre, L.W.N., Villata, S., Jamroga, W., Vasconcelos, W.W. (eds.) Computational Logic in Multi-Agent Systems - 15th International Workshop, CLIMA XV, Prague, Czech Republic, August 18-19, 2014. Proceedings. LNCS, vol. 8624, pp. 70–86. Springer (2014), `http://dx.doi.org/10.1007/978-3-319-09764-0_5`
11. Bistarelli, S., Rossi, F., Santini, F.: Conarg: A tool for classical and weighted argumentation. In: Baroni, P., Gordon, T.F., Scheffler, T., Stede, M. (eds.) Computational Models of Argument - Proceedings of COMMA 2016, Potsdam, Germany, 12-16 September, 2016. Frontiers in Artificial Intelligence and Applications, vol. 287, pp. 463–464. IOS Press (2016)
12. Bistarelli, S., Santini, F.: Conarg: Argumentation with constraints. In: Ossowski, S., Toni, F., Vouros, G.A. (eds.) Proceedings of the First International Conference on Agreement Technologies, AT 2012, Dubrovnik, Croatia, October 15-16, 2012. CEUR Workshop Proceedings, vol. 918, pp. 197–198. CEUR-WS.org (2012)
13. Caminada, M.: On the issue of reinstatement in argumentation. In: Fisher, M., van der Hoek, W., Konev, B., Lisitsa, A. (eds.) Logics in Artificial Intelligence, 10th European Conference, JELIA 2006, Liverpool, UK, September 13-15, 2006, Proceedings. LNCS, vol. 4160, pp. 111–123. Springer (2006)
14. Caminada, M.: A labelling approach for ideal and stage semantics. Argument & Computation 2(1), 1–21 (2011)
15. Caminada, M.W.A., Carnielli, W.A., Dunne, P.E.: Semi-stable semantics. J. Log. Comput. 22(5), 1207–1254 (2012), `http://dx.doi.org/10.1093/logcom/exr033`
16. Caminada, M.W.A., Gabbay, D.M.: A logical account of formal argumentation. Studia Logica 93(2-3), 109–145 (2009), `http://dx.doi.org/10.1007/s11225-009-9218-x`

17. Cerutti, F., Dunne, P.E., Giacomin, M., Vallati, M.: Computing preferred extensions in abstract argumentation: A sat-based approach. In: Black, E., Modgil, S., Oren, N. (eds.) Theory and Applications of Formal Argumentation - Second International Workshop, TAFA 2013, Beijing, China, August 3-5, 2013, Revised Selected papers. LNCS, vol. 8306, pp. 176–193. Springer (2013), http://dx.doi.org/10.1007/978-3-642-54373-9_12
18. Cerutti, F., Giacomin, M., Vallati, M.: Argsemsat: Solving argumentation problems using SAT. In: Parsons, S., Oren, N., Reed, C., Cerutti, F. (eds.) Computational Models of Argument - Proceedings of COMMA 2014, Atholl Palace Hotel, Scottish Highlands, UK, September 9-12, 2014. Frontiers in Artificial Intelligence and Applications, vol. 266, pp. 455–456. IOS Press (2014), http://dx.doi.org/10.3233/978-1-61499-436-7-455
19. Charwat, G., Dvorák, W., Gaggl, S.A., Wallner, J.P., Woltran, S.: Methods for solving reasoning problems in abstract argumentation - A survey. Artif. Intell. 220, 28–63 (2015), http://dx.doi.org/10.1016/j.artint.2014.11.008
20. Dodaro, C., Alviano, M., Faber, W., Leone, N., Ricca, F., Sirianni, M.: The birth of a WASP: preliminary report on a new ASP solver. In: Fioravanti, F. (ed.) Proceedings of the 26th Italian Conference on Computational Logic, Pescara, Italy, August 31 - September 2, 2011. CEUR Workshop Proceedings, vol. 810, pp. 99–113. CEUR-WS.org (2011), http://ceur-ws.org/Vol-810/paper-l06.pdf
21. Dung, P.M.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artif. Intell. 77(2), 321–358 (1995), http://dx.doi.org/10.1016/0004-3702(94)00041-X
22. Dung, P.M., Mancarella, P., Toni, F.: Computing ideal sceptical argumentation. Artif. Intell. 171(10-15), 642–674 (2007), http://dx.doi.org/10.1016/j.artint.2007.05.003
23. Faber, W., Vallati, M., Cerutti, F., Giacomin, M.: Solving set optimization problems by cardinality optimization with an application to argumentation. In: Kaminka, G.A., Fox, M., Bouquet, P., Hüllermeier, E., Dignum, V., Dignum, F., van Harmelen, F. (eds.) ECAI 2016 - 22nd European Conference on Artificial Intelligence, 29 August-2 September 2016, The Hague, The Netherlands - Including Prestigious Applications of Artificial Intelligence (PAIS 2016). Frontiers in Artificial Intelligence and Applications, vol. 285, pp. 966–973. IOS Press (2016), http://dx.doi.org/10.3233/978-1-61499-672-9-966
24. Gaggl, S.A., Manthey, N., Ronca, A., Wallner, J.P., Woltran, S.: Improved answer-set programming encodings for abstract argumentation. TPLP 15(4-5), 434–448 (2015), http://dx.doi.org/10.1017/S1471068415000149
25. Lifschitz, V.: On the satisfiability of circumscription. Artif. Intell. 28(1), 17–27 (1986), http://dx.doi.org/10.1016/0004-3702(86)90028-7
26. McCarthy, J.: Circumscription - A form of non-monotonic reasoning. Artif. Intell. 13(1-2), 27–39 (1980), http://dx.doi.org/10.1016/0004-3702(80)90011-9
27. Verheij, B.: Two approaches to dialectical argumentation: Admissible sets and argumentation stages. In: In Proceedings of the biannual International Conference on Formal and Applied Practical Reasoning (FAPR) workshop. pp. 357–368. Universiteit (1996)
28. Wallner, J.P., Weissenbacher, G., Woltran, S.: Advanced SAT techniques for abstract argumentation. In: Leite, J., Son, T.C., Torroni, P., van der Torre, L., Woltran, S. (eds.) Computational Logic in Multi-Agent Systems - 14th International Workshop, CLIMA XIV, Corunna, Spain, September 16-18, 2013. Proceedings. LNCS, vol. 8143, pp. 138–154. Springer (2013), http://dx.doi.org/10.1007/978-3-642-40624-9_9