A SCC Recursive Meta-Algorithm for Computing Preferred Labellings in Abstract Argumentation

Federico Cerutti
School of Natural and Computing Science
King's College
University of Aberdeen
AB24 3UE, Aberdeen, United Kingdom
e-mail: f.cerutti@abdn.ac.uk

Massimiliano Giacomin
Department of Information Engineering
University of Brescia
via Branze, 38
25123, Brescia, Italy
e-mail: massimiliano.giacomin@ing.unibs.it

Mauro Vallati
School of Computing and Engineering
University of Huddersfield
Huddersfield, HD1 3DH, United Kingdom
e-mail: m.vallati@hud.ac.uk

Marina Zanella
Department of Information Engineering
University of Brescia
via Branze, 38
25123, Brescia, Italy
e-mail: marina.zanella@ing.unibs.it

Abstract

This paper presents a meta-algorithm for the computation of preferred labellings, based on the general recursive schema for argumentation semantics called SCC-Recursiveness. The idea is to recursively decompose a framework so as to compute semantics labellings on restricted sub-frameworks, in order to reduce the computational effort. The meta-algorithm can be instantiated with a specific "base algorithm", applied to the base case of the recursion, which can be obtained by generalizing existing algorithms in order to compute labellings in restricted sub-frameworks. We devise for this purpose a generalization of a SAT-based algorithm, and provide an empirical investigation to show the significant improvement of performances obtained by exploiting the SCC-recursive schema.

Introduction

Dung's theory of abstract argumentation (Dung 1995) is a unifying framework able to encompass a large variety of specific formalisms in the areas of nonmonotonic reasoning, logic programming and computational argumentation. It is based on the notion of argumentation framework (AF), consisting of a set of arguments and an *attack* relation between them. Different *argumentation semantics* introduce in a declarative way the criteria to determine which arguments emerge as justified from the conflict, by identifying a number of *extensions*, i.e. sets of arguments that can "survive the conflict together". In (Dung 1995) four "traditional" semantics were introduced, namely *complete*, *grounded*, *stable*, and *preferred* semantics. For an introduction on alternative semantics, see (Baroni, Caminada, and Giacomin 2011).

The main computational problems in abstract argumentation include *decision* and *construction* problems, and turn out to be computationally intractable for most of argumentation semantics (Dunne and Wooldridge 2009). In this paper we focus on the *extension enumeration* problem, i.e. constructing *all* extensions prescribed for a given AF: its solution provides complete information concerning the justification status of arguments and subsumes the solutions to the other problems.

On the practical side, few results are available on the development of efficient algorithms for abstract argumentation and their empirical assessment. In particular, in (Cerutti et al. 2013; 2014) a SAT-based approach has been proposed to solve the extension enumeration problem for preferred semantics. Preferred semantics represents the main contribution in Dung's theory, as it allows multiple extensions (differently from grounded semantics), the existence of extensions is always guaranteed (differently from stable semantics), and no extension is a proper subset of another extension (differently from complete semantics). The proposed approach basically performs a search in the space of complete extensions to enumerate the maximal ones, exploiting a SAT solver to identify the relevant search states. In (Cerutti et al. 2014) it has been shown to be competitive w.r.t. other state-of-the-art systems including ASPARTIX (Egly, Alice Gaggl, and Woltran 2010), ASPARTIX-META (Dvořák et al. 2011) and the system presented in (Nofal, Dunne, and Atkinson 2012). A limitation of this approach is that it is always applied to the AF as a whole, without dividing the enumeration problem into a number of simpler sub-problems.

In this paper, we aim at showing (i) an approach for dividing the problem of enumerating the preferred extensions into sub-problems, and (ii) that such an approach reduces the overall computational effort. In particular, we rely on the SCC-recursive schema, first introduced in (Baroni, Giacomin, and Guida 2005), which is a semantics definition schema where extensions are defined at the level of the

sub-frameworks identified by the strongly connected components. It is worth to mention that the focus of this paper is on preferred semantics, but the devised meta-algorithm can be instantiated with various specific "base algorithms" and applied to a variety of semantics.

The paper is organised as follows. After recallling some necessary background in the first section, the SCC-recursive schema is reviewed in the next one. Then the parametric meta-algorithm is presented, and the generalization of the SAT-based algorithm of (Cerutti et al. 2014) is described. Next section describes the test setting and comments the experimental results, while the last section provides a final discussion and concludes the paper. Proofs are omitted or sketched due to space limitations.

Background

An argumentation framework (Dung 1995) consists of a set of arguments[1] and a binary attack relation between them.

Definition 1. *An* argumentation framework *(AF) is a pair $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$ where $\mathcal{A}$ is a set of arguments and $\mathcal{R} \subseteq \mathcal{A} \times \mathcal{A}$. We say that $\boldsymbol{b}$* attacks *$\boldsymbol{a}$ iff $\langle \boldsymbol{b}, \boldsymbol{a} \rangle \in \mathcal{R}$, also denoted as $\boldsymbol{b} \rightarrow \boldsymbol{a}$. The set of attackers of an argument $\boldsymbol{a}$ will be denoted as $\boldsymbol{a}^- \triangleq \{\boldsymbol{b} : \boldsymbol{b} \rightarrow \boldsymbol{a}\}$, the set of arguments attacked by $\boldsymbol{a}$ will be denoted as $\boldsymbol{a}^+ \triangleq \{\boldsymbol{b} : \boldsymbol{a} \rightarrow \boldsymbol{b}\}$. We also extend these notations to sets of arguments, i.e. given $E \subseteq \mathcal{A}$, $E^- \triangleq \{\boldsymbol{b} \mid \exists \boldsymbol{a} \in E, \boldsymbol{b} \rightarrow \boldsymbol{a}\}$ and $E^+ \triangleq \{\boldsymbol{b} \mid \exists \boldsymbol{a} \in E, \boldsymbol{a} \rightarrow \boldsymbol{b}\}$.*

An argument $\mathbf{a}$ without attackers, i.e. such that $\mathbf{a}^- = \emptyset$, is said *initial*. Moreover, each argumentation framework has an associated directed graph where the vertices are the arguments, and the edges are the attacks.

The basic properties of conflict–freeness, acceptability, and admissibility of a set of arguments are fundamental for the definition of argumentation semantics.

Definition 2. *Given an AF $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$:*

- *a set $S \subseteq \mathcal{A}$ is a* conflict–free *set of Γ if $\nexists\, \boldsymbol{a}, \boldsymbol{b} \in S$ s.t. $\boldsymbol{a} \rightarrow \boldsymbol{b}$;*
- *an argument $\boldsymbol{a} \in \mathcal{A}$ is* acceptable *with respect to a set $S \subseteq \mathcal{A}$ of Γ if $\forall \boldsymbol{b} \in \mathcal{A}$ s.t. $\boldsymbol{b} \rightarrow \boldsymbol{a}$, $\exists\, \boldsymbol{c} \in S$ s.t. $\boldsymbol{c} \rightarrow \boldsymbol{b}$;*
- *a set $S \subseteq \mathcal{A}$ is an* admissible set *of Γ if S is a conflict–free set of Γ and every element of S is acceptable with respect to S of Γ.*

An argumentation semantics σ prescribes for any AF Γ a set of *extensions*, denoted as $\mathcal{E}_\sigma(\Gamma)$, namely a set of sets of arguments satisfying the conditions dictated by σ. Here we need to recall the definitions of complete (denoted as $\mathcal{CO}$), grounded (denoted as $\mathcal{GR}$) and preferred (denoted as $\mathcal{PR}$) semantics only.

Definition 3. *Given an AF $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$:*

- *a set $S \subseteq \mathcal{A}$ is a* complete extension *of Γ, i.e. $S \in \mathcal{E}_{\mathcal{CO}}(\Gamma)$, iff S is admissible and $\forall \boldsymbol{a} \in \mathcal{A}$ s.t. $\boldsymbol{a}$ is acceptable w.r.t. S, $\boldsymbol{a} \in S$;*
- *a set $S \subseteq \mathcal{A}$ is the* grounded extension *of Γ, i.e. $S \in \mathcal{E}_{\mathcal{GR}}(\Gamma)$, iff S is the minimal (w.r.t. set inclusion) complete extension of Γ. Its existence and uniqueness have been proved in (Dung, Mancarella, and Toni 2006);*
- *a set $S \subseteq \mathcal{A}$ is a* preferred extension *of Γ, i.e. $S \in \mathcal{E}_{\mathcal{PR}}(\Gamma)$, iff S is a maximal (w.r.t. set inclusion) complete extension of Γ.*

It can be noted that each extension S implicitly defines a three-valued *labelling* of arguments: an argument $\mathbf{a}$ is labelled `in` iff $\mathbf{a} \in S$; is labelled `out` iff $\exists\, \mathbf{b} \in S$ s.t. $\mathbf{b} \rightarrow \mathbf{a}$; is labelled `undec` if neither of the above conditions holds. In the light of this correspondence, argumentation semantics can be equivalently defined in terms of labellings rather than of extensions (Caminada 2006; Baroni, Caminada, and Giacomin 2011). For technical reasons, we introduce the notion of labelling both for argumentation frameworks and for arbitrary sets of arguments.

Definition 4. *Given a set of arguments S, a* labelling *of S is a total function $\mathcal{L}ab : S \longrightarrow \{\mathtt{in}, \mathtt{out}, \mathtt{undec}\}$. The set of all* labellings *of S is denoted as $\mathfrak{L}_S$. Given an AF $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$, a* labelling *of Γ is a labelling of $\mathcal{A}$. The set of all* labellings *of Γ is denoted as $\mathfrak{L}(\Gamma)$.*

In particular, complete labellings can be defined as follows.

Definition 5. *Let $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$ be an argumentation framework. A labelling $\mathcal{L}ab \in \mathfrak{L}(\Gamma)$ is a* complete labelling *of Γ iff it satisfies the following conditions for any $\boldsymbol{a} \in \mathcal{A}$:*

- *$\mathcal{L}ab(\boldsymbol{a}) = \mathtt{in} \Leftrightarrow \forall \boldsymbol{b} \in \boldsymbol{a}^- \mathcal{L}ab(\boldsymbol{b}) = \mathtt{out}$;*
- *$\mathcal{L}ab(\boldsymbol{a}) = \mathtt{out} \Leftrightarrow \exists \boldsymbol{b} \in \boldsymbol{a}^- : \mathcal{L}ab(\boldsymbol{b}) = \mathtt{in}$;*

The grounded and preferred labelling can then be defined on the basis of complete labellings.

Definition 6. *Let $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$ be an argumentation framework. A labelling $\mathcal{L}ab \in \mathfrak{L}(\Gamma)$ is* the grounded labelling *of Γ if it is the complete labelling of Γ minimizing the set of arguments labelled* `in`*, and it is a* preferred labelling *of Γ if it is a complete labelling of Γ maximizing the set of arguments labelled* `in`*.*

In order to show the connection between extensions and labellings, let us recall the definition of the function `Ext2Lab`, returning the labelling corresponding to a conflict–free set of arguments S.

Definition 7. *Given an AF $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$ and a conflict–free set $S \subseteq \mathcal{A}$, the corresponding labelling $\mathtt{Ext2Lab}(S)$ is defined as $\mathtt{Ext2Lab}(S) \equiv \mathcal{L}ab$, where*

- *$\mathcal{L}ab(\boldsymbol{a}) = \mathtt{in} \Leftrightarrow \boldsymbol{a} \in S$*
- *$\mathcal{L}ab(\boldsymbol{a}) = \mathtt{out} \Leftrightarrow \exists\, \boldsymbol{b} \in S$ s.t. $\boldsymbol{b} \rightarrow \boldsymbol{a}$*
- *$\mathcal{L}ab(\boldsymbol{a}) = \mathtt{undec} \Leftrightarrow \boldsymbol{a} \notin S \wedge \nexists\, \boldsymbol{b} \in S$ s.t. $\boldsymbol{b} \rightarrow \boldsymbol{a}$*

(Caminada 2006) shows that there is a bijective correspondence between the complete, grounded, preferred extensions and the complete, grounded, preferred labellings, respectively.

Proposition 1. *Given an an AF $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$, $\mathcal{L}ab$ is a complete (grounded, preferred) labelling of Γ if and only if there is a complete (grounded, preferred) extension S of Γ such that $\mathcal{L}ab = \mathtt{Ext2Lab}(S)$.*

[1] In this paper we consider only *finite* sets of arguments: see (Baroni et al. 2013) for a discussion on infinite sets of arguments.

The set of complete labellings of Γ is denoted as $\mathcal{L}_{\mathcal{CO}}(\Gamma)$, the set of preferred labellings as $\mathcal{L}_{\mathcal{PR}}(\Gamma)$, while $\mathcal{L}_{\mathcal{GR}}(\Gamma)$ denotes the set including the grounded labelling.

SCC-Recursiveness Revisited

In (Baroni and Giacomin 2004) it has been recognised that each argumentation framework can be partitioned into a set of sub-frameworks in such a way that most common argumentation semantics (originally defined at the global level) can be equivalently defined at the level of these sub-frameworks. In particular, an extension-based semantics definition schema has been introduced, called *SCC (strongly connected component)-recursiveness*, based on the graph-theoretical notion of SCCs i.e. the equivalence classes induced by the path equivalence (i.e. mutual reachability) relation between the nodes of the associated graph of an argumentation framework (given the direct correspondence between each AF and its associated graph, in the following we will equivalently refer to both of them as AF).

Definition 8. *Given an AF $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$, the binary relation of* path-equivalence *between nodes, denoted as $PE_\Gamma \subseteq \mathcal{A} \times \mathcal{A}$, is defined as follows:*

- $\forall \boldsymbol{a} \in \mathcal{A}, \langle \boldsymbol{a}, \boldsymbol{a} \rangle \in PE_\Gamma$;
- $\forall \boldsymbol{a}, \boldsymbol{b} \in \mathcal{A}$, $\boldsymbol{a} \neq \boldsymbol{b}$, $\langle \boldsymbol{a}, \boldsymbol{b} \rangle \in PE_\Gamma$ *iff there is a path from* $\boldsymbol{a}$ *to* $\boldsymbol{b}$ *and vice versa.*

Two main features of the SCC-recursive schema are worth remarking.

First, the SCC-recursive schema exploits the partial order of SCCs induced by the attack relation and can be regarded as a constructive procedure to incrementally build extensions following such partial order. At the beginning, the extensions of the frameworks restricted to the *initial* SCCs (i.e. those not receiving attacks from others) are computed and combined together. Then each SCC which is attacked only from initial SCCs is considered, and for each extension E already obtained, the extensions of such a SCC are locally computed and merged with E. The process is then applied to all SCCs following their partial order, until no remaining SCCs are left to process.

A second feature is that the schema, as the name suggests, is recursive. In particular, for every SCC considered in the above procedure, the local computation is performed as follows. First, all arguments attacked by the extension selected in the previous SCCs are suppressed. Then, the procedure is recursively applied to the remaining part of the SCC. The base of the recursion is reached when there is one SCC only: in this case, a base function $\mathcal{BF}$ is called. Such a function receives as input an argumentation framework $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$ and a set $C \subseteq \mathcal{A}$: Γ is a sub-framework of the original AF $\Gamma' = \langle \mathcal{A}', \mathcal{R}' \rangle$, and C contains the arguments that are "externally accepted" (i.e. their attackers in $\mathcal{A}' \setminus \mathcal{A}$ are attacked by the part of the extension constructed so far).

The following definitions introduce the SCC-recursive schema in its original extension-based form (Baroni, Giacomin, and Guida 2005).

First, let us recall the definition of *restriction* of an AF Γ to a set of arguments I, in symbol $\Gamma{\downarrow_I}$.

Definition 9. *Given an argumentation framework $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$ and a set $I \subseteq \mathcal{A}$, the restriction of Γ to I is defined as $\Gamma{\downarrow_I} \equiv (\mathcal{A} \cap I, \mathcal{R} \cap (I \times I))$.*

Then, Definition 10 introduces the function $\mathcal{GF}(\Gamma, C)$ which recursively computes the semantics extensions on the basis of the SCCs of Γ. Let us denote as SCC_Γ the set including the SCCs of an argumentation framework Γ.

Definition 10. *A given argumentation semantics σ is SCC-recursive if for any argumentation framework $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$, $\mathcal{E}_\sigma(\Gamma) = \mathcal{GF}(\Gamma, \mathcal{A}) \subseteq 2^{\mathcal{A}}$. For any $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$ and for any set $C \subseteq \mathcal{A}$, $E \in \mathcal{GF}(\Gamma, C)$ if and only if*

- $E \in \mathcal{BF}_\sigma(\Gamma, C)$ *if* $|SCC_\Gamma| = 1$
- $\forall S \in SCC_\Gamma \ (E \cap S) \in \mathcal{GF}(\Gamma{\downarrow_{S \setminus (E \setminus S)^+}}, U_\Gamma(S, E) \cap C)$ *otherwise*

where

- $\mathcal{BF}_\sigma(\Gamma, C)$ *is a function, called* base function*, that, given an argumentation framework $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$ such that $|SCC_\Gamma| = 1$ and a set $C \subseteq \mathcal{A}$, gives a subset of $2^{\mathcal{A}}$*
- $U_\Gamma(S, E) = \{\boldsymbol{a} \in S \setminus (E \setminus S)^+ \mid \forall \boldsymbol{b} \in (\boldsymbol{a}^- \setminus S), \boldsymbol{b} \in E^+\}$

As shown below, this schema is based on the notions of admissible set, complete, grounded, preferred extension of an AF *in a set of arguments*.

Definition 11. *Given an AF $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$ and a set $C \subseteq \mathcal{A}$, a set $E \subseteq \mathcal{A}$ is:*

- *an* admissible set of Γ in C *if and only if E is an admissible set of Γ and $E \subseteq C$*
- *a* complete extension of Γ in C *if and only if E is an admissible set of Γ in C, and every argument $\alpha \in C$ which is acceptable with respect to E belongs to E*
- *the* grounded extension of Γ in C *if and only if it is the least (with respect to set inclusion) complete extension of Γ in C*
- *a* preferred extension of Γ in C *if and only if it is a maximal (with respect to set inclusion) complete extension of Γ in C.*

The existence and uniqueness of the grounded extension in C, as well as the existence of at least a preferred extension in C, have been proved in (Baroni, Giacomin, and Guida 2005). The set of admissible sets in C is denoted as $\mathcal{E}_{\mathcal{AD}}(\Gamma, C)$, the set of complete extensions in C as $\mathcal{E}_{\mathcal{CO}}(\Gamma, C)$, the set of preferred extensions in C as $\mathcal{E}_{\mathcal{PR}}(\Gamma, C)$, while $\mathcal{E}_{\mathcal{GR}}(\Gamma, C)$ denotes the set including the grounded extension in C.

(Baroni, Giacomin, and Guida 2005) proves that $\mathcal{GF}(\Gamma, C)$, as defined in Def. 10, returns $\mathcal{E}_\sigma(\Gamma, C)$ (with $\sigma \in \{\mathcal{CO}, \mathcal{GR}, \mathcal{PR}\}$), provided that $\mathcal{BF}_\sigma(\Gamma, C)$ returns the complete, grounded, and preferred extensions in C, respectively. The correctness of the schema then follows from the fact that, according to Definition 11, $\mathcal{E}_\sigma(\Gamma, \mathcal{A}) = \mathcal{E}_\sigma(\Gamma)$.

A Parametric Meta-Algorithm to Enumerate Preferred Labellings

In this section, we develop a "meta-algorithm" based on the SCC-recursive schema for the computation of preferred labellings. We use the term meta-algorithm since it is parametric w.r.t. an algorithm B-PR to compute the base function

Algorithm 1 Enumerating the preferred labellings of an AF

PREF(Γ)

1: **Input:** $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$
2: **Output:** $E_p \in 2^{\mathfrak{L}(\Gamma)}$
3: **return R-PREF**$(\Gamma, \mathcal{A})$

$\mathcal{BF}_{\mathcal{PR}}$, without committing on the way B-PR performs its computation.

The meta-algorithm consists of Algorithms 1 and 2. It represents an implementation of the SCC-recursive schema with two main improvements. First, the schema has been adapted to the labelling-based approach: the following definition introduces the labelling-based counterpart of the complete, grounded and preferred extension in C.

Definition 12. *Given an AF $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$ and a set $C \subseteq \mathcal{A}$, a labelling $\mathcal{L}ab \in \mathfrak{L}(\Gamma)$ is a* complete (grounded, preferred) labelling of Γ in C *if there is a complete (grounded, preferred) extension S of Γ in C such that $\mathcal{L}ab = \texttt{Ext2Lab}(S)$. The set of complete and preferred labellings of Γ in C are denoted as $\mathcal{L}_{\mathcal{CO}}(\Gamma, C)$ and $\mathcal{L}_{\mathcal{PR}}(\Gamma, C)$, respectively, and the set including the grounded labelling of Γ in C is denoted as $\mathcal{L}_{\mathcal{GR}}(\Gamma, C)$.*

Finally, a pre-processing step computing the grounded labelling $\mathcal{L}ab^*$ in C is performed. The labellings of the whole argumentation framework can then be obtained by extending $\mathcal{L}ab^*$, on the basis to the following result.

Proposition 2. *Let $\langle \mathcal{A}, \mathcal{R} \rangle$ be an argumentation framework and let $C \subseteq \mathcal{A}$ a set of arguments. Considering the grounded labelling $\mathcal{L}ab^*$ of Γ in C and the set U including the* undec*-labelled arguments according to $\mathcal{L}ab^*$, it holds that $\mathcal{L}_{\mathcal{PR}}(\Gamma, C) = \{\mathcal{L}ab^* \cup E \mid E \in \mathcal{L}_{\mathcal{PR}}(\Gamma{\downarrow_U}, C \cap U)\}$.*

Sketch of proof. This extends the known result (Baroni, Caminada, and Giacomin 2011) that the grounded extension is a subset of the intersections of preferred extensions. □

Let us turn to the description of the meta-algorithm. The function PREF (Algorithm 1) receives as input an argumentation framework $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$ and returns the set of preferred labellings of Γ. This is simply achieved by invoking (at line 3) R-PREF$(\Gamma, \mathcal{A})$, where the function R-PREF ($\mathcal{GF}$ in Def. 10) receives as input an argumentation framework $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$ and a set $C \subseteq \mathcal{A}$, and computes the set $\mathcal{L}_{\mathcal{PR}}(\Gamma, C)$, i.e. the set of preferred labellings of Γ in C.

Algorithm 2 implements the function R-PREF. First, a pre-processing step which computes the grounded labelling in C is executed at line 3 by means of the call $(\mathcal{L}ab, U) =$ GROUNDED(Γ, C).

The procedure GROUNDED (Alg. 3) receives as input an argumentation framework $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$ and a set $C \subseteq \mathcal{A}$, and returns $\mathcal{L}ab$ and U: $\mathcal{L}ab$ is the restriction of the grounded labelling to the arguments that are `in` or `out`-labelled; U is the set including the remaining (`undec`-labelled) arguments. The GROUNDED procedure iteratively labels `in` each argument $\mathbf{a} \in C$ which either is initial, or receives attacks from arguments labelled `out`, and then labels `out` each argument attacked by $\mathbf{a}$.

Algorithm 2 Enumerating the preferred labellings in a set C of an AF

R-PREF(Γ, C)

1: **Input:** $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle, C \subseteq \mathcal{A}$
2: **Output:** $E_p \in 2^{\mathfrak{L}(\Gamma)}$
3: $(\mathcal{L}ab, U) =$ GROUNDED(Γ, C)
4: $E_p := \{\mathcal{L}ab\}$
5: $\Gamma = \Gamma{\downarrow_U}$
6: $(S_1, \ldots, S_n) :=$ SCCSSEQ(Γ)
7: **for** $i \in \{1, \ldots, n\}$ **do**
8: $\quad E'_p := \emptyset$
9: $\quad$ **for** $\mathcal{L}ab \in E_p$ **do**
10: $\quad\quad (O, I) :=$ BOUNDCOND$(\Gamma, S_i, \mathcal{L}ab)$
11: $\quad\quad$ **if** $I = \emptyset$ **then**
12: $\quad\quad\quad \mathcal{L}ab = \mathcal{L}ab \cup \{(\mathbf{a}, \texttt{out}) \mid \mathbf{a} \in O\} \cup \{(\mathbf{a}, \texttt{undec}) \mid \mathbf{a} \in S_i \setminus O\}$
13: $\quad\quad\quad E'_p = E'_p \cup \{\mathcal{L}ab\}$
14: $\quad\quad$ **else**
15: $\quad\quad\quad$ **if** $O = \emptyset$ **then**
16: $\quad\quad\quad\quad E_* =$ B-PR$(\Gamma{\downarrow_{S_i}}, I \cap C)$
17: $\quad\quad\quad$ **else**
18: $\quad\quad\quad\quad \mathcal{L}ab = \mathcal{L}ab \cup \{(\mathbf{a}, \texttt{out}) \mid \mathbf{a} \in O\}$
19: $\quad\quad\quad\quad E_* =$ R-PREF$(\Gamma{\downarrow_{S_i \setminus O}}, I \cap C)$
20: $\quad\quad\quad$ **end if**
21: $\quad\quad\quad E'_p = E'_p \cup (\mathcal{L}ab \otimes E_*)$
22: $\quad\quad$ **end if**
23: $\quad$ **end for**
24: $\quad E_p := E'_p$
25: **end for**
26: **return** E_p

After the computation of $\mathcal{L}ab$ and U, Alg. 2 initialises the variable E_p to $\{\mathcal{L}ab\}$ at line 4. E_p stores the set of labellings that are incrementally constructed. Then the computation proceeds by considering the argumentation framework $\Gamma{\downarrow_U}$ (line 5).

At line 6 the strongly connected components of the argumentation framework are identified. We assume that an algorithm is available, denoted as SCCSSEQ, which receives as input an argumentation framework Γ and returns as output a sequence $(S_1, \ldots, S_n)$ including the strongly connected components of Γ in a topological order, i.e. if $\exists \mathbf{a} \in S_i, \mathbf{b} \in S_j$ such that $\mathbf{a} \rightarrow \mathbf{b}$ then $i \leq j$. This can be done in linear time under the number of attacks (Cormen et al. 2009, p. 617).

The incremental construction of the preferred labellings is performed by the outer loop (lines 7-25), which iteratively selects the i-th SCC. At the beginning of the iteration, E_p includes the partial labellings constructed for $S_1 \cup \ldots \cup S_{i-1}$.Then the inner loop is entered, which extends any labelling $\mathcal{L}ab \in E_p$ to S_i (lines 9-23) storing the labellings thus obtained in E'_p (initialised at line 8). At line 10 variable O is set to include arguments of S_i that are attacked by "outside" `in`-labelled arguments according to $\mathcal{L}ab$, and variable I is set to include arguments of S_i that are only attacked by "outside" `out`-labelled arguments. Formally, BOUNDCOND$(\Gamma, S_i, \mathcal{L}ab)$ returns (O, I) where

Algorithm 3 Determining the grounded labelling of an AF in a set C

GROUNDED(Γ, C)

1: **Input:** $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle, C \subseteq \mathcal{A}$
2: **Output:** $(\mathcal{L}ab, U) : U \subseteq \mathcal{A}, \mathcal{L}ab \in \mathfrak{L}_{\mathcal{A} \setminus U}$
3: $\mathcal{L}ab := \emptyset$
4: $U := \mathcal{A}$
5: **repeat**
6: $initialfound := \bot$
7: **for a** $\in C$ **do**
8: **if** $\{\mathbf{b} \in U \mid \mathbf{b} \rightarrow \mathbf{a}\} = \emptyset$ **then**
9: $initialfound := \top$
10: $\mathcal{L}ab := \mathcal{L}ab \cup \{(\mathbf{a}, \mathtt{in})\}$
11: $U := U \setminus \mathbf{a}$
12: $C := C \setminus \mathbf{a}$
13: **for b** $\in (U \cap \mathbf{a}^+)$ **do**
14: $\mathcal{L}ab := \mathcal{L}ab \cup \{(\mathbf{b}, \mathtt{out})\}$
15: $U := U \setminus \mathbf{b}$
16: $C := C \setminus \mathbf{b}$
17: **end for**
18: **end if**
19: **end for**
20: **until** $(initialfound)$
21: **return**$(\mathcal{L}ab, U)$

$O = \{\mathbf{a} \in S_i \mid \exists \mathbf{b} \in S \cap \mathbf{a}^- : \mathcal{L}ab(\mathbf{b}) = \mathtt{in}\}$ and $I = \{\mathbf{a} \in S_i \mid \forall\, \mathbf{b} \in S \cap \mathbf{a}^-, \mathcal{L}ab(\mathbf{b}) = \mathtt{out}\}$, with $S \equiv S_1 \cup \ldots \cup S_{i-1}$.

If $I = \emptyset$ then no argument of S_i can be labelled `in`, in particular each argument in O is labelled `out` and each arguments in $S_i \setminus O$ is labelled `undec`. Accordingly, $\mathcal{L}ab$ is extended with such a labelling (line 12) and included in E'_p (line 13). Then a new iteration of the loop is entered to process a new labelling $\mathcal{L}ab \in E_p$ (if any).

Otherwise, two cases are considered. If $O = \emptyset$ then there are no arguments to suppress from S_i, thus the base case of the recursion applies. This is a small improvement w.r.t. Def. 10 which allows to avoid an unnecessary recursive step. In the base case, the preferred labellings of $\Gamma{\downarrow_{S_i}}$ in $S_i \cap C$ are computed by means of the algorithm B-PR, and assigned to E_* (line 16). If $O \neq \emptyset$ then each argument in O is labelled `out`, and the R-PREF procedure is recursively invoked on $\Gamma{\downarrow_{S_i \setminus O}}$ and $I \cap C$ (line 19). In both cases, line 21 extends the labelling $\mathcal{L}ab$ to cover the whole component S_i — by combining it with all labelling of E_* — and update E'_p. Indeed, $\mathcal{L}ab \otimes E_*$ denotes the set $\{\mathcal{L}ab \cup \mathcal{L}ab^* \mid \mathcal{L}ab^* \in E_*\}$.

After the inner loop is exited, i.e. when all labellings of E_p have been considered, the obtained labellings E'_p are those covering $S_1, \ldots, S_i$, thus E_p is set to E'_p at line 24. After the outer loop is exited all strongly connected components have been processed, thus E_p is returned as the set of preferred labellings in C (line 26).

The following theorem shows that Algorithm 2 (and thus Algorithm 1) is correct

Theorem 1. *Given an AF $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$ and a set $C \subseteq \mathcal{A}$, Algorithm 1 returns $E_p = \mathcal{L}_{\mathcal{PR}}(\Gamma, C)$.*

A SAT-Based Approach for the Base Case

R-PREF (Alg. 2) relies on an external algorithm (B-PR) for enumerating the preferred labellings of Γ in a set of arguments C. In order to empirically prove the improvements obtained using Alg. 2, in this section we discuss a specific implementation of B-PR generalizing the approach of (Cerutti et al. 2014).

The proposed algorithm performs a search in the space of complete extensions in C, in order to maximise the set of `in` arguments (cf. Definitions 12 and 11). Each step of the search process requires to encode in a propositional formula the constraints corresponding to complete labellings of an AF in C, with opportune modifications due to the search process. Then, a SAT solver checks whether the formula is satisfiable, i.e. there exists a truth assignment of the variables such that the formula evaluates to $\top$, and if this is the case returns such an assignment.

As a first step to identify the encoding, here we provide the definition of *complete labelling of Γ in C* in a more algorithmic way than Definition 12.

Definition 13. *Let $\langle \mathcal{A}, \mathcal{R} \rangle$ be an argumentation framework and $C \subseteq \mathcal{A}$ be a set of arguments. A total function $\mathcal{L}ab : \mathcal{A} \mapsto \{\mathtt{in}, \mathtt{out}, \mathtt{undec}\}$ is a* complete labelling of Γ in C *iff it satisfies the following conditions for any $\boldsymbol{a} \in C$:*

$\mathcal{L}^1_C$*:* $\mathcal{L}ab(\boldsymbol{a}) = \mathtt{in} \Leftrightarrow \forall \boldsymbol{b} \in \boldsymbol{a}^- \mathcal{L}ab(\boldsymbol{b}) = \mathtt{out}$*;*
$\mathcal{L}^2_C$*:* $\mathcal{L}ab(\boldsymbol{a}) = \mathtt{out} \Leftrightarrow \exists \boldsymbol{b} \in (\boldsymbol{a}^- \cap C) : \mathcal{L}ab(\boldsymbol{b}) = \mathtt{in}$*;*
$\mathcal{L}^3_C$*:* $\mathcal{L}ab(\boldsymbol{a}) = \mathtt{undec} \Leftrightarrow \forall \boldsymbol{b} \in (\boldsymbol{a}^- \cap C), \mathcal{L}ab(\boldsymbol{b}) \neq \mathtt{in} \wedge \exists \boldsymbol{c} \in \boldsymbol{a}^- : \mathcal{L}ab(\boldsymbol{c}) = \mathtt{undec}$*;*

and the following conditions for any $\boldsymbol{a} \in (\mathcal{A} \setminus C)$:

$\mathcal{L}^1_{\mathcal{A} \setminus C}$*:* $\mathcal{L}ab(\boldsymbol{a}) = \mathtt{out} \Leftrightarrow \exists \boldsymbol{b} \in (\boldsymbol{a}^- \cap C) : \mathcal{L}ab(\boldsymbol{b}) = \mathtt{in}$*;*
$\mathcal{L}^2_{\mathcal{A} \setminus C}$*:* $\mathcal{L}ab(\boldsymbol{a}) = \mathtt{undec} \Leftrightarrow \forall \boldsymbol{b} \in (\boldsymbol{a}^- \cap C), \mathcal{L}ab(\boldsymbol{b}) \neq \mathtt{in}$*.*

The following proposition shows that Definition 13 actually identifies the complete labellings in C as introduced in Definition 12.

Proposition 3. *Given an an AF $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$ and a set $C \subseteq \mathcal{A}$, $\mathcal{L}ab$ satisfies the conditions of Definition 13 if and only if there is a complete extension S of Γ in C such that $\mathcal{L}ab = \mathtt{Ext2Lab}(S)$.*

Sketch of proof. The proof generalizes (Cerutti et al. 2014, Proposition 2), to take into account the parameter C. □

The Complete Labelling Formula of Γ in C

Given an AF $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$ and a set $C \subseteq \mathcal{A}$, we are interested in identifying a boolean formula in conjunctive normal form (as required by the SAT solver), called *complete labelling formula of Γ in C* and denoted as $\Pi_{\Gamma, C}$, such that the satisfying assignments of the formula are in one-to-one correspondence with the complete labellings of Γ in C.

To this purpose, we have to introduce some notation. Let ϕ be a bijection $\phi : \{1, \ldots, |\mathcal{A}|\} \mapsto \mathcal{A}$ (the inverse map will be denoted as ϕ^{-1}): ϕ is an indexing of $\mathcal{A}$. Argument $\phi(i)$ will be sometimes referred to as argument i for brevity. For each argument i we define three boolean variables, I_i, O_i,

and U_i, with the intended meaning that I_i is $\top$ when argument i is labelled `in`, $\bot$ otherwise (and analogously O_i and U_i correspond to labels `out` and `undec`). Formally, given $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$ we define the corresponding set of variables as $\mathcal{V}(\Gamma) \triangleq \cup_{i \in \phi^{-1}(\mathcal{A})} \{I_i, O_i, U_i\}$.

The following definition expresses the constraints of Definition 5 in terms of the variables $\mathcal{V}(\Gamma)$.

Definition 14. *Given an AF $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$ with $|\mathcal{A}| = k$ and a set $C \subseteq \mathcal{A}$, let $\phi : \{1, \ldots, k\} \mapsto \mathcal{A}$ be an indexing of $\mathcal{A}$. The ENC_{all}* encoding *defined on the variables in $\mathcal{V}(\Gamma)$ is given by the conjunction of the formulae listed below:*

$$\bigwedge_{i \in \phi^{-1}(C)} \Big((I_i \vee O_i \vee U_i) \wedge (\neg I_i \vee \neg O_i) \wedge (\neg I_i \vee \neg U_i) \wedge (\neg O_i \vee \neg U_i) \Big) \tag{1}$$

$$\bigwedge_{i \in \phi^{-1}(C)} \left(I_i \vee \left(\bigvee_{\{j | \phi(j) \rightarrow \phi(i)\}} (\neg O_j) \right) \right) \tag{2}$$

$$\bigwedge_{i \in \phi^{-1}(C)} \left(\bigwedge_{\{j | \phi(j) \rightarrow \phi(i)\}} \neg I_i \vee O_j \right) \tag{3}$$

$$\bigwedge_{i \in \phi^{-1}(C)} \left(\bigwedge_{\{j \in \phi^{-1}(C) | \phi(j) \rightarrow \phi(i)\}} \neg I_j \vee O_i \right) \tag{4}$$

$$\bigwedge_{i \in \phi^{-1}(C)} \left(\neg O_i \vee \left(\bigvee_{\{j \in \phi^{-1}(C) | \phi(j) \rightarrow \phi(i)\}} I_j \right) \right) \tag{5}$$

$$\bigwedge_{i \in \phi^{-1}(C)} \Big(\bigwedge_{\{k | \phi(k) \rightarrow \phi(i)\}} \Big(U_i \vee \neg U_k \vee \Big(\bigvee_{\{j \in \phi^{-1}(C) | \phi(j) \rightarrow \phi(i)\}} I_j \Big) \Big) \Big) \tag{6}$$

$$\bigwedge_{i \in \phi^{-1}(C)} \Big(\Big(\bigwedge_{\{j \in \phi^{-1}(C) | \phi(j) \rightarrow \phi(i)\}} (\neg U_i \vee \neg I_j) \Big) \wedge \Big(\neg U_i \vee \Big(\bigvee_{\{k | \phi(k) \rightarrow \phi(i)\}} U_k \Big) \Big) \Big) \tag{7}$$

$$\bigwedge_{i \in \phi^{-1}(\mathcal{A} \setminus C)} \Big(\neg I_i \wedge (O_i \vee U_i) \wedge (\neg O_i \vee \neg U_i) \Big) \tag{8}$$

$$\bigwedge_{i \in \phi^{-1}(\mathcal{A} \setminus C)} \left(\bigwedge_{\{j \in \phi^{-1}(C) | \phi(j) \rightarrow \phi(i)\}} \neg I_j \vee O_i \right) \tag{9}$$

$$\bigwedge_{i \in \phi^{-1}(\mathcal{A} \setminus C)} \left(\neg O_i \vee \left(\bigvee_{\{j \in \phi^{-1}(C) | \phi(j) \rightarrow \phi(i)\}} I_j \right) \right) \tag{10}$$

$$\bigwedge_{i \in \phi^{-1}(\mathcal{A} \setminus C)} \left(U_i \vee \left(\bigvee_{\{j \in \phi^{-1}(C) | \phi(j) \rightarrow \phi(i)\}} I_j \right) \right) \tag{11}$$

$$\bigwedge_{i \in \phi^{-1}(\mathcal{A} \setminus C)} \left(\bigwedge_{\{j \in \phi^{-1}(C) | \phi(j) \rightarrow \phi(i)\}} \neg U_i \vee \neg I_j \right) \tag{12}$$

Formulas (1) and (8) encode the fact that $\mathcal{L}ab$ is a total function. The other formulas are in direct correspondence with the conditions of Definition 13:

- formulas (2) and (3) encode $\mathcal{L}^1_C$;
- formulas (4) and (5) encode $\mathcal{L}^2_C$;
- formulas (6) and (7) encode $\mathcal{L}^3_C$;
- formulas (9) and (10) encode $\mathcal{L}^1_{\mathcal{A} \setminus C}$;
- formulas (11) and (12) encode $\mathcal{L}^2_{\mathcal{A} \setminus C}$.

The following proposition shows that ENC_{all} is a complete labelling formula of Γ in C, i.e. every satisfying assignment of the ENC_{all} encoding corresponds to a complete extension in C, and vice versa. For ease of notation, if a variable is not assigned to $\top$ then it is $\bot$.

Proposition 4. *Let $\langle \mathcal{A}, \mathcal{R} \rangle$ be an argumentation framework and $C \subseteq \mathcal{A}$ be a set of arguments. If $\mathcal{L}ab$ is a complete labelling of Γ in C, then the assignment $\Phi_{\mathcal{V}(\Gamma)} \equiv \{(I_i, \top) \mid \mathcal{L}ab(\phi(i)) = \texttt{in}\} \cup \{(O_i, \top) \mid \mathcal{L}ab(\phi(i)) = \texttt{out}\} \cup \{(U_i, \top) \mid \mathcal{L}ab(\phi(i)) = \texttt{undec}\}$ satisfies the ENC_{all} encoding of Definition 14. Conversely, if $\Phi_{\mathcal{V}(\Gamma)}$ is a satisfying assignment of the ENC_{all} encoding, then the labelling $\mathcal{L}ab \equiv \{(\boldsymbol{a}, \texttt{in}) \mid I_{\phi^{-1}(\boldsymbol{a})} \in \Phi_{\mathcal{V}(\Gamma)}\} \cup \{(\boldsymbol{b}, \texttt{out}) \mid O_{\phi^{-1}(\boldsymbol{b})} \in \Phi_{\mathcal{V}(\Gamma)}\} \cup \{(\boldsymbol{c}, \texttt{undec}) \mid U_{\phi^{-1}(\boldsymbol{c})} \in \Phi_{\mathcal{V}(\Gamma)}\}$ is a complete labelling of Γ in C.*

Sketch of proof. The result derives from Proposition 3 by transforming logical implications in AND/OR. □

Several syntactically different encodings can be devised which, while being logically equivalent, can significantly affect the performance of the overall process (Cerutti et al. 2014). The following proposition shows the 20 non redundant encodings (i.e. it is not possible to drop out some clauses so as to obtain a simpler logically equivalent encoding) equivalent to ENC_{all}.

Proposition 5. *Let $\langle \mathcal{A}, \mathcal{R} \rangle$ be an argumentation framework and $C \subseteq \mathcal{A}$ be a set of arguments. Referring to the formulae listed in Definition 14, let us define the following formulae (for the arguments in C):*

$C_1^a : (1) \wedge (2) \wedge (3) \wedge (4) \wedge (5)$
$C_1^b : (1) \wedge (4) \wedge (5) \wedge (6) \wedge (7)$
$C_1^c : (1) \wedge (2) \wedge (3) \wedge (6) \wedge (7)$

$C_2: (1) \wedge (3) \wedge (5) \wedge (7)$
$C_3: (1) \wedge (2) \wedge (4) \wedge (6)$

as well as the following ones (for the arguments in $\mathcal{A} \setminus C$):

$NC_1^a: (8) \wedge (9) \wedge (10)$
$NC_1^b: (8) \wedge (11) \wedge (12)$
$NC_2: (8) \wedge (10) \wedge (12)$
$NC_3: (8) \wedge (9) \wedge (11)$

Any encoding of the form $C \wedge NC$, where $C \in \{C_1^a, C_1^b, C_1^c, C_2, C_3\}$ and $NC \in \{NC_1^a, NC_1^b, NC_2, NC_3\}$, is equivalent to ENC_{all}, i.e. it has the same satisfying assignments of the variables in $\mathcal{V}(\Gamma)$.

Sketch of proof. The proof exploits the fact that $\mathcal{L}ab$ is a total function and shows that the conditions in $C \wedge NC$ entail those included in ENC_{all}. □

Corollary 1. *Let $\langle \mathcal{A}, \mathcal{R} \rangle$ be an argumentation framework and $C \subseteq \mathcal{A}$ be a set of arguments. Any encoding of the form $C \wedge NC$ (see Proposition 5) is a complete labelling formula of Γ in C.*

The SAT_basepref Algorithm

We are now able to describe the proposed procedure to implement B-PR. The procedure, called SAT_basepref, resorts to five external functions: SS, I-ARGS, O-ARGS, U-ARGS and LAB. SS is a SAT solver able to prove unsatisfiability too: it accepts as input a CNF formula and returns a variable assignment satisfying the formula if it exists, ε otherwise. All the other functions accept as input a variable assignment concerning $\mathcal{V}(\Gamma)$: I-ARGS returns the corresponding set of arguments labelled as `in`, O-ARGS returns the corresponding set of arguments labelled as `out`, U-ARGS returns the corresponding set of arguments labelled as `undec`, while LAB returns the labelling corresponding to the input variable assignment. Moreover, INC_C denotes the clause $\bigvee_{i \in \phi^{-1}(C)} I_i$.

The SAT_basepref procedure (Alg. 4) initialises, at line 3, the variable E_p which stores the generated labellings. Then cnf is initialized to $\Pi_{\Gamma,C} \wedge INC_C$: $\Pi_{\Gamma,C}$ is a complete labelling formula of Γ in C, e.g. one of the equivalent encodings shown in Proposition 5, while INC_C restricts the search process to a non empty labelling (this is non restrictive due to the check of line 34). The search process is carried out by two nested **repeat-until** loops. Roughly, the inner loop (lines 8–24) resembles a depth-first search which, starting from a non-empty complete labelling in C, produces a sequence of complete labellings in C strictly ordered by set inclusion. When the sequence can no more be extended, its last element corresponds to a complete labelling in C maximizing the set of `in`-labelled arguments, namely to a preferred labelling. The outer loop (lines 5–33) drives the search: it ensures, through proper settings of the variables, that the inner loop is entered with different initial conditions, so that all preferred labellings in C are found.

The following theorem shows that Algorithm 4 is correct.

Theorem 2. *Given an AF $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$ and a set $C \subseteq \mathcal{A}$, Algorithm 4 returns $\mathcal{L}_{\mathcal{PR}}(\Gamma, C)$.*

Algorithm 4 Enumerating the preferred labellings in a set C of a AF

SAT_basepref(Γ, C)

1: **Input:** $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle, C \subseteq \mathcal{A}$
2: **Output:** $E_p \in 2^{\mathfrak{L}(\Gamma)}$
3: $E_p := \emptyset$
4: $cnf := \Pi_{\Gamma,C} \wedge INC_C$
5: **repeat**
6: $\quad prefcand := \emptyset$
7: $\quad cnfdf := cnf$
8: $\quad$ **repeat**
9: $\quad\quad lastcompfound := SS(cnfdf)$
10: $\quad\quad$ **if** $lastcompfound \,!= \varepsilon$ **then**
11: $\quad\quad\quad prefcand := lastcompfound$
12: $\quad\quad\quad$ **for a** $\in$ I-ARGS$(lastcompfound)$ **do**
13: $\quad\quad\quad\quad cnfdf := cnfdf \wedge I_{\phi^{-1}(\mathbf{a})}$
14: $\quad\quad\quad$ **end for**
15: $\quad\quad\quad$ **for a** $\in$ O-ARGS$(lastcompfound)$ **do**
16: $\quad\quad\quad\quad cnfdf := cnfdf \wedge O_{\phi^{-1}(\mathbf{a})}$
17: $\quad\quad\quad$ **end for**
18: $\quad\quad\quad remaining := \bot$
19: $\quad\quad\quad$ **for a** $\in C \cap$ U-ARGS$(lastcompfound)$ **do**
20: $\quad\quad\quad\quad remaining := remaining \vee I_{\phi^{-1}(\mathbf{a})}$
21: $\quad\quad\quad$ **end for**
22: $\quad\quad\quad cnfdf := cnfdf \wedge remaining$
23: $\quad\quad$ **end if**
24: $\quad$ **until** $(lastcompfound \,!= \varepsilon)$
25: $\quad$ **if** $prefcand \,!= \emptyset$ **then**
26: $\quad\quad E_p := E_p \cup \{\text{LAB}(prefcand)\}$
27: $\quad\quad oppsolution := \bot$
28: $\quad\quad$ **for a** $\in C \setminus$ I-ARGS$(prefcand)$ **do**
29: $\quad\quad\quad oppsolution := oppsolution \vee I_{\phi^{-1}(\mathbf{a})}$
30: $\quad\quad$ **end for**
31: $\quad\quad cnf := cnf \wedge oppsolution$
32: $\quad$ **end if**
33: **until** $(prefcand \,!= \emptyset)$
34: **if** $E_p = \emptyset$ **then**
35: $\quad E_p = \{(\mathbf{a}, \texttt{undec}) \mid \mathbf{a} \in \mathcal{A}\}$
36: **end if**
37: **return** E_p

The Empirical Analysis

In this section, we present the results of a large experimental study examining the reduction of computational effort in enumerating preferred extensions due to the application of the proposed meta-algorithm (Alg. 1) instantiated with Alg. 4 as B-PR.

Experimental Setup

The experiments were performed on AMD Opteron™ 2.4 Ghz, 8 Gb of RAM and Linux operating system. As in the International Planning Competition (IPC) (Jiménez et al. 2012), a limit of 15 minutes was imposed to compute the preferred labellings for each AF. No limit was imposed on the RAM usage, but a run fails at saturation of the available memory. Moreover, we adopted the IPC speed score,

also borrowed from the planning community, which is defined as follows. For each AF, each system gets a score of $1/(1+\log_{10}(T/T^*))$, where T is its execution time and T^* the best execution time among the compared systems, or a score of 0 if it fails in that case. Runtimes below 0.01 sec get by default the maximal score of 1.

In order to evaluate PREF, we compared it with the implementation of the approach discussed in (Cerutti et al. 2014), which has been proved to be competitive w.r.t other state-of-the-art systems. In the following we refer to SCC-P as the prototype implementing the algorithm PREF with C_2 and NC_2, and SAT-P as the current version (rev. 128) of the software presented in (Cerutti et al. 2014) with C_2. We chose to compare SCC-P not with the implementation of Alg. 4 called as PREF$(\Gamma, \mathcal{A})$, but with SAT-P in order to give additional strength to this empirical evaluation. Both SCC-P and SAT-P are implemented in C++ and both adopt Glucose (Audemard and Simon 2009; 2012) as integrated SAT solver.

We designed three experiments aimed at addressing the following hypotheses:

$I1$: on Γ s.t. $|\text{SCC}_\Gamma| = 1$, SCC-P performs worse than SAT-P;

$I2$: there exists a value χ such that on Γ where $|\text{SCC}_\Gamma| > \chi$, SCC-P performs better that SAT-P;

$I3$: on Γ s.t. $|\text{SCC}_\Gamma| > \chi$, the greater $|\mathcal{E}_{\mathcal{PR}}(\Gamma)|$, the more SAT-P performs worse than SCC-P.

To check the first hypothesis, we conducted an experiment on 790 randomly generated AFs (Γ), s.t. $|\text{SCC}_\Gamma| = 1$, varying $\mathcal{A}$ between 25 and 250 with a step of 25.

Coming to the second hypothesis, we randomly generated 720 AFs varying $|\text{SCC}_\Gamma|$ between 5 and 45 with a step of 5. The size of the SCCs is determined by normal distributions with means between 20 and 40 with a step of 5, and with a fixed standard deviation of 5. We similarly varied the probability of having attacks between arguments among SCCs.

Finally, we addressed the third hypothesis conducting an experiment on 2800 AFs randomly generated as before, such that $50 \leq |\text{SCC}_\Gamma| \leq 80$ with a step of 5.

In the following we rely on the Wilcoxon Signed-Rank Test (WSRT) in order to identify significant subset of data.

Experiments on Γ s.t. $|\text{SCC}_\Gamma| = 1$

Figure 1 depicts the significant ($p < 0.05$) IPC values (normalised to 100) for SCC-P and SAT-P when $|\text{SCC}_\Gamma| = 1$ varying $|\mathcal{A}|$ and $|\mathcal{E}_{\mathcal{PR}}(\Gamma)|$. From this, we can observe that there is a significant statistical evidence supporting hypothesis $I1$. It is worth to mention that when $|\text{SCC}_\Gamma| = 1$, PREF (i) identifies the presence of a single SCC, (ii) calls the B-PR on the single computed SCC. Since B-PR is a variation (cf. Alg. 4) of the algorithm SAT-P, the great difference of performance shown in Fig. 1 suggests that refining the prototypical implementation we used in these experiments would lead to better results.

Experiments on Γ s.t. $5 \leq |\text{SCC}_\Gamma| \leq 45$

Figure 2 summarises the results obtained in the experiment aimed at proving hypothesis $I2$, about the existence of a number χ such that on Γ with $|\text{SCC}_\Gamma| > \chi$, SCC-P performs better than SAT-P.

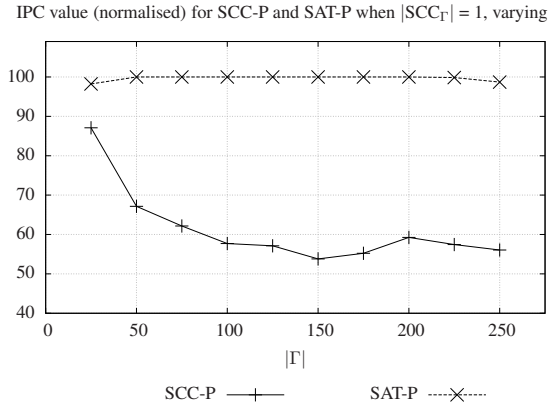

Figure 1: Significant ($p < 0.05$) IPC values (normalised) for SCC-P and SAT-P when $|\text{SCC}_\Gamma| = 1$ varying $|\mathcal{A}|$.

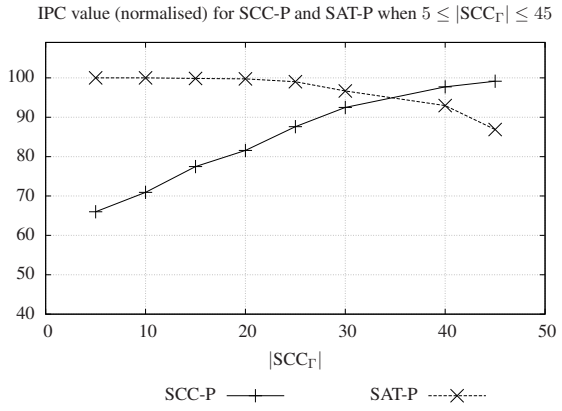

Figure 2: Significant ($p < 0.05$) IPC values (normalised) for SCC-P and SAT-P when $5 \leq |\text{SCC}_\Gamma| \leq 45$ varying $|\text{SCC}_\Gamma|$.

In particular, it turns out that:

- for $|\text{SCC}_\Gamma| = 30$, $Md(\text{SCC-P}) = 6.43$, $Md(\text{SAT-P}) = 5.63$, $z = -2.87$, $p < 0.01$;
- for $|\text{SCC}_\Gamma| = 35$, $Md(\text{SCC-P}) = 8.81$, $Md(\text{SAT-P}) = 8.53$, $z = -0.35$, $p = 0.73$;
- for $|\text{SCC}_\Gamma| = 40$, $Md(\text{SCC-P}) = 10.49$, $Md(\text{SAT-P}) = 12.53$, $z = -3.56$, $p < 0.01$;

where $Md(\cdot)$ indicates the median of execution times.

In other terms, when $|\text{SCC}_\Gamma| = 35$, the performances of SCC-P and SAT-P are statistically indistinguishable ($p = 0.73 > 0.05$). However, for $|\text{SCC}_\Gamma| > 35$, Fig. 2 shows that SCC-P performs significantly ($p < 0.05$) better than SAT-P.

Experiments on Γ s.t. $50 \leq |\text{SCC}_\Gamma| \leq 80$

Figure 3 depicts the medians of significant execution times of SCC-P and SAT-P varying the number of computed labellings. It illustrates a dependency of the execution times

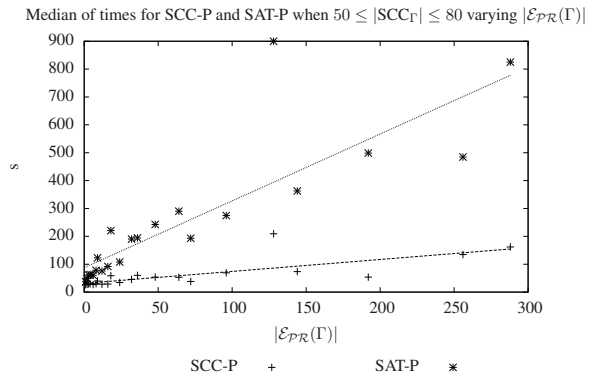

Figure 3: Significant median values for SCC-P and SAT-P when $50 \leq |\text{SCC}_\Gamma| \leq 80$ varying $|\mathcal{E}_{\mathcal{PR}}(\Gamma)|$, and their regression to the function $f(x) = a\ x + b$: SCC-P, $a = 0.43$, $b = 31.33$; SAT-P, $a = 2.40$, $b = 87.53$.

on the number of computed preferred labellings. Given the linear regression of these data, it turns out that the execution times of SAT-P grows $\approx 2 \cdot |\mathcal{E}_{\mathcal{PR}}(\Gamma)|$, while SCC-P grows $\approx \frac{1}{2} \cdot |\mathcal{E}_{\mathcal{PR}}(\Gamma)|$, thus supporting hypothesis $I3$.

Moreover, the third experiment provides additional statistically significant ($p < 0.05$) evidence in support of hypothesis $I2$, as Figure 3 summarises. Roughly speaking, computing the increment of performance as the average of the difference of execution times normalised on the maximum time between SCC-P and SAT-P, we can conclude that there is a significant increment of performances up to 56%.

Conclusions

In this paper we devised and evaluated a general parametric algorithm in order to decompose the preferred extensions enumeration problem into restricted subframeworks. To this purpose, several research challenges have been tackled.

First, an efficient algorithmic implementation of the SCC-recursive schema, originally introduced in a declarative form, has been developed, also leading to a labelling-based formulation. Let us notice that, recently, an approach has been proposed in (Liao, Lei, and Dai 2013) which partially exploits the SCC-recursive schema. In particular, while it borrows from (Baroni, Giacomin, and Guida 2005) the idea of decomposing the argumentation framework into its SCCs, it does not exploit the recursion step to achieve a deeper decomposition of the AF. Other "splitting" techniques have been proposed in literature. In particular, in the context of argumentation dynamics (Baumann et al. 2012) introduces a decomposition of an argumentation framework into two parts, while (Dvořák, Pichler, and Woltran 2012) exploits a tree-based decomposition. A general study of decomposability properties w.r.t. arbitrary partitions of an argumentation framework is also presented in (Baroni et al. 2012). Comparing different approaches for dividing an AF in subframeworks represents an interesting future work.

A second contribution of the present paper is the generalization of the approach in (Cerutti et al. 2014) to the computation of labellings in a restricted sub-framework, which is mandatory for applying the SCC-recursive schema. As acknowledged in (Cerutti et al. 2014), the relationship between argumentation semantics and the satisability problem has been already considered in the literature, e.g. in (Besnard and Doutre 2004; Dvořák et al. 2012; Arieli and Caminada 2013), where different encodings for labellings of an AF according to several semantics have been introduced. Generalizing these proposals in order to integrate them into the SCC-recursive schema is another challenge for future work.

Finally, an empirical investigation has been carried out to show a statistical evidence in favour of the reduction of computational effort by the exploitation of the SCC-recursive schema. In particular, as shown in the previous section, despite the fact that the current prototypical implementation needs further refinements (cf. the results of experiments on Γ s.t. $|\text{SCC}_\Gamma| = 1$), for Γ with $|\text{SCC}_\Gamma| > \chi = 35$ there is a statistical evidence showing how implementing the SCC-recursive schema reduces the computational effort of enumerating the preferred labellings. Moreover, as depicted in Fig. 3, the execution time of the SCC-recursive implementation is less sensible to the number of labellings, i.e. its execution time grows less than the non SCC-recursive implementation when the number of computed preferred labellings increases. This suggests to explore an hybrid approach where, at each recursion step, a choice is made on whether computing the preferred labellings with a direct call to B-PR or applying the SCC-recursive schema. Such choice can be based on an heuristic to estimate the number of SCCs and labellings, and can be easily implemented with an additional step after line 5 of Alg. 2.

Several other future works are envisaged. First, we want to strengthen the statistical evidence in favour of the SCC-recursive schema by exploiting and comparing different encodings of complete labellings derived from ENC_{all}, including the redundant ones. Moreover, Alg. 2 can be instantiated with a different B-PR algorithm, e.g. generalizing the algorithms presented in (Egly, Alice Gaggl, and Woltran 2010; Nofal, Atkinson, and Dunne 2014; Dvořák et al. 2014) for computing labellings in sub-frameworks.

Finally, we intend to apply the devised meta-algorithm to stable and CF2 semantics, which directly fit the SCC-recursive schema (Baroni, Giacomin, and Guida 2005), as well as to semi-stable and ideal semantics by exploiting the relationship with preferred semantics.

Acknowledgements

The authors would like to acknowledge the use of the University of Huddersfield Queensgate Grid in carrying out this work. The authors also thank the anonymous reviewers for their helpful comments.

References

Arieli, O., and Caminada, M. W. 2013. A QBF-based formalization of abstract argumentation semantics. *Journal of Applied Logic* 11(2):229–252.

Audemard, G., and Simon, L. 2009. Predicting Learnt Clauses Quality in Modern SAT Solvers. In *Proceedings*

of the 21st International Joint Conference on Artifical Intelligence (IJCAI 2009), 399–404.

Audemard, G., and Simon, L. 2012. GLUCOSE 2.1. `http://www.labri.fr/perso/lsimon/glucose/`.

Baroni, P., and Giacomin, M. 2004. A General Recursive Schema for Argumentation Semantics. In *Proceedings of the 14th European Conference on Artificial Intelligence (ECAI 2004)*, 783–787.

Baroni, P.; Boella, G.; Cerutti, F.; Giacomin, M.; van der Torre, L.; and Villata, S. 2012. On Input/Output Argumentation Frameworks. In *Proceedings of the 4th International Conference on Computational Models of Arguments (COMMA 2012)*, 358–365.

Baroni, P.; Cerutti, F.; Dunne, P. E.; and Giacomin, M. 2013. Automata for Infinite Argumentation Structures. *Artificial Intelligence* 203(0):104–150.

Baroni, P.; Caminada, M.; and Giacomin, M. 2011. An introduction to argumentation semantics. *Knowledge Engineering Review* 26(4):365–410.

Baroni, P.; Giacomin, M.; and Guida, G. 2005. SCC-recursiveness: a general schema for argumentation semantics. *Artificial Intelligence* 168(1-2):165–210.

Baumann, R.; Brewka, G.; Dvořák, W.; and Woltran, S. 2012. Parameterized Splitting: A Simple Modification-Based Approach. In Erdem, E.; Lee, J.; Lierler, Y.; and Pearce, D., eds., *Correct Reasoning*, volume 7265 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg. 57–71.

Besnard, P., and Doutre, S. 2004. Checking the acceptability of a set of arguments. In *Proceedings of the 10th International Workshop on Non-Monotonic Reasoning (NMR 2004)*, 59–64.

Caminada, M. 2006. On the Issue of Reinstatement in Argumentation. In *Proceedings of the 10th European Conference on Logics in Artificial Intelligence (JELIA 2006)*, 111–123.

Cerutti, F.; Dunne, P. E.; Giacomin, M.; and Vallati, M. 2013. A SAT-based Approach for Computing Extensions in Abstract Argumentation. In *Second International Workshop on Theory and Applications of Formal Argumentation (TAFA-13)*.

Cerutti, F.; Dunne, P. E.; Giacomin, M.; and Vallati, M. 2014. Computing Preferred Extensions in Abstract Argumentation: A SAT-Based Approach. In Black, E.; Modgil, S.; and Oren, N., eds., *TAFA 2013*, volume 8306 of *Lecture Notes in Computer Science*. Springer-Verlag Berlin Heidelberg. 176–193.

Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; and Stein, C. 2009. *Introduction to Algorithms*. MIT Press.

Dung, P. M.; Mancarella, P.; and Toni, F. 2006. A dialectic procedure for sceptical, assumption-based argumentation. In *Prooceedings of the 1st International Conference on Computational Models of Arguments (COMMA 2006)*, 145–156.

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):321–357.

Dunne, P. E., and Wooldridge, M. 2009. Complexity of abstract argumentation. In Rahwan, I., and Simari, G., eds., *Argumentation in AI*. Springer-Verlag. chapter 5, 85–104.

Dvořák, W.; Gaggl, S. A.; Wallner, J.; and Woltran, S. 2011. 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)*.

Dvořák, W.; Järvisalo, M.; Wallner, J. P.; and Woltran, S. 2012. Complexity-Sensitive Decision Procedures for Abstract Argumentation. In *Proceedings of 13th International Conference on Principles of Knowledge Representation and Reasoning (KR 2012)*, 54–64.

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

Dvořák, W.; Pichler, R.; and Woltran, S. 2012. Towards fixed-parameter tractable algorithms for abstract argumentation. *Artificial Intelligence* 186:1–37.

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

Jiménez, S.; de la Rosa, T.; Fernández, S.; Fernández, F.; and Borrajo, D. 2012. A review of machine learning for automated planning. *Knowledge Engineering Review* 27(4):433–467.

Liao, B.; Lei, L.; and Dai, J. 2013. Computing Preferred Labellings by Exploiting SCCs and Most Sceptically Rejected Arguments. In *Second International Workshop on Theory and Applications of Formal Argumentation (TAFA-13)*.

Nofal, S.; Atkinson, K.; and Dunne, P. E. 2014. Algorithms for decision problems in argument systems under preferred semantics. *Artificial Intelligence* 207:23–51.

Nofal, S.; Dunne, P. E.; and Atkinson, K. 2012. On Preferred Extension Enumeration in Abstract Argumentation. In *Proceedings of 3rd International Conference on Computational Models of Arguments (COMMA 2012)*, 205–216.