
THE PYGLAF ARGUMENTATION REASONER (ICCMA2021)

Mario Alviano

Department of Mathematics and Computer Science
University of Calabria
Rende (CS), IT 87036
mario.alviano@unical.it

ABSTRACT

The PYGLAF reasoner takes advantage of circumscription to solve computational problems of abstract argumentation frameworks. In fact, many of these problems 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. Within PYGLAF, Python is used to build the encodings and to control the execution of the external circumscription solver, which extends the SAT solver GLUCOSE and implements algorithms taking advantage of unsatisfiable core analysis and incremental computation.

1 Introduction

Circumscription [1] 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. 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. Since many semantics of abstract argumentation frameworks are based on a preference relation that essentially amount to inclusion relationships, PYGLAF (<http://alviano.com/software/pyglaf/>) uses circumscription as a target language to solve computational problems of abstract argumentation frameworks.

PYGLAF [2] is implemented in Python and uses CIRCUMSCRIPTINO (<http://alviano.com/software/circumscriptino/>), a circumscription solver extending the SAT solver GLUCOSE [3]. Linear reductions are used for all semantics [4]. For the ideal extension, the reduction requires the union of all admissible extensions of the input graph; such a set is computed by means of iterative calls to 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 reasoner supports the following ICCMA 2021 problems: CE-CO, CE-PR, CE-SST, CE-ST, CE-STG, DC-CO, DC-PR, DC-SST, DC-ST, DC-STG, DS-CO, DS-ID, DS-PR, DS-SST, DS-ST, DS-STG, SE-CO, SE-ID, SE-PR, SE-SST, SE-ST, and SE-STG.

2 Circumscription

Let \mathcal{A} be a fixed, countable set of *atoms* including \perp . A *literal* is an atom possibly preceded by the connective \neg . For a literal ℓ , let $\bar{\ell}$ denote its *complementary literal*, that is, $\bar{p} = \neg p$ and $\overline{\neg p} = p$ for all $p \in \mathcal{A}$; for a set L of literals, let \bar{L} be $\{\bar{\ell} \mid \ell \in L\}$. *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 $\neg\perp$; the set of atoms occurring in T is denoted by $atoms(T)$. An *assignment* is a set A of literals such that $A \cap \bar{A} = \emptyset$. An *interpretation* for a theory T is an assignment I such that $(I \cup \bar{I}) \cap \mathcal{A} = atoms(T)$. Relation \models is defined as usual. I is a *model* of a theory T if $I \models T$. Let $models(T)$ denote the set of models of T .

Circumscription applies to a theory T and a set P of literals subject to minimization. Formally, relation \leq^P is defined as follows: for I, J interpretations of T , $I \leq^P J$ if $I \cap P \subseteq J \cap P$. $I \in models(T)$ is a *preferred model* of T with respect to

\leq^P if there is no $J \in models(T)$ such that $I \not\leq^P J$ and $J \leq^P I$. Let $CIRC(T, P)$ denote the set of preferred models of T with respect to \leq^P .

3 From Argumentation Frameworks to Circumscription

An *abstract argumentation framework* (AF) is a directed graph G whose nodes $arg(G)$ are arguments, and whose arcs $att(G)$ represent an attack relation. An *extension* E is a set of arguments. The *range* of E in G is $E_G^+ := E \cup \{x \mid \exists yx \in att(G) \text{ with } y \in E\}$. In the following, the semantics of ICCMA'17 are characterized by means of circumscription.

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 E , and an atom r_x is possibly introduced to enforce that x belongs to the range E_G^+ :

$$attacked(G) := \left\{ a_x \leftrightarrow \bigvee_{yx \in att(G)} y \mid x \in arg(G) \right\} \quad (1)$$

$$range(G) := \left\{ r_x \rightarrow x \vee \bigvee_{yx \in att(G)} y \mid x \in arg(G) \right\} \quad (2)$$

The following set of formulas characterize semantics not based on preferences:

$$conflict-free(G) := \{\neg \perp\} \cup \{\neg x \vee \neg y \mid xy \in att(G)\} \quad (3)$$

$$admissible(G) := conflict-free(G) \cup attacked(G) \cup \{x \rightarrow a_y \mid yx \in att(G)\} \quad (4)$$

$$complete(G) := admissible(G) \cup \left\{ \left(\bigwedge_{yx \in att(G)} a_y \right) \rightarrow x \mid x \in arg(G) \right\} \quad (5)$$

$$stable(G) := complete(G) \cup range(G) \cup \{r_x \mid x \in arg(G)\} \quad (6)$$

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.

The ideal semantic is defined as follows (Proposition 3.6 by [5]): Let X be the set of admissible extensions of G that are not attacked by any admissible extensions, that is, $X := \{E \in models(admissible(G)) \mid \nexists E' \in models(admissible(G)) \text{ such that } yx \in att(G), x \in E, y \in E'\}$. E is the ideal extension of G if $E \in X$, and there is no $E' \in X$ such that $E' \supseteq E$.

All semantics of ICCMA'19 are characterized in circumscription as follows:

$$co(G) := CIRC(complete(G), \emptyset) \quad (7)$$

$$st(G) := CIRC(stable(G), \emptyset) \quad (8)$$

$$gr(G) := CIRC(complete(G), arg(G)) \quad (9)$$

$$pr(G) := CIRC(complete(G), \overline{arg(G)}) \quad (10)$$

$$sst(G) := CIRC(complete(G) \cup range(G), \{\neg r_x \mid x \in arg(G)\}) \quad (11)$$

$$stg(G) := CIRC(conflict-free(G) \cup range(G), \{\neg r_x \mid x \in arg(G)\}) \quad (12)$$

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

where in (13) U is the union of all admissible extensions of G , and Y is $U \setminus \{x \mid \exists yx \in att(G), y \in U\}$.

4 Implementation

Abstract argumentation frameworks can be encoded in trivial graph format (TGF) as well as in aspartix format (APX). The following data 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 $\{y \mid yx \in att(G)\}$. Within these data structures, theories (7)–(13) are constructed in amortized linear time. Single extension computation and extension enumeration is then demanded to the underlying circumscription solver [6].

The union U of all admissible extensions is computed by iteratively asking to CIRCUMSCRIPTINO to compute an admissible extension that maximize the accepted arguments not already in U , so to expand U as much as possible at each iteration.

For complete, stable, and preferred extensions, credulous acceptance is addressed by checking consistency of the theory extended with the query argument. Similarly, skeptical acceptance is addressed by adding the complement of the query argument for complete, and stable extensions. Grounded and ideal extensions are unique, and therefore credulous acceptance is 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 by a recent algorithm for query answering in circumscription [7]. In a nutshell, given a query atom q and a circumscribed theory, the computational problem amounts to search for a model of the circumscribed theory that contains the query atom. The algorithm implemented in CIRCUMSCRIPTINO searches for a classical model of the theory that contains the query atom, and checks that no more preferred model not containing the query atom exists. In this way, queries are possibly answered without computing any optimal model.

Counting of extensions is implemented by enumeration, with no particular optimization.

References

- [1] John McCarthy. Circumscription - A form of non-monotonic reasoning. *Artif. Intell.*, 13(1-2):27–39, 1980.
- [2] Mario Alviano. The pyglaf argumentation reasoner. In Ricardo Rocha, Tran Cao Son, Christopher Mears, and Neda Saeedloei, editors, *TC of ICLP 2017*, volume 58 of *OASICS*, pages 2:1–2:3. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.
- [3] Gilles Audemard and Laurent Simon. Predicting learnt clauses quality in modern SAT solvers. In Craig Boutilier, editor, *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, pages 399–404, 2009.
- [4] Mario Alviano. Ingredients of the argumentation reasoner pyglaf: Python, circumscription, and glucose to taste. In Marco Maratea and Ivan Serina, editors, *RCRA 2017*, volume 2011 of *CEUR Workshop Proceedings*, pages 1–16. CEUR-WS.org, 2017.
- [5] Martin Caminada. A labelling approach for ideal and stage semantics. *Argument & Computation*, 2(1):1–21, 2011.
- [6] Mario Alviano. Model enumeration in propositional circumscription via unsatisfiable core analysis. *TPLP*, 17(5-6):708–725, 2017.
- [7] Mario Alviano. Query answering in propositional circumscription. In Jérôme Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden.*, pages 1669–1675. ijcai.org, 2018.