The pyglaf argumentation reasoner

Mario Alviano

Department of Mathematics and Computer Science, University of Calabria, Italy
alviano@mat.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 [7] 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 [3] is implemented in Python and uses CIRCUMSCRIPTINO (`http://alviano.com/software/circumscriptino/`), a circumscription solver extending the SAT solver GLUCOSE [5]. Linear reductions are used for all semantics [1]. 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 Docker container is `malvi/pyglaf`. All problems from ICCMA'17 are supported.

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 $\{\overline{\ell} \mid \ell \in L\}$. *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 $atoms(T)$. 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. 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 $I \cap P \subseteq J \cap P$.

$I \in \mathit{models}(T)$ is a *preferred model* of T with respect to $\leq^P$ if there is no $J \in \mathit{models}(T)$ such that $I \not\leq^P J$ and $J \leq^P I$. Let $\mathit{CIRC}(T,P)$ 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 $\mathit{arg}(G)$ are arguments, and whose arcs $\mathit{att}(G)$ represent an attack relation. An *extension* E is a set of arguments. The *range* of E in G is $E^+_G := E \cup \{x \mid \exists\, yx \in \mathit{att}(G) \text{ with } y \in E\}$. 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:

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

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

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

$$\mathit{conflict\text{-}free}(G) := \{\neg\bot\} \cup \{\neg x \vee \neg y \mid xy \in \mathit{att}(G)\} \tag{3}$$

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

$$\mathit{complete}(G) := \mathit{admissible}(G) \cup \left\{ \left(\bigwedge_{yx \in \mathit{att}(G)} a_y \right) \rightarrow x \;\middle|\; x \in \mathit{arg}(G) \right\} \tag{5}$$

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

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 [6]): Let X be the set of admissible extensions of G that are not attacked by any admissible extensions, that is, $X := \{E \in \mathit{models}(\mathit{admissible}(G)) \mid \nexists E' \in \mathit{models}(\mathit{admissible}(G)) \text{ such that } yx \in \mathit{att}(G), x \in E, y \in E'\}$. E is the ideal extension of G if $E \in X$, and there is no $E' \in X$ such that $E' \supsetneq E$.

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

$$\mathit{co}(G) := \mathit{CIRC}(\mathit{complete}(G), \varnothing) \tag{7}$$

$$\mathit{st}(G) := \mathit{CIRC}(\mathit{stable}(G), \varnothing) \tag{8}$$

$$\mathit{gr}(G) := \mathit{CIRC}(\mathit{complete}(G), \mathit{arg}(G)) \tag{9}$$

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

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

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

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

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

3.1 Dynamic Track

An instance of the dynamic track can be represented by a triple (G, f, n), where G is a graph, $f : att(G) \to 2^{[0..n]}$ is a function, and $n \geq 0$. Intuitively, for all $i \in [0..n]$, the i-th graph has nodes $arg(G)$ and arcs $\{xy \in att(G) \mid i \in f(xy)\}$. In order to maintain the translation to circumscription compact, additional atoms are introduced: xy to represent that arc xy is present in the processed graph; e_{xy} to represent that the attack xy is enabled, that is, xy is present and x belongs to the computed extension; d_{xy} to represent that the attack xy is disabled, that is, either xy is not present or x is attacked by some argument in the computed extension.

Formulas (1)–(6) are replaced by the following formulas:

$$attacked^D(G) := \{e_{xy} \leftrightarrow xy \wedge x \mid xy \in att(G)\} \cup \left\{ a_x \leftrightarrow \bigvee_{yx \in att(G)} e_{yx} \;\middle|\; x \in arg(G) \right\} \tag{14}$$

$$range^D(G) := \left\{ r_x \to x \vee \bigvee_{yx \in att(G)} e_{yx} \;\middle|\; x \in arg(G) \right\} \tag{15}$$

$$c\text{-}free^D(G) := \{\neg\bot\} \cup \{xy \to \neg x \vee \neg y \mid xy \in att(G)\} \tag{16}$$

$$admissible^D(G) := c\text{-}free^D(G) \cup attacked^D(G) \cup \{yx \wedge x \to a_y \mid yx \in att(G)\} \tag{17}$$

$$\begin{aligned} complete^D(G) := {} & admissible^D(G) \cup \{d_{xy} \leftrightarrow \neg xy \vee a_x \mid xy \in att(G)\} \\ & \cup \left\{ \left(\bigwedge_{yx \in att(G)} d_{yx} \right) \to x \;\middle|\; x \in arg(G) \right\} \end{aligned} \tag{18}$$

$$stable^D(G) := complete^D(G) \cup range^D(G) \cup \{r_x \mid x \in arg(G)\} \tag{19}$$

Accordingly, theories (7)–(10) are replaced by the following theories:

$$co(G,f,i) := CIRC(complete^D(G) \cup \{xy \in att(G) \mid i \in f(xy)\}, \emptyset) \tag{20}$$

$$st(G,f,i) := CIRC(stable^D(G) \cup \{xy \in att(G) \mid i \in f(xy)\}, \emptyset) \tag{21}$$

$$gr(G,f,i) := CIRC(complete^D(G) \cup \{xy \in att(G) \mid i \in f(xy)\}, arg(G)) \tag{22}$$

$$pr(G,f,i) := CIRC(complete^D(G) \cup \{xy \in att(G) \mid i \in f(xy)\}, \overline{arg(G)}) \tag{23}$$

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 [2].

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 [4]. In a nutshell, given a query atom q and a circuscribed 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.

Concerning the Dynamic Track, the theories from Section 3.1 are optimized by avoiding additional atoms for arcs that are present in all graphs of the instance in input. Moreover, CIRCUMSCRIPTINO is kept online for the full computation, and instructed to add clauses and solve a computational problem with a given set of assumptions. In this way, each subsequent graph is processed incrementally, taking advantage of the knowledge discovered by the solver while processing the previous graphs. As a further optimization, when an arc xy is added at step i and not removed in any subsequent step, the unit clause xy is added to CIRCUMSCRIPTINO before processing the i-th graph. Analogously, when xy is removed at step i and not added in any subsequent step, $\neg xy$ is added before processing the i-th graph.

References

[1] 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.

[2] Mario Alviano. Model enumeration in propositional circumscription via unsatisfiable core analysis. *TPLP*, 17(5-6):708–725, 2017.

[3] 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.

[4] 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.

[5] 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.

[6] Martin Caminada. A labelling approach for ideal and stage semantics. *Argument & Computation*, 2(1):1–21, 2011.

[7] John McCarthy. Circumscription - A form of non-monotonic reasoning. *Artif. Intell.*, 13(1-2):27–39, 1980.