Extension Enforcement under Grounded Semantics in Abstract Argumentation

Andreas Niskanen  
University of Helsinki, Finland

Johannes P. Wallner  
TU Wien, Austria

Matti Järvisalo  
University of Helsinki, Finland

Abstract

The study of dynamics in abstract argumentation gives rise to optimization problems that are NP-hard also under the grounded semantics, in contrast to argument acceptance problems over argumentation frameworks (AF). Developing efficient systems for AF reasoning under grounded semantics has received less attention compared to other central AF semantics under which acceptance is NP-hard. In particular, grounded semantics is not currently supported by recent systems for extension enforcement, despite (or due to) its non-triviality. In this work, we propose and empirically evaluate three first approaches to enforcement under grounded semantics. While each of the approaches is based on employing constraint optimization solvers, we show empirically that there are significant differences in the scalability of the approaches.

1 Introduction

Argumentation is a vibrant area of modern KR and AI research. Abstract argumentation provides via argumentation frameworks (AFs) (Dung 1995) one of the central knowledge representation and reasoning formalisms for the study of computational aspects of argumentation. In contrast to the case of various other central semantics in abstract argumentation (Baroni, Caminada, and Giacomin 2011), the well-studied acceptance problems over AFs are polynomial-time decidable under the grounded semantics (Dung 1995), one of the central AF semantics. Therby developing efficient systems for AF reasoning under the grounded semantics has received less attention in comparison to the other important AF semantics under which deciding acceptance is NP-hard (Dunne and Wooldridge 2009; Dunne and Bench-Capon 2002; Dimopoulos and Torres 1996). However, the recent line of research on dynamics in abstract argumentation (Baumann and Brewka 2010; Cayrol, de Saint-Cyr, and Lagasquie-Schiex 2010; Baumann 2012; Baumann and Brewka 2015; Coste-Marquis et al. 2015; Diller et al. 2015; de Saint-Cyr et al. 2016)—motivated by the fact that argumentation is intrinsically a dynamic process—gives rise to interesting optimization problems which have proven to be NP-hard also under the grounded semantics. Specifically, extension enforcement (Baumann and Brewka 2010) under the grounded semantics, where the task is to make a given set of arguments the grounded extension of a given AF by changing the attack relation of the AF, has been recently shown to be NP-hard (Wallner, Niskanen, and Järvisalo 2017) under the distance measure of minimizing the number of changes to the attack structure. In terms of application scenarios, enforcement as a formal problem situates e.g. within the field of persuasion, where a certain goal is to be achieved by bringing new information to a debate (de Saint-Cyr et al. 2016).

First algorithms with implemented systems for extension enforcement (Coste-Marquis et al. 2015; Niskanen, Wallner, and Järvisalo 2016), covering various other semantics have been proposed. However, no systems for extension enforcement under grounded semantics are currently available, despite—or due to—the fact that supporting this semantics has turned out to be computationally non-trivial. In this work, we bridge this gap by proposing and empirically evaluating several algorithmic approaches to enforcement under grounded semantics. The approaches are based on declarative techniques, each making use of distinct declarative aspects. In particular, we propose (i) to extend an earlier-proposed answer set programming (ASP) encoding (Egly, Gaggl, and Woltran 2010) for the grounded extension to the realm of extension enforcement; (ii) a new direct maximum satisfiability (MaxSAT) encoding for the problem; as well as (iii) a MaxSAT-based counterexample-guided abstraction refinement (CEGAR) (Clarke et al. 2003; Clarke, Gupta, and Strichman 2004) procedure that makes use of earlier theoretical observations (Boella, Kaci, and van der Torre 2009a; 2009b; Rienstra, Sakama, and van der Torre 2015) for achieving a non-trivial, tighter refinement step. We show empirically that there are significant differences in the scalability of the approaches when applied on extension enforcement instances arising from benchmarks from the latest ICCMA AF solver competition (Gaggl et al. 2016), with the two new MaxSAT-based approaches dominating the ASP encoding.

2 Argumentation Frameworks

We recall argumentation frameworks (Dung 1995) and their semantics (Baroni, Caminada, and Giacomin 2011).

Definition 1. An argumentation framework (AF) is a pair $F = (A, R)$ where $A$ is a finite set of arguments and $R \subseteq A \times A$...
is the attack relation. The pair \((a, b) \in R\) means that \(a\) attacks \(b\). An argument \(a \in A\) is defined \((\text{in } F)\) by a set \(S \subseteq A\) if, for each \(b \in A\) s.t. \((b, a) \in R\), there is a \(c \in S\) s.t. \((c, b) \in R\).

Semantics for AFs are defined through a function \(\sigma\) which assigns to each AF \(F = (A, R)\) a set \(\sigma(F) \subseteq 2^A\) of extensions. Here we consider the functions \(\text{adm, com, and } \text{grd}\), which stand for admissible, complete, and grounded extensions, respectively.

Definition 2. Given an AF \(F = (A, R)\), the characteristic function \(\mathcal{T}_F : 2^A \to 2^A\) of \(F\) is \(\mathcal{T}_F(S) = \{x \in A \mid x \text{ is defended by } S\}\). A set \(S \subseteq A\) is conflict-free \((S \in \text{cf}(F))\), if there are no \(a, b \in S\), such that \((a, b) \in R\). For a conflict-free set \(S \in \text{cf}(F)\), it holds that \(S \in \text{adm}(F)\) iff \(S \subseteq \mathcal{T}_F(S)\); \(S \in \text{com}(F)\) iff \(S = \mathcal{T}_F(S)\); and \(S \in \text{grd}(F)\) iff \(S\) is the \(\subseteq\)-least fixed-point of \(\mathcal{T}_F\).

For any AF \(F\) we have \(\text{cf}(F) \supseteq \text{adm}(F) \supseteq \text{com}(F) \supseteq \text{grd}(F)\). A \(\sigma\)-extension is an extension under a semantics \(\sigma \in \{\text{adm, com, grd}\}\).

Example 1. Let \(F = (A, R)\) be an AF with \(A = \{a, b, c, d, e\}\) and \(R = \{(a, b), (b, c), (d, e), (d, c), (d, e), (e, e)\}\). We have \(\text{adm}(F) = \{\emptyset, \{a\}, \{a, c\}, \{a, d\}, \{d\}\}\), \(\text{com}(F) = \{\{a\}, \{a, c\}, \{a, d\}\}\), and \(\text{grd}(F) = \{\{a\}\}\).

3 Extension Enforcement

We focus on argument-fixed extension enforcement (Coste-Marquis et al. 2015; Wallner, Niskanen, and Järvisalo 2017) under grounded semantics. In strict extension enforcement, given an AF \(F = (A, R)\), a subset of its arguments \(T \subseteq A\), and an AF’s semantics \(\sigma\), the goal is to modify \(R\) such that \(T\) becomes a \(\sigma\)-extension. In non-strict extension enforcement, it suffices that \(T\) is a subset of a \(\sigma\)-extension. We denote by \(\text{enf}(s, F, T, \sigma) = \{R' \mid F' = (A, R'), T \in \sigma(F')\}\), the set of attack structures that strictly \((s)\) enforce \(T\) under \(\sigma\) for an AF \(F\), and for non-strict \((ns)\) enforcement by \(\text{enf}(ns, F, T, \sigma) = \{R' \mid F' = (A, R'), \exists T' \in \sigma(F') : T' \supseteq T\}\).

The Hamming distance between two attack structures is \(|R \Delta R'| = |R \setminus R'| + |R' \setminus R|\), i.e., the number of changes between \(R\) and \(R'\). We consider extension enforcement as the optimization problem of minimizing the Hamming distance between the original and modified attack structures.

Grounded Extension Enforcement \((M \in \{s, ns\})\)

Input: \(AF = (A, R), T \subseteq A\)
Task: Find an \(AF F' = (A, R')\) with \(R' \in \text{arg min}_{R' \in \text{enf}(M, F, T, \sigma)} |R \Delta R'|\).

The corresponding decision problem, asking whether there is a solution attack structure \(R'\) with \(|R \Delta R'| \leq k\) for a given integer \(k\), is NP-complete (Wallner, Niskanen, and Järvisalo 2017). Note that \(T = \emptyset\) is trivial for both the strict and non-strict case. Further, for any \(R' \in \text{enf}(M, F, T, \text{grd})\), if \(T \neq \emptyset\), there is an unattacked argument in \(R'\) (the grounded extension is empty iff all arguments are attacked).

<table>
<thead>
<tr>
<th>Listing 1: Module (\pi_{\text{enf}})</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\text{mAtt}(X, Y) \leftarrow \text{not } \text{n_mAtt}(X, Y), \text{arg}(X), \text{arg}(Y)).</td>
</tr>
<tr>
<td>(\text{n_mAtt}(X, Y) \leftarrow \text{not } \text{mAtt}(X, Y), \text{arg}(X), \text{arg}(Y)).</td>
</tr>
<tr>
<td>(\text{enf}(X), \text{not } \text{in}(X)).</td>
</tr>
<tr>
<td>(\text{in}(X), \text{not } \text{enf}(X), \text{strict}).</td>
</tr>
<tr>
<td>(\text{n_root}(X) \leftarrow \text{arg}(X), \text{arg}(Y), \text{mAtt}(Y, X)).</td>
</tr>
<tr>
<td>(\text{root}(X) \leftarrow \text{arg}(X), \text{not } \text{n_root}(X)).</td>
</tr>
<tr>
<td>(\text{root_exists} \leftarrow \text{enf}(X), \text{root}(X), \text{strict}).</td>
</tr>
<tr>
<td>(\text{root_exists} \leftarrow \text{arg}(X), \text{root}(X), \text{not strict}).</td>
</tr>
<tr>
<td>(\leftarrow \text{not root_exists}).</td>
</tr>
<tr>
<td>(\leftarrow \text{mAtt}(X, Y), \text{not } \text{att}(X, Y)). [1, X, Y].</td>
</tr>
<tr>
<td>(\leftarrow \text{not } \text{mAtt}(X, Y), \text{att}(X, Y)). [1, X, Y].</td>
</tr>
</tbody>
</table>

4 Grounded Enforcement via ASP

We give an ASP encoding for grounded enforcement by extending an encoding for the simpler problem of computing the grounded extension (Egly, Gaggl, and Woltran 2010). Here we only briefly recall the ASP language features we make use of; for more details, see (Gebser et al. 2012). We use programs with weak constraints of the form \("\prec b_1, \ldots, b_n, w.t_1, \ldots, t_m\"\), with each \(b_i\) an atom, each \(t_j\) a term, and \(w\) an integer; intuitively, if an answer set violates this constraint, then each tuple \((t_1, \ldots, t_m)\) contributes cost \(w\). An optimal answer set is one with minimum cost.

To encode enforcement under grounded semantics, we conjoin Egly, Gaggl, and Woltran’s (2010) encoding for the grounded extension, \(\pi_{\text{ground}}\), with the encoding \(\pi_{\text{enf}}\) given in Listing 1. The encoding computes the grounded extension for an attack structure in predicate \(\text{in}\) which depends now on \(\text{mAtt}\), the modified attack structure. Encoding \(\pi_{\text{enf}}\) guesses a modified attack structure and constrains the answer sets so that the enforced arguments \((\text{enf})\) are a subset of \(\text{in}\) (and under \(\text{strict}\) enforcement to coincide with \(\text{in}\)). The next four rules compute \(\text{roots}\) (unattacked arguments) of the modified attack structure, and check whether there is such a root. While these rules are redundant (every grounded extension has an unattacked argument), we observed that solvers benefit from them. Finally, the weak constraints ensure that unit weight is given to each changed attack.

5 Grounded Enforcement via MaxSAT

As an alternative to the ASP approach, we propose a direct MaxSAT encoding for grounded extension enforcement. Recall that a propositional clause is a disjunction of literals, i.e., positive and negative Boolean variables. Satisfaction of a clause by a truth assignment \(\tau\) is defined in the standard way. An instance \(\varphi = (\varphi_0, \varphi_1)\) of the \((\text{partial})\) MaxSAT problem consists of a set \(\varphi_0\) of \text{hard} clauses and a set \(\varphi_1\) of \text{soft} clauses. An assignment \(\tau\) that satisfies every clause in \(\varphi_0\) is a \text{solution} to \(\varphi\). The \text{cost} of a solution \(\tau\) to \(\varphi\) is \(C(\varphi, \tau) = \sum_{\varphi \in \varphi_0} (1 - \tau(\varphi))\), i.e., the number of soft clauses not satisfied by \(\tau\). A solution \(\tau\) is \text{optimal} if \(C(\varphi, \tau) \leq C(\varphi, \tau')\) holds for any solution \(\tau'\) to \(\varphi\). The MaxSAT problem asks to find an optimal solution to a given \(\varphi\).

Let \(F = (A, R)\) be the AF of an enforcement instance. For each \(a, b \in A\), define a Boolean variable \(r_{a,b}\) with the
interpretation \( \tau(r_{a,b}) = 1 \) iff the attack \((a, b)\) is included in the attack structure \( R' \) of the solution AF \( F' = (A, R') \). We minimize the Hamming distance between the original and solution attack structures via the soft clauses \( \phi_f(F) = \bigwedge_{a,b \in A} r_{a,b}' \), where \( r_{a,b}' \) is \( r_{a,b} \) if \((a, b) \in R\), and else \( \neg r_{a,b} \).

The computation of the grounded extension of \( F' \) is encoded via hard clauses. The idea is to map each argument \( a \) in the grounded extension of \( F' \) to a level number \( \text{Level}(a) \) corresponding to the number of applications of \( R_{F'}^n \) to \( \emptyset \) after which the argument is included, that is, \( \text{Level}(a) = \min\{ n \in \mathbb{Z}_+ \mid a \in R_{F'}^n(\emptyset) \} \). To encode this, we define Boolean variables \( l_a^n \) for each \( a \in A \) and \( n = 1, \ldots, \lceil |A|/2 \rceil \) with the interpretation \( \tau(l_a^n) = 1 \) iff \( \text{Level}(a) \leq n \), and impose the following constraints.

\[
\forall a \in A: \phi_a = \left( l_a^1 \leftrightarrow \bigwedge_{b \in A} \neg r_{b,a} \right)
\]

\[
\forall a \in A, n \geq 2: \phi_a^n = \left( l_a^n \leftrightarrow \bigwedge_{b \in A} \left( r_{b,a} \lor \bigwedge_{c \in A} (r_{c,b} \land l_{c,b}^{n-1}) \right) \right)
\]

\[
\forall a \in A: \phi_{prop} = \bigvee_{n=2}^{\lceil |A|/2 \rceil} \left( l_a^{n-1} \rightarrow l_a^n \right)
\]

\[
\phi_{first} = \bigvee_{a \in A} l_a^1
\]

Here \( \phi_a^1 \) encodes that for argument \( a \), \( \text{Level}(a) = 1 \) if and only if \( a \) is unattacked. Likewise, \( \phi_a^n \) for \( n \geq 2 \) encodes that \( \text{Level}(a) \leq n \) if and only if \( a \) is defended by the arguments on levels \( 1, \ldots, n-1 \). The formulas \( \phi_{prop}^a \) and \( \phi_{first} \) are redundant, but help guiding the MaxSAT solver in practice. In analogy with \texttt{root_exists} of the ASP encoding, \( \phi_{first} \) encodes that every grounded extension contains an unattacked argument.

Let \( T \subseteq A \) be a set to be enforced under grounded. For non-strict enforcement, the unit clauses \( \bigwedge_{a \in T} l_a^{\lceil |A|/2 \rceil} \) make \( T \) a subset of the grounded extension. For strict enforcement, we need in addition the clauses \( \bigwedge_{a \in T} \neg l_a^{\lceil |A|/2 \rceil} \) to make sure that no other arguments are part of the grounded extension. Furthermore, for \( a, b \in T \), the unit clause \( \neg r_{a,b} \) encodes the fact that \( r_{a,b} \) must be false. For strict enforcement \( l_a^n \) is false for all \( n \) and for all \( a \not\in T \); thereby we can replace the bound \( \lceil |A|/2 \rceil \) by \( |T| \).

6 Grounded Enforcement via CEGAR

Instead of a direct MaxSAT encoding, another way to solve extension enforcement under grounded can be derived based on the fact that the grounded extension is one of the complete (and admissible) extensions. This gives rise to a counterexample-guided abstraction refinement (CEGAR) procedure for grounded extension enforcement. The approach iteratively optimally enforces \( T \) under complete semantics using a direct MaxSAT encoding, as an abstraction of grounded enforcement. If \( T \) is the grounded extension of the AF resulting from the enforcement, the AF is an optimal solution to grounded extension enforcement. If \( T \) is not the grounded extension of the obtained AF, a constraint ruling out the AF is added to the MaxSAT encoding as a refinement to the abstraction, and a MaxSAT solver is again invoked. For obtaining a tighter and non-trivial abstraction refinement and ruling out more non-solutions from further consideration after each MaxSAT solver call, we make use of theoretical observations from (Boella, Kaci, and van der Torre 2009a; 2009b; Rienstra, Sakama, and van der Torre 2015) on what type of changes to an attack structure can and cannot influence the grounded extension of an AF.

In more detail, we propose the CEGAR algorithm presented as Algorithm 1, adapting (Wallner, Niskanen, and Järvisalo 2017). For strict enforcement, we choose complete as the semantics for the abstraction, and admissible for non-strict (since complete and admissible coincide for non-strict). In both cases we make use of the same soft clauses \( \phi_f \), as in the previous section. Strict enforcement of \( T \) under complete semantics is encoded via the formula

\[
\text{EXT}(s,F,T,\text{com}) = \bigwedge_{a,b \in T} \neg r_{a,b} \land \bigwedge_{a \in T} \bigwedge_{b \in A \setminus T} \left( r_{b,a} \lor \bigwedge_{c \in A} (r_{c,b} \land l_{c,b}^1) \right) \land \bigwedge_{a,b \in T} \bigwedge_{c \in T} (r_{b,a} \land \neg r_{c,b}).
\]

For non-strict enforcement under admissible, we also define for each \( a \in A \) a Boolean variable \( x_a \) with the interpretation \( \tau(x_a) = 1 \) iff argument \( a \) is included in the extension of the solution AF. The hard clauses

\[
\text{EXT}(ns,F,T,\text{adm}) = \bigwedge_{a \in T} x_a \land \bigwedge_{a,b \in A} \left( r_{a,b} \rightarrow (\neg x_a \lor x_b) \right) \land \bigwedge_{a,b \in T} \bigwedge_{c \in A} \left( x_a \land r_{b,a} \land (x_c \lor r_{c,b}) \right)
\]

are used to solve non-strict enforcement under admissible.

In addition, we augment the abstraction by including clauses \( \text{ROOT}(s,F,T) = \bigwedge_{a \in T} \phi_f^a \lor \bigwedge_{a \in T} \phi_f^1 \) (analogous to encoding roots in the ASP and MaxSAT encodings) for strict enforcement, stating that \( T \) should contain at least one unattacked argument, which together with \( \text{EXT}(s,F,T,\text{com}) \) implies that all unattacked arguments will be contained in \( T \). For non-strict, we add clauses \( \text{ROOT}(ns,F,T) = \bigwedge_{a \in A \setminus T} \phi_f^a \land \bigwedge_{a \in T} \bigwedge_{b \in A \setminus T} (l_{b,a} \rightarrow x_a) \) expressing that all unattacked arguments should be contained in an admissible superset.

The abstraction is solved in the main loop using MaxSAT, obtaining the attack relation \( \text{CONSTRUCTAF}(\tau) = \{(a,b) \in A \times A \mid \tau(r_{a,b}) = 1\} \). We then check if \( T \) is (for strict) or is

\begin{algorithm}
\begin{algorithmic}
\State 1: if \( M = s \) then \( \sigma \leftarrow \text{com else} \sigma \leftarrow \text{adm} \)
\State 2: \( \phi_f \leftarrow \text{EXT}(M,F,T,\sigma) \land \text{ROOT}(M,F,T) \)
\While {true}
\State 4: \((c, \tau) \leftarrow \text{MAXSAT}(\phi_f, \phi_f(F))\)
\State 5: \( F' \leftarrow (A,\text{CONSTRUCTAF}(\tau))\)
\If {\( M = s \) and \( T = \text{grd}(F') \)} return \( F' \)
\If {\( M = ns \) and \( T \subseteq \text{grd}(F') \)} return \( F' \)
\State 8: \( \phi_f \leftarrow \phi_f \land \text{REFINE}(F') \)
\EndWhile
\end{algorithmic}
\end{algorithm}
Figure 1: Comparison of approaches for strict (left) and non-strict (right) enforcement under grounded.

7 Experiments
We evaluate the three approaches to grounded extension refinement in (for non-strict) the grounded extension of the AF \( F' = (A, R') \). If the check fails, we proceed by refining the abstraction. At this point we make use of theoretical observations from (Boella, Kaci, and van der Torre 2009a; 2009b; Rienstra, Sakama, and van der Torre 2015). Specifically, in the labelling-based setting (Baroni, Caminada, and Giacomin 2011), under grounded semantics an argument \( a \in \text{in}_{\text{grd}}(F) \) if \( a \in \text{grd}(F) \), \( a \in \text{out}_{\text{grd}}(F) \) if there is \( a b \in \text{grd}(F) \) such that \( (b, a) \in R \), and \( a \in \text{undec}_{\text{grd}}(F) \) otherwise. Let

\[
\text{ATTADD}(F) = \{(a, b) \notin R \mid (a, b) \in (\text{in}_{\text{grd}}(F) \times \text{in}_{\text{grd}}(F)) \cup (\text{out}_{\text{grd}}(F) \times \text{in}_{\text{grd}}(F)) \cup (\text{undec}_{\text{grd}}(F) \times \text{in}_{\text{grd}}(F))\}
\]

be the attacks that may alter the grounded extension of \( (A, R) \) when added to \( R \); e.g., if both \( a \) and \( b \) are part of the grounded extension, adding attack \( (a, b) \) makes this set conflicting. Further, let\n
\[
\text{ATTREM}(F) = \{(a, b) \in R \mid (a, b) \in (\text{in}_{\text{grd}}(F) \times \text{out}_{\text{grd}}(F)) \cup (\text{undec}_{\text{grd}}(F) \times \text{undec}_{\text{grd}}(F))\}
\]

be the attacks that can change the grounded extension of \( (A, R) \) when removed from \( R \). This yields a non-trivial abstraction refinement \( \text{REFINE}(F) = \forall (a, b) \in \text{ATTADD}(F)(a, b) \lor \forall (a, b) \in \text{ATTREM}(F) \neg (a, b) \), stating that in the subsequent iterations, we should add (respectively remove) an attack not present (respectively present) in the current solution in such a way that we obtain a different grounded extension.

8 Conclusions
We proposed the first declarative approaches to the NP-hard extension enforcement problem under grounded semantics. Each of the approaches also allows for integrating structural constraints e.g. on which particular attacks and non-attacks may be subjected to change. The alternative MaxSAT and
CEGAR approaches show noticeably better scaling than the ASP encoding based on an earlier ASP encoding for computing the grounded extension. The approaches may be of interest for potential further optimization problems related to argumentation dynamics which turn out to be NP-hard under the grounded semantics.

Acknowledgements

Work supported in part by Academy of Finland (grants 276412 and 312662), DoCS Doctoral Programme in Computer Science and Research Funds of the University of Helsinki, and Austrian Science Fund (FWF): P30168-N31.

References


