CEGARTIX: A SAT-Based Argumentation System*

Wolfgang Dvořák[1], Matti Järvisalo[2], Johannes Peter Wallner[1], and Stefan Woltran[1]

[1] Institute of Information Systems, Technische Universität Wien, Austria
[2] HIIT & Department of Computer Science, University of Helsinki, Finland

Abstract. CEGARTIX is a system for solving decision problems beyond NP in the domain of abstract argumentation. The system implements multiple counter-example guided abstraction refinement (CEGAR) algorithms in which Boolean satisfiability (SAT) solvers play a central role as the core NP-solvers. The performance of CEGARTIX is comparable and in cases superior to previously developed state-of-the-art systems for second-level argumentation problems. We briefly overview as an example one of the algorithms CEGARTIX implements and present experimental results on the performance of the system using three different modern SAT solvers as the core NP solvers.

1 Introduction

Argumentation is present in various real-world scenarios, including mediation processes, law research, analysis of social interactions e.g. on the Internet. Computational argumentation has been identified as an important contemporary area of research that deals with computationally hard reasoning problems related to such scenarios. The main formalism for presenting computational argumentation problems is provided by abstract argumentation frameworks (AFs) [3]. Developing decisions procedures for argumentation problems is challenging since important reasoning problems for AFs are complete for the second level of the polynomial hierarchy (i.e., Π_2^P- / Σ_2^P-complete).

In this paper we describe CEGARTIX, a generic system for solving Π_2^P- / Σ_2^P-complete decision problems in the domain of abstract argumentation, namely, *skeptical* and *credulous reasoning* under so-called *preferred* [4], *semi-stable*, and *stage* semantics [5]) Based on counter-example guided abstraction refinement (CEGAR) [1,2], CEGARTIX implements multiple *complexity-sensitive* algorithms [7] that rely on Boolean satisfiability (SAT) solvers as the core NP solvers. The algorithms CEGARTIX implements are based on our recent complexity analysis [7] of so-called *semantical* fragments of AF reasoning problems with full second-level complexity.

Monolithic SAT-encodings of second-level argumentation problems are deemed to be of exponential size. In contrast, our complexity-sensitive procedures provable avoid the exponential space requirements in the following sense: if the input instances falls into a predefined less intractable *base* class, it suffices to consider a small part of a monolithic encoding to decide the actual query.

* Work has been funded by the Vienna Science and Technology Fund (WWTF) through project ICT08-028 and by Academy of Finland (grants 132812 and 251170).

CEGARTIX outperforms current state-of-the-art systems which are based on monolithic encodings of the considered second-level argumentation problems [8,6]. First results on the applicability of a first version of CEGARTIX were presented in [7]. In this description, we present an extended evaluation of the current version 0.3 of CEGARTIX. As new features CEGARTIX 0.3 gives a choice between two state-of-the-art conflict-driven clause learning (CDCL) SAT solvers (Minisat and Clasp) used as libraries. Further, any complete SAT solver that adheres to the standard input-output specification of the SAT competitions can be used externally. In addition to a comparison with other state-of-the-art argumentation systems, we evaluate the effect of incremental employment of a SAT solver on the performance of CEGARTIX, as well as the effect of the choice of the SAT solver.

In addition to demonstrating that abstract argumentation is a novel and successful application for SAT solvers, we believe that developing SAT-based complexity-sensitive algorithms similar to the ones implemented within CEGARTIX could provide novel algorithms for other important decision and optimization problems beyond NP.

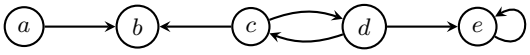
2 Background

Due to the page limit, here we focus on describing the algorithm CEGARTIX implements for the problem of *skeptical acceptance* for *preferred semantics*. More details for the other second-level problems CEGARTIX targets can be found in [7].

An *argumentation framework (AF)* [3] is a pair $F = (A, R)$ where A is a finite set of arguments and $R \subseteq A \times A$ is the attack relation. An argument $a \in A$ is *defended* (in F) by a set $S \subseteq A$ if for each $b \in A$, such that $(b, a) \in R$, there is a $c \in S$ with $(c, b) \in R$. A set $S \subseteq A$ is *conflict-free* (in F), denoted $S \in cf(F)$, iff there are no $a, b \in S$, such that $(a, b) \in R$. If $S \in cf(F)$ and each $a \in S$ is defended by S, then S is said to be *admissible* in F, denoted by $S \in adm(F)$. *Semantics* for AFs assign to each AF $F = (A, R)$ a set $\sigma(F) \subseteq 2^A$ of extensions. The following defines *complete* and respectively *preferred* extensions:

$$com(F) = \{S \in adm(F) \mid \text{ for each } a \in A \text{ defended by } S,\ a \in S\}$$
$$prf(F) = \{S \in adm(F) \mid \text{ there is no } T \in adm(F) \text{ with } T \supset S\}$$

For example, consider the AF $F = (A, R)$, with $A = \{a, b, c, d, e\}$ and $R = \{(a, b)$, (c, b), (c, d), (d, c), (d, e), $(e, e)\}$. The graph representation of F is



The admissible sets of F are $\emptyset$, $\{a\}$, $\{c\}$, $\{d\}$, $\{a, c\}$, $\{a, d\}$, $prf(F) = \{\{a, c\}, \{a, d\}\}$ and $com(F) = \{\{a\}, \{a, c\}, \{a, d\}\}$. In fact, $prf(F) \subseteq com(F)$ holds for any AF F. One can thus use complete extensions as candidates for preferred ones.

In the next section we describe as an example the algorithm within CEGARTIX for the Π_2^P-complete problem of skeptical acceptance for preferred semantics, Skept_{prf}: Given an AF $F = (A, R)$ and argument $a \in A$, is a contained in each $S \in prf(F)$?

3 CEGARTIX

CEGARTIX implements the CEGAR-style algorithms recently introduced in [7] for credulous and skeptical reasoning under preferred, semi-stable and stage semantics. In the following we present more details on the algorithms within CEGARTIX using as an example the algorithmically perhaps simplest case of skeptical reasoning under preferred semantics. As regards complexity-sensitivity, the algorithm is exploits the following result [7]. Let sol^k_{prf} the class of all AFs F such that $|prf(F)| \leq k$. Given a fixed k, the problem Skept_{prf} when restricted to AFs from the class sol^k_{prf}, can be solved in deterministic polynomial time using a bounded number of NP-oracle calls (i.e., it is in P^{NP}). Indeed, given that the AF given as input belongs to sol^k_{prf}, the algorithm is guaranteed to make only a polynomial number of SAT solver calls.

The main idea is to use the SAT-solver calls to check candidate AF extensions of the input instance. Candidate AF extensions for the preferred semantics can be identified via the simpler complete semantics. To this end, let us start with the following module for a given AF $F = (A, R)$ using variables $X = \{x_a \mid a \in A\}$ and $Y = \{y_a \mid a \in A\}$:

$$\varphi_{com}(F) = \bigwedge_{(a,b)\in R} (\neg x_a \vee \neg x_b) \wedge \bigwedge_{(b,a)\in R} (x_a \rightarrow \bigvee_{(c,b)\in R} x_c) \wedge$$
$$\bigwedge_{a\in A} (y_a \leftrightarrow x_a \vee \bigvee_{(b,a)\in R} x_b) \wedge \bigwedge_{a\in A} (\bigwedge_{(b,a)\in R} y_b) \rightarrow y_a$$

The first line declares the conditions for admissible sets following the definition: any admissible set must be (i) conflict free and (ii) each argument in the set must be defended by the set. The second line declares that (i) the value of auxiliary atoms y_a: y_a is true iff either x_a is true or some x_b is true where b attacks a in F, and that (ii) each argument a defended by the extension is contained in the extension. Models (i.e., full satisfying assignments) of $\varphi_{com}(F)$ characterize the complete extensions of F in the sense that x_a is true in a model I ($x_a \in I$) iff argument a is in the extension characterized by I.

In our algorithm the candidate AF extension checks will be solved (incrementally when possible) with a SAT solver. In addition to reducing the set of remaining candidate solutions, we also exploit the results of the SAT-oracle calls to strengthen the base formula $\varphi_{com}(F)$ as a form of no-good learning on the level of the propositional encoding. The structure of the algorithm for Skept_{prf}, testing an argument a for skeptical acceptance, implemented within CEGARTIX is the following:

Input: AF $F = (A, R)$, argument $a \in A$

1. **if** $\varphi_{com}(F) \wedge (\bigvee_{(b,a)\in R} x_b)$ is satisfiable, **then** *reject*
2. $\varphi \leftarrow \varphi_{com}(F) \wedge \neg x_a \wedge (\bigwedge_{(b,a)\in R} \neg x_b)$.
3. **while** (φ is satisfiable)
   (a) find model I of φ
   (b) **while** (there is a model I' of $\varphi_{com}(F) \wedge \bigwedge_{x\in I\cap X} x \wedge (\bigvee_{x\in X\setminus I} x) \wedge \neg x_a$)
   $I \leftarrow I'$
   (c) **if** $\varphi_{com}(F) \wedge \bigwedge_{x\in I\cap X} x \wedge (\bigvee_{x\in X\setminus I} x)$ is unsatisfiable **then** *reject*
   (d) **else** $\varphi \leftarrow \varphi \wedge (\bigvee_{x\in X\setminus I} x)$
4. *accept*

Overall, the procedure works as follows. Step 1 exploits the fact that a cannot be skeptically accepted in case there exists a complete extension containing an argument b attacking a (since then, also such a preferred extension exists). In Step 2, we initialize a formula φ (which will be refined in the algorithm) that provides a model I describing a complete extension that does not contain a. Moreover, we add the information that no argument b attacking a can be contained in such an extension. If no such I exists, we can conclude that all preferred extensions of the given AF contain a; in this case we do not enter the **while** loop and accept. Otherwise, we proceed with the loop in order to "enlarge" I as much as possible such that it remains complete but still does not contain a. If adding a to such a maximal set violates completeness, we have found a preferred extension not containing a, and thus we have to **reject** here. Otherwise, we adapt φ such that it will deliver complete extensions not being subsets of (the maximized) I. As before, if the new φ turns out to be unsatisfiable, we know that all possible remaining candidates for a preferred extension have to contain a and thus we can **accept**. Otherwise, we enter the loop and proceed as already outlined.

4 Performance of CEGARTIX

In the current version (v0.3) of CEGARTIX there is an option to choose from two CDCL SAT solvers (used as libraries) as the core solvers: Minisat (v2.2.0) and Clasp (v2.0.5). Minisat can be employed in an incremental mode, which allows us to avoid starting search from scratch when adding new learnt information to a current satisfiable working formula. Furthermore, a command line option is provided for alternatively using any external SAT solver that adheres to the standard input-output specification used in the SAT competitions. The input format of CEGARTIX follows the input format of the ASPARTIX [8] system: an argument a is declared by "`arg(`a`).`", and "`att(`a`,`b`).`" declares that argument a attacks argument b. The current version of CEGARTIX is available at

`http://www.dbai.tuwien.ac.at/research/project/argumentation/cegartix/`.

The URL also gives access to the benchmark generators used in our experiments.

Experiment Setup At the moment, argumentation systems are commonly benchmarked using instances based on different (structured) models of generating random graphs that are interpreted as the AFs. As benchmarks, we randomly generated AFs using two parameterized methods for generating the attack relation. The first generates *random* AFs and inserts for any pair of arguments (a, b) the attack from a to b with a given probability p. The other method generates AFs of an $n \times m$ *grid structure* . For the grid instances, we consider two different neighborhoods: one connecting arguments vertically and horizontally, and one that additionally connects the arguments diagonally. Such a connection is (i) a mutual attack with probability p, and (ii) an attack from a to b with probability $(p - 1)$. In each case, the probability values $p \in \{0.1, 0.2, 0.3, 0.4\}$ were used. We generated 10 random AFs for each pair of $(p, args)$, $args$ being the number of arguments. For the grid structure AFs we generated 11 AFs for each triple of $(p, args, m)$ and $m \in \{5, 10, 15\}$.

For the random instances two arguments were queried; for the grid instances three. The number of attacks scales linearly with the number of arguments for grid instances, and quadratically for random instances. We would like to stress that the generated AFs are by no means tailored to the fragments our approach is based on.

The experiments were executed under OpenSUSE with Intel Xeon processors (2.33 GHz) and 49 GB memory, using a timeout of 5 minutes for each individual run.

4.1 Comparison of CEGARTIX with State-of-the-Art Argumentation Systems

We compare CEGARTIX to a recent state-of-the-art argumentation reasoning system [6] that exploits advances in answer set programming (ASP) via the so-called `metasp` approach. This system is a further improvement of the ASP-based ASPARTIX system [8], and applies the native second-level disjunctive ASP solver claspD (v1.1.1) combined with the grounder gringo (v3.0.3). For this comparison, we employed Minisat as the SAT solver within CEGARTIX. The results of are shown in Figure 1. We observe that CEGARTIX clearly dominates the current state-of-the-art system ASPARTIX that employs the metasp approach both on the random instances (results presented for instances having up to 200 arguments) and the grid instances (results presented for up to 1000 arguments). As has also been shown earlier [8], until now metasp-based ASPARTIX has represented the state-of-the-art, dominating the earlier version of ASPARTIX that did not employ metasp [6]. Interestingly, the performance of CEGARTIX is especially pronounced on the more structured grid instances. For a comparison of the metasp-based approach and the earlier version of ASPARTIX on grid structure AFs see [6].

4.2 Effect of the Choice of SAT Solver within CEGARTIX

As already mentioned, the current implementation of CEGARTIX allows to choose from two state-of-the-art solvers as the core SAT solver (Minisat and Clasp), using the solvers as libraries. Minisat can be used either in an incremental fashion or non-incrementally. Furthermore, there is a command line option for employing an external SAT solver binary as the core solver.

We investigated how the choice of the core SAT solver effects the performance of CEGARTIX. In addition to Minisat and Clasp, we used March_nh (the latest, SAT Challenge 2012 version of March) as an external solver. The results are shown in Fig. 2 for both the random and grid instances. First, one can observe that March_nh is not a competitive choice as the core SAT solver.[3] On the random instances, we observe quite similar performance when employing Minisat (non-)incrementally; in other words, it appears that for these instances incrementality does not improve performance. Employing Clasp appears to yield slightly better scaling than employing Minisat. However, the situation is different on the grid instances. First, we observe that non-incremental Minisat clearly yields better performance than Clasp on these more structured instances. Furthermore, employing the incremental interface of Minisat give an additional improvement of a similar order over the non-incremental employment of Minisat.

[3] Furthermore, we excluded the following number of timeouts for March_nh on the grid instances:1 instance with 600 nodes, 2 with 700, 4 with 800, 1 with 900, 11 with 1000 nodes.

References

1. E.M. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith. Counterexample-guided abstraction refinement for symbolic model checking. *J. ACM*, **50**(5):752–794, 2003.
2. E.M. Clarke, A. Gupta, and O. Strichman. SAT-based counterexample-guided abstraction refinement. *IEEE T-CAD*, **23**(7):1113–1123, 2004.
3. P.M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artif. Intell.*, **77**(2):321–358, 1995.
4. P.E. Dunne and T.J.M. Bench-Capon. Coherence in finite argument systems. *Artif. Intell.*, **141**(1/2):187–203, 2002.
5. W. Dvořák and S. Woltran. Complexity of semi-stable and stage semantics in argumentation frameworks. *Inf. Process. Lett.*, **110**(11):425–430, 2010.
6. W. Dvořák, S.A. Gaggl, J.P. Wallner, and S. Woltran. Making use of advances in answer-set programming for abstract argumentation systems. In *Proc. INAP*, 2011.
7. W. Dvořák, M. Järvisalo, J.P. Wallner, and S. Woltran. Complexity-sensitive decision procedures for abstract argumentation. In *Proc. KR*. AAAI Press, 2012. to appear.
8. U. Egly, S.A Gaggl, and S. Woltran. Answer-set programming encodings for argumentation frameworks. *Argument and Computation*, **1**(2):147–177, 2010.

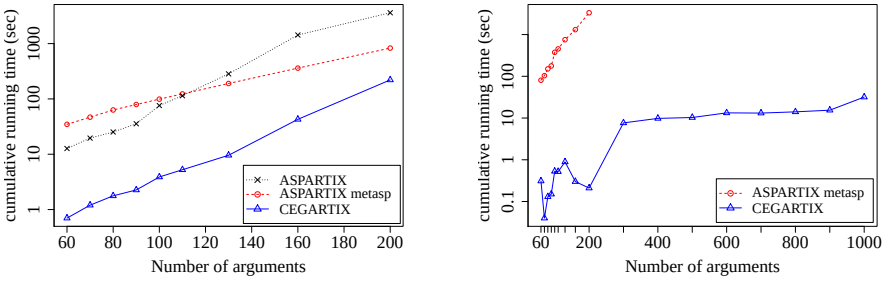


Fig. 1: Comparison of CEGARTIX (using Minisat) and ASPARTIX (both with the metasp variant and on the left also the non-metasp variant): cumulative running times over the random instances (left) and grid instances (right).

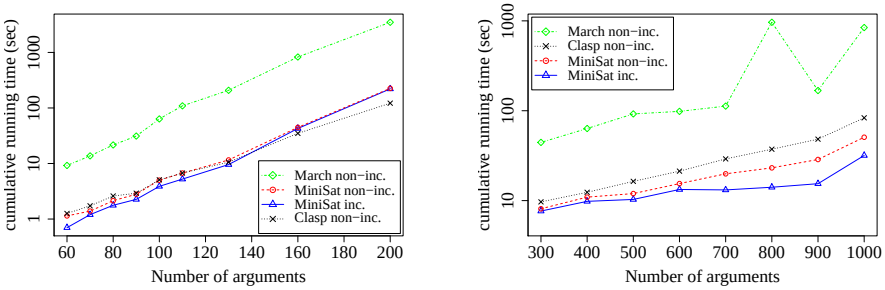


Fig. 2: CEGARTIX using different SAT solvers (non-incremental and incremental applications of Minisat, non-incremental application of Clasp, external application of March_rw): cumulative running times over the random (left) and grid instances (right).