Pakota: A System for Enforcement in Abstract Argumentation*

Andreas Niskanen, Johannes P. Wallner, and Matti Järvisalo

Helsinki Institute for Information Technology HIIT, Department of Computer Science,
University of Helsinki, Finland

Abstract. In this paper we describe Pakota, a system implementation that allows for solving enforcement problems over argumentation frameworks. Via harnessing Boolean satisfiability (SAT) and maximum satisfiability (MaxSAT) solvers, Pakota implements algorithms for extension and status enforcement under various central AF semantics, covering a range of NP-complete—via direct MaxSAT encodings—and Σ_2^P-complete—via MaxSAT-based counterexample-guided abstraction refinement—enforcement problems. We overview the algorithmic approaches implemented in Pakota, and describe in detail the system architecture, features, interfaces, and usage of the system. Furthermore, we present an empirical evaluation on the impact of the choice of MaxSAT solvers on the scalability of the system, and also provide benchmark generators for extension and status enforcement.

1 Introduction

Argumentation is a core area of modern artificial intelligence research, with strong connections to knowledge representation and classical and non-monotonic logics. Argumentation frameworks (AFs) [23], a central graph-based knowledge representation formalism, provide a formal basis for abstract argumentation.

Motivated also by practical applications, AFs under various semantics give rise to important—and often computationally very hard—reasoning problems over AFs. This includes what we refer to as *static* (or *non-dynamic*) AF reasoning tasks, such as the much studied problems of skeptical and credulous acceptance of arguments. Static AF reasoning tasks have been extensively studied, to the point that today several systems implementing static AF reasoning [33,13,14,24,26,30,31] are available. Most often these systems are based on declarative approaches, using propositional satisfiability (SAT) solver technology or extensions thereof for solving the core reasoning task at hand [13,14,24,26]. However, argumentation is intrinsically a dynamic process, and hence understanding and reasoning about the dynamics of AFs is a central and recent direction of research [9,8,10,12,18,19,21,22,34,29]. In contrast to static AF reasoning problems, few system implementations are currently available for reasoning about different aspects of AF dynamics [17,34,29].

* Work funded by Academy of Finland, grants 251170 COIN, 276412, and 284591; and Research Funds of the University of Helsinki.

In this paper, we describe in detail Pakota, a system for optimal *extension enforcement* [8,12,17,34] and *status enforcement* [29], two recently proposed hard computational problems dealing with dynamics (in connection to belief change) in abstract argumentation. In short, enforcement deals with the question of how a given AF should be revised (changed) in order for it to support (in terms of, e.g., skeptical or credulous acceptance) specific arguments.

Pakota implements algorithms for solving *optimally*—in terms of structural modifications to a given AF—various variants of NP-complete and Σ_2^P-complete extension and status enforcement problems under various AF semantics, being the first system for optimal enforcement in its generality. Pakota is based on NP-encoding enforcement problems using the Boolean optimization paradigm of maximum satisfiability (MaxSAT), and further implements *counterexample-guided abstraction refinement* (CEGAR) [15,16] algorithms based on SAT and MaxSAT solvers for Σ_2^P-complete enforcement.

2 Enforcement in Abstract Argumentation

We start by reviewing argumentation frameworks and their semantics [23,7], and the extension enforcement and status enforcement problems central to this work.

2.1 Argumentation Frameworks

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* defended *(in F) by a set $S \subseteq A$ if, for each $b \in A$ such that $(b, a) \in R$, there exists a $c \in S$ such that $(c, b) \in R$.*

Semantics for AFs are defined by functions σ which assign to each AF $F = (A, R)$ a set $\sigma(F) \subseteq 2^A$ of extensions. We consider for σ the functions stb, adm, com and prf, which stand for stable, admissible, complete and preferred, respectively.

Definition 2. *Given an AF $F = (A, R)$, the* characteristic function $\mathcal{F}_F : 2^A \to 2^A$ *of F is $\mathcal{F}_F(S) = \{a \in A \mid a \text{ is defended by } S\}$. Moreover, for a set $S \subseteq A$, the* range of *S is $S_R^+ = S \cup \{a \in A \mid (b, a) \in R, b \in S\}$.*

Definition 3. *Let $F = (A, R)$ be an AF. A set $S \subseteq A$ is* conflict-free (in F)*, if there are no $a, b \in S$ such that $(a, b) \in R$. We denote the collection of conflict-free sets of F by $cf(F)$. For a conflict-free set $S \in cf(F)$, it holds that*

- *$S \in stb(F)$ iff $S_R^+ = A$;*
- *$S \in adm(F)$ iff $S \subseteq \mathcal{F}_F(S)$;*
- *$S \in com(F)$ iff $S = \mathcal{F}_F(S)$;*
- *$S \in prf(F)$ iff $S \in adm(F)$ and there is no $T \in adm(F)$ with $S \subset T$;*

We use "σ-extension" to refer to an extension under a semantics $\sigma \in \{stb, adm, com, prf\}$.

Example 1. As an example AF, consider $F = (A, R)$ with three arguments, $A = \{a, b, c\}$, and attacks $R = \{(a, b), (b, a), (b, c), (c, b), (c, a)\}$, with a graphical illustration shown in Fig. 1(a). This AF has the following stable extensions which, in this particular case, coincide with the preferred extensions: $stb(F) = \{\{b\}, \{c\}\}$.

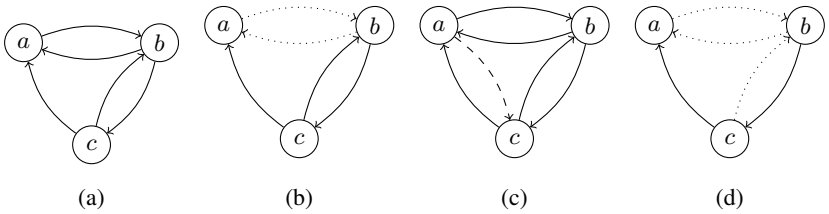

Fig. 1. An argumentation framework (a); enforcing $\{a, b\}$ to be a stable extension (b); credulous (c) and skeptical (d) status enforcement of $P = \{a, b\}$, $N = \emptyset$ under stable semantics

When comparing attack structures of two AFs $F = (A, R)$ and $F' = (A, R')$ with the same set of arguments, we make use of the cardinality of the symmetric difference of the attack relations defined by $|R \Delta R'| = |R \setminus R'| + |R' \setminus R|$.

2.2 Extension Enforcement

We continue by recalling the problem of extension enforcement [8,17,34], where we are given an AF $F = (A, R)$ and a subset $T \subseteq A$ of its arguments, and the goal is to modify the attack structure R such that T becomes (a subset of) an extension under the semantics σ in the modified AF $F' = (A, R')$.

Strict enforcement requires that the given set T of arguments has to be exactly a σ-extension. In *non-strict* enforcement, T is required to be a subset of some σ-extension. We denote the set of attack structures that strictly enforce T under σ for F by

$$enf(s, F, T, \sigma) = \{R' \mid F' = (A, R'),\ T \in \sigma(F')\},$$

$$\text{and by} \quad enf(ns, F, T, \sigma) = \{R' \mid F' = (A, R'),\ \exists T' \in \sigma(F')\ :\ T' \supseteq T\}$$

for non-strict enforcement. The number of changes of an enforcement is the size of the symmetric difference of the attack structures R and R'. From the computational perspective, we view extension enforcement as an optimization problem, seeking to minimize the number of changes to the attack structure.

Extension enforcement ($x \in \{s, ns\}$)
Input: AF $F = (A, R)$, $T \subseteq A$, semantics σ.
Task: Find an AF $F^* = (A, R^*)$ with

$$R^* \in \underset{R' \in enf(x,F,T,\sigma)}{\arg\min} |R \Delta R'|.$$

Example 2. Consider AF F from Example 1 (shown in Fig. 1(a)). For enforcing set $\{a, b\}$ to be a stable extension, an optimal solution AF is shown in Fig. 1(b) where the mutual attacks between a and b are removed. In this modified AF both $\{a, b\}$ and $\{c\}$ are stable extensions.

Table 1. Complexity of extension and status enforcement.

	extension enf.		status enf. ($N = \emptyset$)		status enf. (unrestr. case)	
σ	strict	non-strict	credulous	skeptical	credulous	skeptical
Conflict-free	in P	in P	in P	trivial	in P	trivial
Admissible	in P	NP-c	NP-c	trivial	Σ_2^P-c	trivial
Stable	in P	NP-c	NP-c	Σ_2^P-c	Σ_2^P-c	Σ_2^P-c
Complete	NP-c	NP-c	NP-c	NP-c	Σ_2^P-c	NP-c
Preferred	Σ_2^P-c	NP-c	NP-c	in Σ_3^P	Σ_2^P-c	in Σ_3^P

In the corresponding decision problem we are given in addition an integer $k \geq 0$ and are asked whether it is possible to enforce T with $|R \Delta R'| \leq k$. We recall the computational complexity results from [34] for this decision problem in Table 1. Note that non-strict extension enforcement under admissible, complete, and preferred semantics coincide; thus it suffices to implement an algorithm for one of these problems to cover all three.

2.3 Status Enforcement

In the status enforcement problem [29] we are given an AF $F = (A, R)$ and two disjoint subsets $P, N \subseteq 2^A$, $P \cap N = \emptyset$. The goal is to enforce the arguments in P *positively* and arguments in N *negatively*, i.e., to modify the attack structure R so that all arguments in P are credulously or skeptically accepted and all arguments in N are not accepted in the modified AF $F' = (A, R')$.

For *credulous status enforcement*, we denote the set of attack structures that enforce (P, N) under σ for F by

$$cr(F, P, N, \sigma) = \{R' \mid F' = (A, R'), P \subseteq \bigcup \sigma(F'), N \cap \bigcup \sigma(F') = \emptyset\},$$

and, for *skeptical status enforcement*,

$$sk(F, P, N, \sigma) = \{R' \mid F' = (A, R'), P \subseteq \bigcap \sigma(F'), N \cap \bigcap \sigma(F') = \emptyset\}.$$

For $\sigma = stb$ we additionally require for skeptical status enforcement that a solution AF F' has at least one stable extension. Like extension enforcement, we view status enforcement as an optimization problem, where the goal is to minimize the cardinality of the symmetric difference of the original and the modified attack structures R and R'.

Optimal Credulous Status Enforcement
Input: AF $F = (A, R)$, $P, N \subseteq A$, semantics σ.
Task: Find an AF $F^* = (A, R^*)$ with

$$R^* \in \operatorname*{arg\,min}_{R' \in cr(F,P,N,\sigma)} |R \Delta R'|.$$

Optimal Skeptical Status Enforcement
Input: AF $F = (A, R)$, $P, N \subseteq A$, semantics σ.
Task: Find an AF $F^* = (A, R^*)$ with

$$R^* \in \operatorname*{arg\,min}_{R' \in sk(F,P,N,\sigma)} |R \Delta R'|.$$

Example 3. For the AF from Example 1, we see in Fig. 1 (c) credulous and (d) skeptical status enforcement for $P = \{a, b\}$, $N = \emptyset$ under the stable semantics. In the modified AF shown in Fig. 1(c) we have added an attack from a to c, which results in an AF where $\{a\}$, $\{b\}$, and $\{c\}$ are all stable extensions. In the AF shown in Fig. 1(d) we have removed the mutual attacks between a and b, and removed the attack from c to b. This results in $\{a, b\}$ being the unique stable extension of this modified AF.

The decision problem corresponding to status enforcement is the following: given an AF $F = (A, R)$, positive and negative sets $P, N \subseteq A$ of argument statuses, a semantics σ, and an integer $k \geq 0$, can the statuses in P and N be enforced under σ with at most k modifications to the attack structure R. The computational complexity of the decision problem was established in [29]; Table 1 provides an overview. Note that credulous status enforcement under the admissible, complete, and preferred semantics coincide [29].

3 Maximum Satisfiability

For solving variants of extension and status enforcement problems, Pakota employs constraint optimization encodings using (partial) maximum satisfiability (MaxSAT for short) as the underlying declarative language. In MaxSAT, for each variable x, we have two literals, x and $\neg x$. A clause is a disjunction ($\vee$) of literals. A truth assignment is a function from variables to $\{0, 1\}$. A clause c is satisfied by a truth assignment τ, $\tau(c) = 1$, if $\tau(x) = 1$ for a literal x in c or $\tau(x) = 0$ for a literal $\neg x$ in c; otherwise τ does not satisfy c, $\tau(c) = 0$. An instance $\varphi = (\varphi_h, \varphi_s)$ of the MaxSAT problem consists of a set φ_h of *hard* clauses, and a set φ_s of *soft* clauses. Any truth assignment τ which satisfies each hard clause is a *solution* to φ. The *cost* of a solution is defined by $\textsc{cost}(\varphi, \tau) = \sum_{c \in \varphi_s} (1 - \tau(c))$, which is the number of soft clauses not satisfied by τ. A solution τ is *optimal* for φ if $\textsc{cost}(\varphi, \tau) \leq \textsc{cost}(\varphi, \tau')$ for all solutions τ' to φ. The output of a MaxSAT solver is an optimal solution to φ.

4 Pakota

The Pakota system is implemented in the C++ programming language. The source code is available at `http://www.cs.helsinki.fi/group/coreo/pakota/` under the MIT license. In what follows, we describe the main components and system architecture of the system (Section 4.1) and main features of Pakota (Section 4.2), detail the implemented algorithms (Section 4.3), input and output specifications (Section 4.4), and usage (Section 4.5).

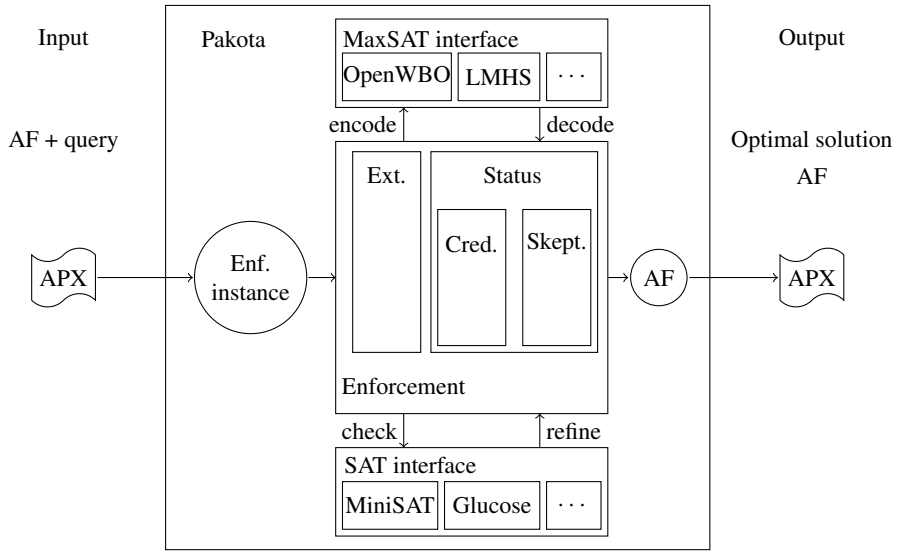

Fig. 2. System architecture of Pakota

4.1 System Architecture

The system architecture of Pakota is shown in Figure 2. Pakota accepts input for the extension enforcement problem and for the credulous and skeptical status enforcement problem in the so-called APX format (see Section 4.4), which is parsed into an enforcement instance. The algorithms implemented in Pakota that solve the given enforcement instance form the main component of the system and are described in Section 4.3, employing a MaxSAT solver, or, for problem variants beyond NP, interacting MaxSAT and SAT solvers. Pakota offers a generic MaxSAT interface for plugging in the MaxSAT solver of choice and already includes MaxSAT solvers Open-WBO [27] (version 1.3.1) and LMHS [32] (version 2015.11), and the SAT solvers MiniSAT [25] (version 2.2.0, included with LMHS) and Glucose [4,6,5] (version 3.0, included with Open-WBO). We detail usage of the MaxSAT interface in Section 4.2.

The implemented algorithms for the enforcement problems can be classified according to whether they solve an NP problem or a second-level problem. For the former, the enforcement instance is encoded in a MaxSAT instance and the solution given by a MaxSAT solver is decoded to construct a solution AF to the enforcement problem, again in the APX format. In the case that the given task is a second-level problem, the algorithms implement a counterexample-guided abstraction refinement procedure, thereby iteratively querying the MaxSAT solver to construct candidate solutions and checking whether the candidate is indeed a solution to the enforcement problem via a SAT solver. In case the candidate is a solution, the decoded AF is returned in the APX format. Otherwise, i.e., in case the candidate is a non-solution, the current MaxSAT encoding is iteratively refined until an actual optimal solution is found.

4.2 Features

Supported Semantics and Reasoning Modes An overview of the semantics and reasoning modes currently supported by Pakota is given in Table 2. Implementation of different parameter choices are discussed in more detail in Section 4.3.

MaxSAT and SAT Solver Interfaces Essentially any MaxSAT solver whose source code is available can be plugged into the system. This is enabled in Pakota by offering two interfaces, `MaxSATSolver.h` and `SATSolver.h`. By creating new classes that implement these interfaces and defining the pure virtual functions declared in them, one can compile and link these to the Pakota system, which will then use the corresponding MaxSAT and SAT solvers for solving the enforcement problems. As an implementation-level detail, note that, if the MaxSAT solver uses a SAT solver internally, which is usually the case, an easy solution to potential naming conflicts is to use the same SAT solver as the SAT solver in CEGAR procedures within Pakota. The source code of Pakota already includes implementations of these interfaces for two different MaxSAT solvers, Open-WBO [27] and LMHS [32], allowing the use of these solvers simply by editing the `MAXSAT_SOLVER` parameter in the included Makefile before compiling.

MaxSAT and IP Encodings In addition to directly solving extension and status enforcement instances, Pakota can for the NP variants of the problems output the internal MaxSAT encodings both in the standard WCNF MaxSAT input format as well as integer programs (IPs) in the standard LP format (applying the standard textbook encoding of MaxSAT as IP [3]). The latter option allows for calling state-of-the-art IP solvers, such as CPLEX or Gurobi, on the encodings.

4.3 Algorithms

Depending on the inherent complexity of the problems, Pakota solves the extension or status enforcement problem at hand by either encoding the problem in MaxSAT (NP-complete problems), or within a counterexample-guided abstraction refinement (CEGAR) scheme utilizing a MaxSAT solver in an iterative or incremental fashion (problems complete for the second level of polynomial hierarchy). Table 2 provides details, depending on the chosen parameters and semantics, for each problem variant, whether it is solved via direct encoding to MaxSAT (detailed in Fig. 3) or via a MaxSAT-based CEGAR algorithm (detailed as Algorithms 1 and 2).

Encoding NP Enforcement in MaxSAT Let $F = (A, R)$ be an AF. We utilize Boolean variables x_a and x_a^p for $a, p \in A$, and variables $r_{a,b}$ for $a, b \in A$. The intended meaning of these variables is that if x_a (x_a^p) is assigned true in an assignment then a is contained in a σ-extension in a specific AF. The AF we are referring to is either directly encoded in the formula or encoded via a truth assignment on variables $r_{a,b}$, i.e., if $r_{a,b}$ is assigned true, then there is an attack from a to b. For all considered problems, soft clauses are defined by $\varphi_s(F) = \bigwedge_{a,b \in A} r'_{a,b}$, where $r'_{a,b}$ is $r_{a,b}$ if $(a, b) \in R$, and $\neg r_{a,b}$ if $(a, b) \notin R$. Violating a soft clause corresponds to an attack being removed or added, and incurs an associated unit cost.

Hard clauses are problem dependent. The complete list of encodings used in Pakota is provided in Fig. 3. In particular, EXT refers to encodings for extension enforcement

Table 2. Extension and status enforcement problems currently supported by Pakota.

Problem	parameters	semantics	Encoding/Algorithm
extension enforcement	ns	adm, com, prf	$\text{EXT}(ns, F, T, adm)$
extension enforcement	ns	stb	$\text{EXT}(ns, F, T, stb)$
extension enforcement	s	adm	$\text{EXT}(s, F, T, adm)$
extension enforcement	s	com	$\text{EXT}(s, F, T, com)$
extension enforcement	s	stb	$\text{EXT}(s, F, T, stb)$
extension enforcement	s	prf	Algorithm 1
status enforcement	cr, $N = \emptyset$	adm, com, prf	$\text{STAT}(cr, A, P, \emptyset, adm)$
status enforcement	cr, $N = \emptyset$	stb	$\text{STAT}(cr, A, P, \emptyset, stb)$
status enforcement	cr	adm, com, prf	Algorithm 2
status enforcement	cr	stb	Algorithm 2
status enforcement	sk	adm	Trivial
status enforcement	sk	stb	Algorithm 2

for strict (s) and non-strict (ns) modes. The other parameters are an AF $F = (A, R)$, a semantics $\sigma \in \{adm, com, stb, prf\}$, and $T \subseteq A$. For encoding the semantics, we adapt Boolean formulas from [11], originally presented for static AF reasoning problems. We also note that [17] apply similar encodings to ours in an integer programming based approach to the specific case of extension enforcement under admissible semantics.

Figure 2 shows for each NP-complete extension enforcement problem the corresponding MaxSAT encoding for which it holds that an optimal MaxSAT solution directly corresponds to an optimal solution for the extension enforcement problem. For instance, to optimally solve non-strict extension enforcement under the admissible semantics, we encode the input AF and set of arguments to be enforced via formula $\text{EXT}(ns, F, T, adm)$ and subsequently call $\text{MAXSAT}(\text{EXT}(ns, F, T, adm), \varphi_s(F))$ to compute an optimal MaxSAT solution (c, τ), with cost c and assignment τ, from which we can infer an optimal solution to the corresponding problem by extracting a new AF $F' = (A, R')$ with $R' = \{(a, b) \mid \tau(r_{a,b}) = 1\}$.

For the NP-complete status enforcement problems of credulous status enforcement under the admissible and stable semantics with empty negative set $N = \emptyset$, we implemented an analogous procedure. For the input to this problem, i.e., AF $F = (A, R)$ and positive set $P \subseteq A$, we give the MaxSAT solver the encoding $\text{STAT}(cr, A, P, \emptyset, \sigma)$, with $\sigma \in \{adm, stb\}$. From an optimal MaxSAT solution we can infer an optimal solution to the status enforcement problem similarly as for extension enforcement by generating a new AF $F' = (A, R')$ with $R' = \{(a, b) \mid \tau(r_{a,b}) = 1\}$.

The remaining encodings in Fig. 3 are used in our CEGAR algorithms for the second-level complete problems.

Counterexample-guided Abstraction Refinement Pakota implements the second-level complete problems arising in status enforcement and extension enforcement by a counterexample-guided abstraction refinement (CEGAR) approach. Concretely, we let a MaxSAT solver compute a candidate solution from an NP abstraction of the second-level complete problem, and subsequently check whether the candidate is a solution with a SAT solver. In case a solution is found, i.e., the SAT solver reports unsatisfiability, we extract from the MaxSAT solution an optimal solution to the enforcement problem.

$$\textsc{ext}(ns,F,T,adm) = \bigwedge_{a\in T} x_a \wedge \bigwedge_{a,b\in A} \Big(\big(r_{a,b} \rightarrow (\neg x_a \vee \neg x_b)\big) \wedge \big((x_a \wedge r_{b,a}) \rightarrow \bigvee_{c\in A} (x_c \wedge r_{c,b})\big)\Big)$$

$$\textsc{ext}(ns,F,T,stb) = \bigwedge_{a\in T} x_a \wedge \bigwedge_{a,b\in A} \big(r_{a,b} \rightarrow (\neg x_a \vee \neg x_b)\big) \wedge \bigwedge_{a\in A} \big(\neg x_a \rightarrow \bigvee_{b\in A} (x_b \wedge r_{b,a})\big)$$

$$\textsc{ext}(s,F,T,adm) = \bigwedge_{a,b\in T} \neg r_{a,b} \wedge \bigwedge_{a\in T} \bigwedge_{b\in A\setminus T} \big(r_{b,a} \rightarrow \bigvee_{c\in T} r_{c,b}\big)$$

$$\textsc{ext}(s,F,T,com) = \bigwedge_{a,b\in T} \neg r_{a,b} \wedge \bigwedge_{a\in T} \bigwedge_{b\in A\setminus T} \big(r_{b,a} \rightarrow \bigvee_{c\in T} r_{c,b}\big) \wedge \bigwedge_{a\in A\setminus T} \bigvee_{b\in A} \big(r_{b,a} \wedge \bigwedge_{c\in T} \neg r_{c,b}\big)$$

$$\textsc{ext}(s,F,T,stb) = \bigwedge_{a,b\in T} \neg r_{a,b} \wedge \bigwedge_{a\in A\setminus T} \bigvee_{b\in T} r_{b,a}$$

$$\textsc{check}(\tau) = \bigwedge_{\tau(r_{a,b})=1} (\neg x_a \vee \neg x_b) \wedge \bigwedge_{\tau(r_{b,a})=1} \big(x_a \rightarrow \bigvee_{\tau(r_{c,b})=1} x_c\big) \wedge \bigwedge_{\tau(x_a)=1} x_a \wedge \bigvee_{\tau(x_a)=0} x_a$$

$$\psi(A) = \bigwedge_{a,b\in A} \big(r_{a,b} \rightarrow (\neg x_a^p \vee \neg x_b^p)\big)$$

$$\textsc{stat}(cr,A,P,N,adm) = \bigwedge_{p\in P} \Big(\psi(A) \wedge \bigwedge_{a,b\in A} \big((x_a^p \wedge r_{b,a}) \rightarrow \bigvee_{c\in A} (x_c^p \wedge r_{c,b})\big) \wedge x_p^p \wedge \bigwedge_{n\in N} \neg x_n^p\Big)$$

$$\textsc{stat}(cr,A,P,N,stb) = \bigwedge_{p\in P} \Big(\psi(A) \wedge \bigwedge_{a\in A} \big(\neg x_a^p \rightarrow \bigvee_{b\in A} (x_b^p \wedge r_{b,a})\big) \wedge x_p^p \wedge \bigwedge_{n\in N} \neg x_n^p\Big)$$

$$\textsc{stat}(sk,A,P,N,stb) = \bigwedge_{n\in N} \Big(\psi(A) \wedge \bigwedge_{a\in A} \big(\neg x_a^p \rightarrow \bigvee_{b\in A} (x_b^p \wedge r_{b,a})\big) \wedge \neg x_n^n \wedge \bigwedge_{p\in P} x_p^n\Big)$$

$$\textsc{check}(cr,A,\tau,P,N,adm) = \bigwedge_{\tau(r_{a,b})=1} (\neg x_a \vee \neg x_b) \wedge \bigwedge_{\tau(r_{b,a})=1} (x_a \rightarrow \bigvee_{\tau(r_{c,b})=1} x_c) \wedge \bigvee_{n\in N} x_n$$

$$\textsc{check}(cr,A,\tau,P,N,stb) = \bigwedge_{\tau(r_{a,b})=1} (\neg x_a \vee \neg x_b) \wedge \bigwedge_{a\in A} (\neg x_a \rightarrow \bigvee_{\tau(r_{b,a})=1} x_b) \wedge \bigvee_{n\in N} x_n$$

$$\textsc{check}(sk,A,\tau,P,N,stb) = \bigwedge_{\tau(r_{a,b})=1} (\neg x_a \vee \neg x_b) \wedge \bigwedge_{a\in A} (\neg x_a \rightarrow \bigvee_{\tau(r_{b,a})=1} x_b) \wedge \bigvee_{p\in P} \neg x_p$$

$$\textsc{refine}(\tau) = \neg\Big(\bigwedge_{\tau(r_{a,b})=1} r_{a,b} \wedge \bigwedge_{\tau(r_{a,b})=0} \neg r_{a,b}\Big)$$

Fig. 3. Encoding extension and status enforcement

Otherwise, we call the MaxSAT solver again on a refined formula which includes further hard clauses extracted from the counterexample delivered by the SAT solver.

The CEGAR algorithms implemented in Pakota are shown in Algorithm 1 for extension enforcement, and in Algorithm 2 for status enforcement. We describe the algorithm for extension enforcement, as the CEGAR algorithm for status enforcement is similar (the main difference lies in the used formulas).

For extension enforcement, we implemented the second-level complete problem of strict extension enforcement under the preferred semantics as shown in Algorithm 1. Given an AF $F = (A, R)$, a set $T \subseteq A$ to enforce, we define the initial hard clauses φ_h to be the same as for the NP-complete strict extension enforcement problem under the complete semantics. In the while-loop, we call the MaxSAT solver on this set of hard clauses augmented with the same soft clauses, $\varphi_s(F)$, as for the NP-complete variants. From an optimal solution τ delivered by the MaxSAT solver, we check whether this candidate is a solution to strict extension enforcement under the preferred semantics using the formula $\text{CHECK}(\tau)$ (see Fig. 3). If the SAT solver reports unsatisfiability of this formula, we terminate and return the AF encoded in τ. Otherwise we refine, i.e., increment, the hard clauses by $\text{REFINE}(\tau)$ (see again Fig. 3 for details).

For status enforcement we implemented Algorithm 2. For a given input to the second-level complete problems for status enforcement we consider here, i.e., credulous status enforcement under the admissible and stable semantics, and skeptical status enforcement under the stable semantics, this algorithm computes an optimal solution AF. The input for this problem consists of an AF $F = (A, R)$ and sets $P, N \subseteq A$.

Algorithm 1 Extension enforcement

$\varphi_h \leftarrow \text{EXT}(s, F, T, com)$
while true **do**
  $(c, \tau) \leftarrow \text{MAXSAT}(\varphi_h, \varphi_s(F))$
  $r \leftarrow \text{SAT}(\text{CHECK}(\tau))$
  if $r = unsat$ **then** return (c, τ)
  else $\varphi_h \leftarrow \varphi_h \wedge \text{REFINE}(\tau)$

Algorithm 2 Status enforcement

1: $\varphi_h \leftarrow \text{STAT}(M, A, P, N, \sigma)$
2: **while** true **do**
3: $(c, \tau) \leftarrow \text{MAXSAT}(\varphi_h, \varphi_s(F))$
4: $r \leftarrow \text{SAT}(\text{CHECK}(M, A, \tau, P, N, \sigma))$
5: **if** $r = unsat$ **then** return (c, τ)
6: **else** $\varphi_h \leftarrow \varphi_h \wedge \text{REFINE}(\tau)$

4.4 Input Format

For extension enforcement, the input AF and enforcement request are specified using the following predicates, extending the APX format for specifying AFs.

`arg(X)`: `X` is an argument
`att(X,Y)`: there is an attack from `X` to `Y`
`enf(X)`: enforce argument `X`

Example 4. The enforcement of argument a for the AF in Fig. 4(a) is specified in the Pakota input format as shown in Fig. 4(b). On this input, Pakota may return the output shown in Fig. 4(c), i.e., the AF in Fig. 4(d).

As in extension enforcement, for status enforcement the AF is represented using the `arg` and `att` predicates. The arguments to be positively and negatively enforced are represented via the pos and neg predicates, respectively. For example, `pos`(a). enforces argument a positively. The reasoning mode between credulous and skeptical is chosen from the command line.

4.5 Usage and Options

After compilation, the Pakota system is used from the command line with

```
./pakota <file> <mode> <sem> [options]
```

The command line arguments enabling the choice of AF semantics and reasoning mode are the following.

```
<file>    : Input filename for enforcement instance in apx format.
<mode>    : Enforcement variant: mode={strict|non-strict|cred|skept}
    strict     : strict extension enforcement
    non-strict : non-strict extension enforcement
    cred       : credulous status enforcement
    skept      : skeptical status enforcement
<sem>     : Argumentation semantics. sem={adm|com|stb|prf}
    adm        : admissible
    com        : complete
    stb        : stable
    prf        : preferred
```

Furthermore, command line options `-h` (for help message), `-v` (for version number), `-o` (for specifying output to file) and `-t` (for outputting NP-encodings in WCNF and LP formats) are available.

4.6 Benchmarks and Generators

The Pakota webpage also offers sets of benchmarks for both extension enforcement and status enforcement in the Pakota input format. Furthermore, we provide via the webpage our benchmark generator software, AfGen and EnfGen, which we used to generate the benchmark sets. The AF generator AfGen forms argumentation frameworks in APX format implementing the Erdős-Rényi random digraph model. The generator is called as

```
./afgen <args> <prob>
```

where parameters `<args>` and `<prob>` specify the number of arguments and the probability of an attack in the output AF. The generator forms an argumentation framework with arguments $1, \ldots,$ `<args>`, including an attack between each pair of arguments independently with probability `<prob>`.

The enforcement instance generator EnfGen takes as input an AF in APX format, and produces an enforcement instance. It is called as

```
./enfgen <file> <mode> <enfs>
```

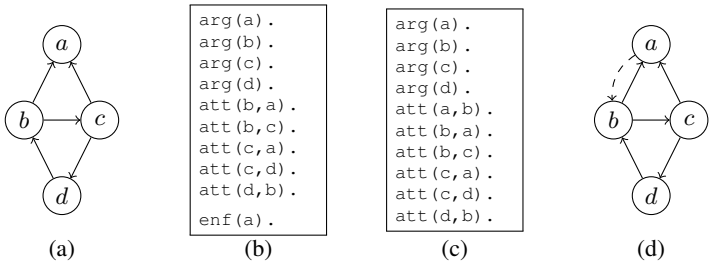

Fig. 4. Example of Pakota input and output formats

where `<file>` is the input AF and `<mode>` is either `ext` or `status`, corresponding to extension and status enforcement, respectively. In case of extension enforcement, `<enfs>` is an integer stating the number of arguments to be enforced, and for status enforcement, `<enfs>` is a pair of integers, corresponding to the number of positively and negatively enforced arguments. The generator reads the arguments from the AF and samples the enforced arguments uniformly at random without replacement.

5 Performance Overview

We empirically evaluate the impact of the choice of the underlying MaxSAT solver on the performance of Pakota on various NP-complete and Σ_2^P-complete variants of extension and status enforcement. This complements the scalability experiments using only a single solver presented in [34,29], as well as the comparison presented in [34] with the IP-based approach to extension enforcement under admissible semantics described in [17]. For the NP problems, we used five state-of-the-art MaxSAT solvers: MaxHS [20], Maxino [1], MSCG [28], Open-WBO [27], and WPM [2], using the newest MaxSAT Evaluation 2015 versions, as well as the commercial IBM CPLEX integer programming solver (version 12.6). For CEGAR, we compare the performance of Open-WBO and LMHS [32] as the underlying MaxSAT solvers, as supported by Pakota. The experiments were run on 2.83-GHz Intel Xeon E5440 quad-core machines with 32-GB RAM and Debian GNU/Linux 8 using a timeout of 900 seconds per instance.

We generated the benchmarks using our AfGen and EnfGen generators. For extension enforcement, for each number of arguments $|A| \in \{25, 50, \dots\}$ and each edge probability $p \in \{0.05, 0.1, 0.2, 0.3\}$, we generated five AFs. For each AF, we generated five enforcement instances with $|T|$ enforced arguments, for each $|T|/|A| \in \{0.05, 0.1, 0.2, 0.3\}$. We thus obtained 400 instances for each $|A|$. For status enforcement, for each $|A| \in \{20, 40, \dots, 200\}$ and $p \in \{0.05, 0.1, \dots, 0.35\}$, we generated 10 AFs. For each AF, we generated an enforcement instance containing $(|P|, |N|) \in \{(1,0), (2,0) \dots, (5,0), (5,1), (2,2), (1,5)\}$ positively and negatively enforced arguments. This gave a total of 560 status enforcement instances for each $|A|$.

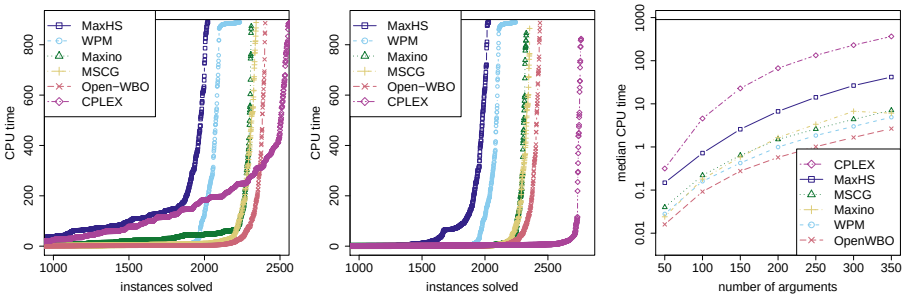

Fig. 5. MaxSAT solver comparison on NP-complete extension enforcement. Left: non-strict admissible; middle: non-strict stable; right: strict complete

An overview of the results, comparing the different underlying MaxSAT solvers, is provided in Fig. 5 (NP-complete extension enforcement), Fig. 6 (NP-complete status enforcement), and Fig. 7 (CEGAR for extension and status enforcement). Fig. 5 left and middle show the number of instances solved (x-axis) by different MaxSAT solvers under different per-instance timeouts (y-axis) for non-strict extension enforcement under the admissible (left) and stable semantics (middle). Interestingly, in both cases CPLEX performs well (although on admissible, on a majority of the instances is solved faster by most of the other solvers).

On strict extension enforcement under the complete semantics (Fig. 5 right), the median runtimes for CPLEX scale noticeably worse than for the rest of the solvers wrt the number of arguments. However, here we note that only CPLEX and Maxino were able to solve all instances; thus Maxino turned out to be clearly the best solver on strict complete. Fig. 6 provides an overview for credulous status enforcement under the admissible semantics. Here we observe that the so-called core-guided MaxSAT solvers perform the best, while the SAT-IP hybrid solver MaxHS—typically competitive mainly on weighted MaxSAT instances—performs the worst. We also observed similar performance under the stable semantics. Overall, for the NP-complete enforcement problems, CPLEX and Maxino tend to provide the best choice of solvers, but the choice of the single best solver tends to depend on the problem variant (strict/non-strict, semantics).

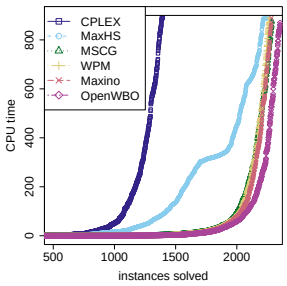

Fig. 6. NP-complete credulous status enforcement under admissible

Turning to the Σ_2^P-complete enforcement problems Fig. 7 gives an overview of the performance of Open-WBO and LMHS within our CEGAR procedures for strict extension enforcement under the preferred semantics (left), and credulous (middle) and skeptical (right) status enforcement under the stable semantics. Evidently, on these instances generated with our EnfGen, out of the two solvers Open-WBO provides the best MaxSAT solver for the CEGAR procedures.

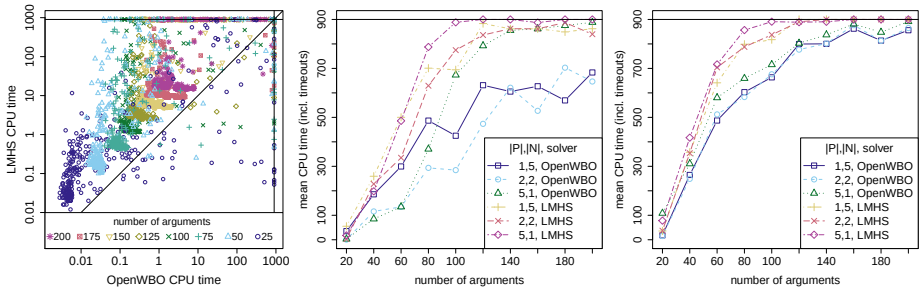

Fig. 7. MaxSAT solver comparison within CEGAR. Left: strict extension enforcement under preferred; credulous (middle) and skeptical (right) status enforcement under stable

6 Conclusions

The Pakota system is a first system implementation in its generality for solving NP-complete and Σ_2^P-complete problem instances of extension enforcement and status enforcement—two related problems motivated by the study of dynamic aspects of argumentation frameworks. We provided a detailed overview of the Pakota system—available in open source—including the input-output format, system design, functionality, details on the underlying MaxSAT encodings and MaxSAT-based CEGAR algorithms implemented in Pakota, and its API allowing for plugging in different SAT and MaxSAT solvers used as core search engines. We also provided a detailed evaluation of the impact of the choice of MaxSAT solvers (including the use of the state-of-the-art integer programming system CPLEX) on the performance of Pakota on various variants of extension and status enforcement problems. In addition to Pakota, we also provide open-source benchmark generators for extension and status enforcement for the use of the research community at large through the Pakota system webpage. Future work on Pakota includes extensions to support further central AF semantics, including grounded, semi-stable, and stage.

References

1. Alviano, M., Dodaro, C., Ricca, F.: A MaxSAT algorithm using cardinality constraints of bounded size. In: Proc. IJCAI. pp. 2677–2683. AAAI Press / IJCAI (2015)
2. Ansótegui, C., Didier, F., Gabàs, J.: Exploiting the structure of unsatisfiable cores in MaxSAT. In: Proc. IJCAI. pp. 283–289. AAAI Press / IJCAI (2015)
3. Ansótegui, C., Gabàs, J.: Solving (weighted) partial MaxSAT with ILP. In: Proc. CPAIOR. Lecture Notes in Computer Science, vol. 7874, pp. 403–409. Springer (2013)
4. Audemard, G., Simon, L.: Predicting learnt clauses quality in modern SAT solvers. In: Proc. IJCAI. pp. 399–404. AAAI Press / IJCAI (2009)
5. Audemard, G., Simon, L.: GLUCOSE 2.1: Aggressive – but reactive – clause database management, dynamic restarts. In: Pragmatics of SAT (Workshop of SAT'12) (2012)
6. Audemard, G., Simon, L.: Refining restarts strategies for SAT and UNSAT. In: Proc. CP. Lecture Notes in Computer Science, vol. 7514, pp. 118–126. Springer (2012)
7. Baroni, P., Caminada, M., Giacomin, M.: An introduction to argumentation semantics. Knowledge Engineering Review 26(4), 365–410 (2011)
8. Baumann, R.: What does it take to enforce an argument? Minimal change in abstract argumentation. In: Proc. ECAI. Frontiers in Artificial Intelligence and Applications, vol. 242, pp. 127–132. IOS Press (2012)
9. Baumann, R.: Normal and strong expansion equivalence for argumentation frameworks. Artificial Intelligence 193, 18–44 (2012)
10. Baumann, R., Brewka, G.: AGM meets abstract argumentation: Expansion and revision for Dung frameworks. In: Proc. IJCAI. pp. 2734–2740. AAAI Press / IJCAI (2015)
11. Besnard, P., Doutre, S.: Checking the acceptability of a set of arguments. In: Proc. NMR. pp. 59–64 (2004)
12. Bisquert, P., Cayrol, C., de Saint-Cyr, F.D., Lagasquie-Schiex, M.: Enforcement in argumentation is a kind of update. In: Proc. SUM. Lecture Notes in Computer Science, vol. 8078, pp. 30–43. Springer (2013)
13. Cerutti, F., Dunne, P.E., Giacomin, M., Vallati, M.: Computing preferred extensions in abstract argumentation: A SAT-based approach. In: Proc. TAFA. Lecture Notes in Computer Science, vol. 8306, pp. 176–193. Springer (2014)

14. Cerutti, F., Giacomin, M., Vallati, M.: ArgSemSAT: Solving argumentation problems using SAT. In: Proc. COMMA. Frontiers in Artificial Intelligence and Applications, vol. 266, pp. 455–456. IOS Press (2014)
15. Clarke, E.M., Grumberg, O., Jha, S., Lu, Y., Veith, H.: Counterexample-guided abstraction refinement for symbolic model checking. Journal of the ACM 50(5), 752–794 (2003)
16. Clarke, E.M., Gupta, A., Strichman, O.: SAT-based counterexample-guided abstraction refinement. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 23(7), 1113–1123 (2004)
17. Coste-Marquis, S., Konieczny, S., Mailly, J., Marquis, P.: Extension enforcement in abstract argumentation as an optimization problem. In: Proc. IJCAI. pp. 2876–2882. AAAI Press (2015)
18. Coste-Marquis, S., Konieczny, S., Mailly, J., Marquis, P.: On the revision of argumentation systems: Minimal change of arguments statuses. In: Proc. KR. pp. 52–61. AAAI Press (2014)
19. Coste-Marquis, S., Konieczny, S., Mailly, J., Marquis, P.: A translation-based approach for revision of argumentation frameworks. In: Proc. JELIA. Lecture Notes in Computer Science, vol. 8761, pp. 397–411. Springer (2014)
20. Davies, J., Bacchus, F.: Exploiting the power of MIP solvers in MAXSAT. In: Proc. SAT. pp. 166–181. Springer (2013)
21. Delobelle, J., Konieczny, S., Vesic, S.: On the aggregation of argumentation frameworks. In: Proc. IJCAI. pp. 2911–2917. AAAI Press / IJCAI (2015)
22. Diller, M., Haret, A., Linsbichler, T., Rümmele, S., Woltran, S.: An extension-based approach to belief revision in abstract argumentation. In: Proc. IJCAI. pp. 2926–2932. AAAI Press / IJCAI (2015)
23. Dung, P.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artificial Intelligence 77(2), 321–358 (1995)
24. Dvořák, W., Järvisalo, M., Wallner, J.P., Woltran, S.: Complexity-sensitive decision procedures for abstract argumentation. Artificial Intelligence 206, 53–78 (2014)
25. Eén, N., Sörensson, N.: An extensible SAT-solver. In: Proc. SAT. Lecture Notes in Computer Science, vol. 2919, pp. 502–518. Springer (2003)
26. Egly, U., Gaggl, S.A., Woltran, S.: Answer-set programming encodings for argumentation frameworks. Argument and Computation 1(2), 147–177 (2010)
27. Martins, R., Manquinho, V.M., Lynce, I.: Open-WBO: A modular MaxSAT solver,. In: Proc. SAT. pp. 438–445. Springer (2014)
28. Morgado, A., Ignatiev, A., Marques-Silva, J.: MSCG: Robust core-guided MaxSAT solving. Journal on Satisfiability, Boolean Modeling and Computation 9, 129–134 (2015)
29. Niskanen, A., Wallner, J.P., Järvisalo, M.: Optimal status enforcement in abstract argumentation. In: Proc. IJCAI. AAAI Press / IJCAI (2016)
30. Nofal, S., Atkinson, K., Dunne, P.E.: Algorithms for decision problems in argument systems under preferred semantics. Artificial Intelligence 207, 23–51 (2014)
31. Nofal, S., Atkinson, K., Dunne, P.E.: Looking-ahead in backtracking algorithms for abstract argumentation. International Journal of Approximate Reasoning 78, 265–282 (2016)
32. Saikko, P., Berg, J., Järvisalo, M.: LMHS: A SAT-IP hybrid maxsat solver. In: Proc. SAT. Lecture Notes in Computer Science, vol. 9710, pp. 539–546. Springer (2016)
33. Thimm, M., Villata, S., Cerutti, F., Oren, N., Strass, H., Vallati, M.: Summary report of the first international competition on computational models of argumentation. AI Magazine 37(1), 102 (2016)
34. Wallner, J.P., Niskanen, A., Järvisalo, M.: Complexity results and algorithms for extension enforcement in abstract argumentation. In: Proc. AAAI. pp. 1088–1094. AAAI Press (2016)