A SAT-based Approach for Argumentation Dynamics

Extended Abstract

Jean-Marie Lagniez
CRIL, Université d’Artois - CNRS
Lens, France
lagniez@cril.fr

Emmanuel Lonca
CRIL, Université d’Artois - CNRS
Lens, France
lonca@cril.fr

Jean-Guy Mailly
Université Paris Cité, LIPADE
F-75006 Paris, France
jean-guy.mailly@u-paris.fr

ABSTRACT

In the realm of multi-agent systems, argumentative dialogues for persuasion and negotiation involve autonomous agents exchanging arguments, necessitating continual re-evaluation of argument acceptability. This study introduces a novel approach using modern SAT solving techniques to dynamically reassess the acceptability status of arguments, aligning with various classical semantics. Our method uses the assumption mechanism in SAT solvers, distinguished by minimal assumptions, ensuring practicality.

KEYWORDS

Abstract Argumentation; Argumentation Dynamics; SAT

1 INTRODUCTION

Abstract argumentation [10] is a robust tool for modeling conflicting information and providing specialized reasoning methods. Its versatile applications span various domains, particularly within multi-agent systems [19]. It serves as a foundation for defining protocols in multi-agent argument-based dialogues [4], crucial in automated negotiations [3] and persuasive interactions [5].

An abstract argumentation framework (AF) is a directed graph \( F = (\mathcal{A}, \mathcal{R}) \) where \( \mathcal{A} \) is a (non-empty and finite) set of arguments and \( \mathcal{R} \subseteq \mathcal{A} \times \mathcal{A} \) is the attack relation. Given \( a, b \in \mathcal{A} \), \( a \) attacks \( b \) when \( (a, b) \in \mathcal{R} \), and for \( S \subseteq \mathcal{A} \), \( S \) attacks \( b \) if \( \exists a \in S \) which attacks \( b \). Finally, \( S \) defends the argument \( c \in \mathcal{A} \) if, for any \( b \) attacking \( c \), \( S \) attacks \( b \). Reasoning with AFs typically uses the concept of extensions, i.e. sets of jointly acceptable arguments. These extension-based semantics are primarily based on two properties: given \( F = (\mathcal{A}, \mathcal{R}) \), \( S \subseteq \mathcal{A} \) is conflict-free if \( \forall a, b \in S, (a, b) \notin \mathcal{R} \); admissible if it is conflict-free and defends all its elements. In this paper, our primary focus lies on the following two types of extensions: \( S \subseteq \mathcal{A} \) is a complete extension if it is admissible and it defends no argument in \( \mathcal{A} \setminus S \); a stable extension if it is conflict-free and it attacks all the arguments in \( \mathcal{A} \setminus S \). We write \( \text{co}(F) \) (resp. \( \text{st}(F) \)) for the set of complete (resp. stable) extensions.

Given \( \sigma \in (\text{co}, \text{st}) \), we focus on solving the decision problems DC-\( \sigma \) (Decide Credulous acceptability) and DS-\( \sigma \) (Decide Skeptical acceptability), formally defined respectively by answering the question: "Given an AF \( F = (\mathcal{A}, \mathcal{R}) \) and an argument \( a \in \mathcal{A} \), is \( a \) a member of some (resp. each) \( \sigma \)-extension of \( F \)?"

We denote by \( \text{cred}_\sigma(F) \) (resp. \( \text{skep}_\sigma(F) \)) the set of arguments that are credulously (resp. skeptically) accepted, meaning they lead to a ‘YES’ response in the problem DC-\( \sigma \) (resp. DS-\( \sigma \)). Reasoning with AFs is typically challenging [11]. Specifically, the problems outlined earlier often reside within the first level of the polynomial hierarchy. To be more precise, both \( \text{cred}_\text{co}(F) \) and \( \text{cred}_\text{st}(F) \) pose NP-complete challenges, while \( \text{skep}_\text{co}(F) \) represents a coNP-complete problem. Notably, \( \text{skep}_\text{co}(F) \) can be resolved using a polynomial algorithm. This paper centers on improving the resolution of problems that necessitate NP oracles.

Despite theoretical complexity, real-world performance of abstract argumentation solvers remains impressive, even with substantial problem sizes [22]. These solvers typically address this challenge by translating the abstract argumentation problem into a Conjunctive Normal Form (CNF) formula using an updated encoding proposed by [8]. Specifically, a semantics \( \sigma \) and an argumentation framework \( (\mathcal{A}, F) \) are transformed into a propositional formula \( \phi_{\sigma, F} \) such that the formula’s models correspond to the extensions of the AF. These formulas represent conflict-free sets, admissible sets, and extensions under both complete and stable semantics. Details on these encodings and their use in prominent argumentation solvers can be found in [17, 20]. Once encoded, modern SAT solvers (see [9] for SAT details) efficiently resolve the problem [12, 17, 20].

While SAT solvers excel in handling various problems, challenges persist, particularly in dynamic AFs requiring updates. Regularly invoking a SAT solver during complex debates with evolving argumentation structures can lead to significant runtime costs, especially when resetting the solver after each modification [1, 2]. In this work, we advance this context using sophisticated SAT-based techniques to compute solutions for DC and DS problems (NP- and coNP-complete). Our approach relies on incremental SAT solving [13] and assumptions [6, 13, 16], enabling the use of prior computations to expedite acceptability determination following framework modifications. Empirical comparisons with other state-of-the-art methods in dynamic AFs, including a naive baseline solving the problem from scratch after each update, highlight the superior performance of our approach.

2 THE CRUSTABRI SOLVER

SAT solvers deduce clauses from the initial problem [9]. In dynamic argumentation, our goal is to retain the same SAT solver and its
learned clauses while adjusting the constraints to align with the evolving AF. If the clauses remains unchanged or becomes more refined, the previously learned clauses can assist in future decision-making. However, relaxing the clauses in the SAT solver makes it uncertain if the previously learned clauses are still consequences of the problem, except when new free variables are introduced.

To address this, assumption variables [13] are employed. Already used in static argumentation [17, 21], it encodes a relaxed version of the problem with additional (assumption) variables, allowing the representation of any set of constraints considered later. When assumption literals are set, the solver learns clauses as if these variables were determined by its internal decision-making process. Consequently, if a learned clause results from these artificial decisions, it becomes a consequence of the conjunction of the original formula and the assumption literals, enabling the utilization of learned clauses under assumptions in subsequent computations.

Employing assumptions in a dynamic context demands a meticulous formulation. If a state cannot be accurately encoded with a set of assumptions, it may necessitate a new encoding, rendering previous clauses obsolete. An excessively relaxed initial formula could also inflate the number of clauses and auxiliary variables for assumptions, introducing unnecessary computational overhead.

### Attacks as assumptions

In an AF with a constant set of arguments, a practical approach involves assuming \(s_{a,b}\) for each pair \((a, b)\) \(\in\mathcal{A}\). For instance, stable semantics requires \(|\mathcal{A}|^2\) assumptions and \(O(|\mathcal{A}|^2)\) clauses; \(\psi^s_{st}(F) = \bigwedge_{a \in \mathcal{A}} (a \lor \bigvee_{b \in \mathcal{A}} (s_{a,b} \land b)) \land \bigwedge_{a \in \mathcal{A}} (\lnot s_{a,b} \lor \lnot a \lor \lnot b)\). However, introducing a new argument weakens specific clauses, making this encoding unsuitable for dynamics. \(\mu\)-toksia reserves fresh variables for future arguments by temporally setting them to false using assumptions, with constraints determined by the current and reserved arguments.

### Disjunction of attackers as assumptions

We present two encodings that exploit the unique structure of stable and complete semantics in a dynamic context, connecting assumptions to the disjunction of attackers (\(P_a\) variables of [17]). They use a minimal number of active assumptions, limited by the current number of arguments. The count of active clauses matches the number of clauses in static encodings. Assumptions link sets of attacks targeting a common argument. When attackers change, the previous set is invalidated, emphasizing the insufficient benefits of adapting the encoding overhead.

#### \(\psi^D_{st}\) and \(\psi^D_{co}\)

\(\psi^D_{st}(F) = \bigwedge_{a \in \mathcal{A}} (\lnot s_{a} \lor \bigvee b \in \mathcal{A} (s_{a,b} \land b)) \land \bigwedge_{a \in \mathcal{A}} (\lnot s_{a,b} \lor \lnot a \lor \lnot b)\) (1.1)

\(\psi^D_{co}(F) = \bigwedge_{a \in \mathcal{A}} (\lnot s_{a} \lor \bigvee b \in \mathcal{A} (s_{a,b} \land P_a \land b)) \land \bigwedge_{a \in \mathcal{A}} (\lnot s_{a,b} \lor \lnot a \lor \lnot b)\) (1.2)

\(\psi_d(F) \equiv \psi^D_{st}(F) \land \bigwedge_{a \in \mathcal{A}} (\lnot s_{a}) \land \psi^D_{co}(F) \land \bigwedge_{a \in \mathcal{A}} (\lnot s_{a,b} \land P_a \land b)\) (2.1)

\(\psi_d(F) \equiv \psi^D_{st}(F) \land \bigwedge_{a \in \mathcal{A}} (\lnot s_{a}) \land \psi^D_{co}(F) \land \bigwedge_{a \in \mathcal{A}} (\lnot s_{a,b} \land P_a \land b)\) (2.2)

\(\psi_d(F) \equiv \psi^D_{st}(F) \land \bigwedge_{a \in \mathcal{A}} (\lnot s_{a}) \land \psi^D_{co}(F) \land \bigwedge_{a \in \mathcal{A}} (\lnot s_{a,b} \land P_a \land b)\) (2.3)

\(\psi_d(F) \equiv \psi^D_{st}(F) \land \bigwedge_{a \in \mathcal{A}} (\lnot s_{a}) \land \psi^D_{co}(F) \land \bigwedge_{a \in \mathcal{A}} (\lnot s_{a,b} \land P_a \land b)\) (2.4)

\(\psi_d(F) \equiv \psi^D_{st}(F) \land \bigwedge_{a \in \mathcal{A}} (\lnot s_{a}) \land \psi^D_{co}(F) \land \bigwedge_{a \in \mathcal{A}} (\lnot s_{a,b} \land P_a \land b)\) (2.5)

Replacing \(s_a\) with \(s'_{a}\) in the assumption set passed to the solver accounts for these changes. Adding a new argument \(a\) involves declaring \(a\) and \(s_a\) in the SAT solver and including \(s_a\) in the assumption set. Removing an argument entails eliminating all attacks involving it and setting its value with a unit clause. Concerning complete semantics, reencoding is necessary when a set of attacks change, following rules (2.2) to (2.5).

### EXPERIMENTAL EVALUATION

We compare our new approach (\(\text{dyn\_att\_disj}\)) with the dynamic attack assumptions method (\(\text{dyn\_att}\)) and a baseline that creates a new AF for every state evolution (static). These were implemented in CRUSTABRI [18] and evaluated by replaying the 5th ICCMA competition [14] on machines with Intel Xeon E5-2637 v4 processors and 128GB RAM (details at [15]). The \(\text{dyn\_att}\) encoding introduces a key parameter: the number of arguments reserved for future additions during reencoding. In \(\mu\)-toksia, this number is set to 2 (for \(n\) arguments at encoding time, \(n\) additional arguments are reserved). Recognizing its importance, we experimented with factors such as 2, 1.5, and 1.25. The results, presented in the following figure, categorize the approaches into three distinct groups based on PAR-2 scores (as in the competition).

[Graph showing experimental results]

Our approach, \(\text{dyn\_att\_disj}\), achieved the top position across all tracks, highlighting its effectiveness. The \(\text{static}\) approach secured second place, consistently demonstrating strong performance. The various iterations of \(\text{dyn\_att}\) approaches, using different factors, yielded comparable results in the third group. Interestingly, the impact of the factor in \(\text{dyn\_att}\) encoding was less substantial than anticipated. Regardless of the factor, the \(\text{static}\) approach consistently outperformed \(\text{dyn\_att}\), emphasizing the insufficient benefits to compensate for encoding overhead. Focusing on the number of instances solved led to the same conclusions. Our \(\text{dyn\_att\_disj}\) approach showcased remarkable ability in leveraging past knowledge, aligning with amortization in modern SAT solvers [7].

### 4 CONCLUSION

This paper explores incremental SAT solving in dynamic argumentation, focusing on efficiently determining argument statuses during updates. Our novel encoding method proves more space-efficient than existing approaches, requiring fewer assumptions. Empirical evaluations demonstrate its clear superiority over the naive baseline and the current state-of-the-art method.
REFERENCES


