Proceedings of the 22nd International Conference on Principles of Knowledge Representation and Reasoning
KR and Constraints

Reasoning in Assumption-Based Argumentation via SAT

Andreas Niskanen[1] , **Masood Feyzbakhsh Rankooh**[1] , **Tuomo Lehtonen**[2] , **Matti Järvisalo**[1]
[1] HIIT, Department of Computer Science, University of Helsinki, Finland
[2] HIIT, Department of Computer Science, Aalto University, Finland
{masood.feyzbakhshrankooh,andreas.niskanen,matti.jarvisalo}@helsinki.fi, tuomo.lehtonen@aalto.fi

Abstract

The dominant approaches for solving NP-hard reasoning problems in computational argumentation are declarative—namely, Boolean satisfiability (SAT) in the case of abstract argumentation and answer set programming (ASP) in the case of structured formalisms such as assumption-based argumentation (ABA). ASP is particularly suited for the commonly-studied logic programming variant of ABA as acyclic derivations in ABA can be naturally modelled in ASP. In this work, we develop and evaluate various alternative approaches to realizing SAT-based reasoning for ABA, motivated by the success of SAT solvers in the realm of abstract argumentation. In contrast to ASP, non-trivial encodings or extensions to SAT solvers are needed to efficiently handle the acyclicity constraint underlying ABA reasoning. We develop and evaluate both advanced encodings and user-defined propagation mechanisms for realizing efficient SAT-based reasoning in ABA. As a result, we provide a first SAT-based ABA reasoner that can outperform the current state-of-the-art ASP approach to ABA.

1 Introduction

The study of computational models of argumentation (Dunne and Wooldridge 2009; Atkinson et al. 2017; Bench-Capon and Dunne 2007; Baroni et al. 2018) is a vibrant area of knowledge representation and reasoning research, with the key aim of developing computational methodology for drawing conclusions from internally inconsistent or incomplete knowledge bases. In contrast to classical logic, formal argumentation represents defeasible reasoning, where new information may revoke previously reached conclusions.

In the realm of computational models of argumentation, major advances in practical algorithms and their implementations have been made for reasoning in abstract argumentation, with a particular focus on Dung's abstract argumentation frameworks (AFs) (Dung 1995) as the *de facto* abstract argumentation formalism. The dominating algorithmic approach to reasoning in AFs is based on encoding reasoning problems over AFs under the various argumentation semantics using declarative languages, in particular Boolean satisfiability (SAT) (Biere 2009) or answer set programming (ASP) (Gelfond and Lifschitz 1988; Niemelä 1999), and employing a state-of-the-art SAT (or ASP) solver (Biere et al. 2024; Gebser et al. 2016) on the encoding (Besnard and Doutre 2004; Dvořák et al. 2014; Lagniez, Lonca, and Mailly 2015; Niskanen and Järvisalo 2020; Jean-Marie Lagniez and Mailly 2023). This approach is also warranted from the theoretical perspective, as reasoning in argumentation formalisms is NP-hard for a great majority of central argumentation semantics (Dvořák and Dunne 2018). The use of NP machinery also allows for extensions to reasoning in AFs for problems that are expectedly harder than NP, such as skeptical reasoning under preferred semantics as one example, via iterative use of the solvers (Dvořák et al. 2014; Niskanen and Järvisalo 2020; Thimm, Cerutti, and Vallati 2021). The prevalence and success of SAT-based approaches to reasoning in abstract argumentation is highlighted by the results of the recent International Competitions of Computational Models of Argumentation (ICCMA) (Thimm and Villata 2017; Gaggl et al. 2020; Bistarelli et al. 2025; Järvisalo, Lehtonen, and Niskanen 2025), where SAT-based approaches have consistently reached top positions.

AFs hide the internal structure of arguments, focusing on modelling the relationships between arguments through pair-wise attacks. However, arguments in reality most often have an intrinsic structure. Computational models for structured argumentation enable making the internal structure of arguments explicit, as derivations from more basic structures. Various different structured formalisms, each with their own use-cases and complementary motivations, have been developed (Dung 1995; Bondarenko et al. 1997; García and Simari 2004; García and Simari 2014; Besnard and Hunter 2018; Besnard et al. 2014; Prakken 2010; Bao, Čyras, and Toni 2017; Brewka, Polberg, and Woltran 2014; Brewka and Woltran 2010). Among the most studied structured formalisms is assumption-based argumentation (ABA) (Bondarenko et al. 1997; Čyras et al. 2018), on which we focus here. Specifically we consider the commonly-studied logic programming fragment of ABA (Bondarenko et al. 1997). In ABA, attacks are defined between sets of assumptions: if a set of assumptions derives the contrary of an assumption a, the set of assumptions attacks a and all assumption sets containing a. Whereas in abstract argumentation sets of arguments are jointly accepted according to semantics, in ABA the semantics sanction acceptable sets of assumptions.

Compared to AFs, developing practical algorithms for

reasoning in ABA is arguably more challenging due to the intrinsically more involved formalism capable of modelling the internal structure of arguments and derivations of attacks between arguments. Regardless, there has been significant recent interest in developing practical algorithmic approaches for structured formalisms (Cerutti et al. 2018; Borg and Odekerken 2022; Calegari et al. 2022; Lehtonen, Wallner, and Järvisalo 2022; Lehtonen et al. 2024a), and in particular for ABA (Craven and Toni 2016; Bao, Čyras, and Toni 2017; Lehtonen, Wallner, and Järvisalo 2017; Lehtonen, Wallner, and Järvisalo 2021a; Diller, Gaggl, and Gorczyca 2021; Diller, Gaggl, and Gorczyca 2022; Popescu and Wallner 2023; Lehtonen et al. 2023; Lehtonen et al. 2024b). In the 2023 instantiation of ICCMA (Järvisalo, Lehtonen, and Niskanen 2025), a special track focusing on ABA reasoning took place, corroborating the progress in practical approaches to reasoning in ABA. While reasoning in ABA can be translated to reasoning in AFs (Dung, Mancarella, and Toni 2007), thereby allowing for taking advantage of state-of-the-art SAT-based AF reasoners, the translation itself causes a significant scalability bottleneck due to worst-case exponential blow-up (Strass, Wyner, and Diller 2019). This underlines the need for algorithmic approaches reasoning natively on the level of ABA. Among such approaches, the current dominant approach is based on encoding reasoning in ABA using ASP (Lehtonen, Wallner, and Järvisalo 2021a; Lehtonen, Wallner, and Järvisalo 2021b), avoiding explicit construction of arguments by reasoning on the level of assumption sets. This approach dominated the ABA track of ICCMA 2023 (Järvisalo, Lehtonen, and Niskanen 2025).

While SAT solvers, reasoning under classical propositional semantics on propositional encodings, are the dominant approach to AF reasoning, for structured argumentation formalisms currently purely SAT-based approaches are essentially non-existing. To directly capture reasoning in ABA using declarative encodings requires in particular modelling *acyclic* derivation of the underlying attack structure from assumptions from the given rule-based knowledge base. This corresponds to the challenge of modelling acyclicity over potentially significantly long derivation chains. Answer set semantics of ASP natively enforce the required acyclicity, and is therefore especially suited for this task as the derivation length does not need to be explicitly encoded in ASP, thereby avoiding, at least on the representational level, a blow-up caused by having to explicitly represent the length of derivations in an encoding. However, the dominance of the SAT-based approach to AF reasoning also motivates further study of more intricate SAT-based methodologies for more efficiently reasoning in ABA.

In this work, we develop and evaluate various alternative approaches to realizing SAT-based reasoning for ABA. To our best understanding, our work constitutes the first to present direct SAT encodings and SAT-based solving techniques for ABA. In terms of reasoning tasks, our main focus is on credulous and skeptical acceptance under various central argumentation semantics (stable, complete, preferred and ideal) for those reasoning modes for which the decision problems are NP-hard. The SAT-based approaches come in two types in terms of how the acyclicity constraint for ABA is enforced in conjunction with a SAT encoding of the semantics (assuming the constraint): we consider both (i) different ways of fully encoding the acyclicity constraint with propositional clauses as part of the SAT encoding, and (ii) two ways of handling the acyclicity constraint via a user-defined propagator, extending the standard propagation mechanism within a SAT solver using the recent IPASIR-UP SAT solver interface (Fazekas et al. 2024). For (i), in addition to the straightforward encoding, we investigate the applicability of a recently proposed compact vertex elimination encoding (Rankooh and Rintanen 2022) based on a heuristically computed tree decomposition, with the benefit of allowing the SAT solver to directly reason and learn on the propositional level also wrt. the acyclicity constraint. For (ii), we investigate both a SAT modulo acyclicity approach (Gebser, Janhunen, and Rintanen 2014) and an unfounded set propagation (UFS) mechanism similar to state-of-the-art ASP solving techniques (Gebser, Kaufmann, and Schaub 2012) but directly implemented in a modern SAT solver. Both approaches are also applicable as the abstraction solver within counterexample-guided abstraction refinement approaches for beyond-NP reasoning tasks such as skeptical reasoning under preferred semantics and computing the ideal extension. We provide open-source implementations of all of the approaches, and rigorously empirically evaluate the relative performance with the dominant ASP-based ASPFORABA system on different types of ABA benchmarks. The results show that, for the first time, purely SAT-based approaches are made competitive with and even outperform the dominant ASP-based approach to reasoning in ABA.

2 Preliminaries

We start with necessary preliminaries on ABA and SAT.

2.1 Assumption-Based Argumentation

We recall assumption-based argumentation (ABA) (Bondarenko et al. 1997; Toni 2014; Čyras et al. 2018). We assume a deductive system $(\mathcal{L}, \mathcal{R})$, where $\mathcal{L}$ is a formal language, i.e., a set of sentences, and $\mathcal{R}$ a set of inference rules over $\mathcal{L}$. A rule $r \in \mathcal{R}$ has the form $a_0 \leftarrow a_1, \ldots, a_n$ with $a_i \in \mathcal{L}$. We denote the head of rule r by $head(r) = \{a_0\}$ and the (possibly empty) body of r with $body(r) = \{a_1, \ldots, a_n\}$. An ABA framework (ABF) is composed of a deductive system and information on which sentences are provisionally assumed and which sentences are contrary to assumptions (inducing a conflict).

Definition 1. *An ABA framework is a tuple $F = (\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-)$, where $(\mathcal{L}, \mathcal{R})$ is a deductive system, $\mathcal{A} \subseteq \mathcal{L}$ a non-empty set of assumptions, and $^-$ a function mapping assumptions $\mathcal{A}$ to sentences $\mathcal{L}$.*

We focus on the commonly-studied logic programming instantiation of ABA (Bondarenko et al. 1997; Čyras et al. 2018), i.e., all sets defining ABFs are finite, the elements in $\mathcal{L}$ are atoms, all rules are finite and explicitly given as a list of body literals, and no assumptions occur as a head of a rule. The last condition implies flatness of the ABA frameworks (Bondarenko et al. 1997).

There are two equivalent definitions for derivations in ABA: forward derivations and tree-derivations (Dung, Mancarella, and Toni 2007). An atom $a \in \mathcal{L}$ is *forward-derivable* from a set $X \subseteq \mathcal{A}$ in ABF $F = (\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-)$, denoted by $X \vdash a$, if $a \in X$ or there is a sequence of rules $(r_1, \ldots, r_n)$ such that $head(r_n) = a$ and for each rule r_i we have $r_i \in \mathcal{R}$ and each atom in the body of r_i is derived from rules earlier in the sequence or in X, i.e., $body(r_i) \subseteq X \cup \bigcup_{j<i} head(r_j)$. It can clearly be decided in polynomial time whether a given $X \subseteq \mathcal{A}$ derives a given $x \in \mathcal{L}$. On the other hand, an atom $a \in \mathcal{L}$ is *tree-derivable* from a set $X \subseteq \mathcal{A}$ and rules $R \subseteq \mathcal{R}$, denoted by $X \models_R s$, if either $X = \{s\}$ and $\mathcal{R} = \emptyset$, or there is a finite tree T with the root labeled with s, and the set of labels for the leaves is X (with the possible addition of $\top$) and for each node that is not a leaf, labelled with $s' \in \mathcal{L}$, there is a rule $r \in R$ with s' as the head and the children of the node labeled with exactly the body elements of r. The symbol "$\top$" represents an empty rule body, signifying that the head of this rule is tree-derivable from the empty set.

Example 1. *Consider the following ABF with eight atoms in $\mathcal{L}$, four of which are assumptions ($\mathcal{A}$), and four rules in $\mathcal{R}$. In this framework, the assumption a has no derivable contrary, and thus the contrary of a is left unmentioned.*

$$\begin{aligned} \text{atoms } \mathcal{L} &= \{a, b, c, d, w, x, y, z, p, q\} \\ \text{assumptions } \mathcal{A} &= \{a, b, c, d\} \\ \text{contraries} \quad & \overline{b} = x,\ \overline{c} = y,\ \overline{d} = z \\ \text{rules } \mathcal{R} &= \{(w \leftarrow a), (y \leftarrow b, w), (x \leftarrow c), \\ & \quad (z \leftarrow a, b), (p \leftarrow q), (q \leftarrow p)\} \end{aligned}$$

Examples of derivations in this ABF are $\{a\} \vdash w, \{a, b\} \vdash y, and \{a, b, c, d\} \vdash w$. Atoms p and q depend on each other and there is no independent way to derive them. Thus no assumption set derives either.

The deductive closure for an assumption set X wrt. rules $\mathcal{R}$ is given by $Th_{\mathcal{R}}(X) = \{a \in \mathcal{L} \mid X \vdash_{\mathcal{R}} a\}$. We move on to the definitions of attacks between assumption sets and the various semantics of ABA.

Definition 2. *Let $F = (\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-)$ be an ABA framework, and $A, B \subseteq \mathcal{A}$ be two sets of assumptions. Assumption set A attacks assumption set B in F if $A' \vdash \overline{b}$ for some $A' \subseteq A$, $R \subseteq \mathcal{R}$, and $b \in B$.*

In other words, set A attacks B if one can derive the contrary of an assumption in B via the given deductive system. Semantics in ABA define acceptable sets of assumptions, and are based on the concepts of conflict-freeness and defense.

Definition 3. *Let $F = (\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-)$ be an ABA framework. An assumption set $A \subseteq \mathcal{A}$ is conflict-free in F if A does not attack itself. Set A defends assumption set $B \subseteq \mathcal{A}$ in F if for all $C \subseteq \mathcal{A}$ that attack B it holds that A attacks C.*

We are now ready to recall the various semantics considered in this work.

Definition 4. *Let $F = (\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-)$ be an ABA framework. Further, let $A \subseteq \mathcal{A}$ be a conflict-free set of assumptions in F. In F, set A is*

- admissible *if A defends itself;*
- complete *if A is admissible and contains every assumption set defended by A;*
- preferred *if A is admissible and there is no admissible set of assumptions B with $A \subset B$;*
- stable *if each $\{x\} \subseteq \mathcal{A} \setminus A$ is attacked by A; and*
- ideal *if A is the $\subseteq$-maximal admissible assumption set within the intersection of all preferred assumption sets.*

A σ-assumption set is an assumption set under a semantics $\sigma \in \{adm, com, stb, prf, ideal\}$, i.e., admissible, complete, stable, preferred, and ideal assumption set, respectively.

Example 2. *Continuing Example 1, consider the assumption set $\{a, b\}$, which derives w, y, and z in addition to a and b. The atoms y and z are contrary to assumptions c and d respectively. Thus $\{a, b\}$ attacks every assumption set containing either c or d. The set $\{a, b\}$ is attacked by any assumption set containing the assumption c, since $\{c\}$) derives x and $\overline{b} = x$ and no other set. As $\{a, b\}$ attacks every set containing c, $\{a, b\}$ is admissible. Moreover, $\{a, b\}$ is complete since any assumption set $A \subseteq \mathcal{A}$ not contained in $\{a, b\}$ (namely $\{c\}$ and $\{d\}$) is not defended by $\{a, b\}$. Any proper superset A of $\{a, b\}$: then either $c \in A$ or $d \in A$, which violates conflict-freeness. Thus, no proper superset is admissible and $\{a, b\}$ is preferred. Finally, $\{a, b\}$ is stable since it attacks all other assumptions. Further, the framework has three complete assumption sets: $\{a\}$, $\{a, b\}$, and $\{a, c, d\}$. The latter two complete assumption sets are also preferred and stable assumption sets. As $\{a\}$ is the intersection of the preferred assumptions sets and is also admissible, it is the ideal assumption set.*

A central decision problem in reasoning with ABA is to find out whether a given atom is acceptable under a semantics. Two prominent reasoning modes are credulous and skeptical acceptance of atoms in an ABA framework.

Definition 5. *Let $F = (\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-)$ be an ABA framework and σ be a semantics. A given atom $s \in \mathcal{L}$ is*

- *credulously accepted in F under semantics σ if there is a σ-assumption set A such that $s \in Th_{\mathcal{R}}(A)$; and*
- *skeptically accepted in F under semantics σ if $s \in Th_{\mathcal{R}}(A)$ for all σ-assumption sets A.*

In flat ABA there is a unique ideal assumption set, implying coincidence of credulous and skeptical reasoning for both of these semantics. Credulous reasoning under admissible, complete, and preferred semantics coincide (Bondarenko et al. 1997; Čyras et al. 2018).

Example 3. *Continuing Example 2, every atom except p and q is credulously accepted under admissible semantics (and therefore also under complete and preferred semantics). No atom is skeptically accepted under admissible semantics (only atoms derivable from the empty set are skeptically accepted under admissible semantics). The atoms a and w are skeptically accepted under complete, preferred, and stable semantics.*

The complexity of reasoning in ABA is well known (Dimopoulos, Nebel, and Toni 2002; Dunne 2009; Lehtonen, Wallner, and Järvisalo 2021a; Cyras, Heinrich, and Toni 2021). Credulous acceptance is NP-complete under admissible, complete, preferred and stable semantics. Skeptical acceptence is in P under admissible and complete semantics, Π^{P}_2-complete under preferred semantics, and coNP-complete under stable semantics. Both acceptance problems are in Θ^{P}_2 under ideal semantics.

2.2 Boolean Satisfiability (SAT)

For a Boolean variable x there are two literals, x and $\neg x$. A clause C is a disjunction ($\vee$) of literals. A conjunctive normal form (CNF) formula F is a conjunction ($\wedge$) of clauses. For convenience we view clauses as sets of literals and formulas as sets of clauses. We denote by $V(F)$ and $L(F)$ the set of variables and literals of F, respectively. A truth assignment $\tau\colon V(F) \to \{0,1\}$ maps each variable to 0 (false) or 1 (true), and is extended to literals via $\tau(\neg x) = 1 - \tau(x)$, to clauses via $\tau(C) = \max\{\tau(l) \mid l \in C\}$, and to formulas via $\tau(F) = \min\{\tau(C) \mid C \in F\}$. We interchangeably represent truth assignments τ as sets of non-contradictory literals: $\{l \in L(F) \mid \tau(l) = 1\}$. The *Boolean satisfiability* problem (SAT) (Biere 2009) asks for an input formula F whether there is an assignment τ with $\tau(F) = 1$. In the positive case F is satisfiable, and otherwise F is unsatisfiable.

In this work, we make use of IPASIR-UP (Fazekas et al. 2024), a generic interface for implementing user propagators in SAT solvers. Briefly, IPASIR-UP allows to define custom propagators via the following functionality. Under the current CNF formula F, a set $O \subseteq V(F)$ of variables is marked as *observed*. The solver *notifies* the user propagator about all assignments (decisions, unit propagations, and learned unit clauses) to these variables. In addition, the solver informs the propagator whenever the *decision level* is increased, and whenever the solver *backtracks* to a lower decision level. During unit propagation, the user propagator can inform the solver about any *additional propagations*. If such a propagation is relevant for conflict analysis, the solver also asks for the corresponding *reason clause*. Finally, after unit propagation the solver also checks whether the user propagator can provide *external clauses* to the solver.

3 SAT Encodings for ABA

Let $F = (\mathcal{L}, \mathcal{R}, \mathcal{A}, {}^-)$ be an ABA framework and $\sigma \in \{adm, com, stb\}$. Our goal is to encode a σ-assumption set of F. To this end, we declare Boolean variables x_a, y_a, and z_a for each $a \in \mathcal{A}$. For a satisfying truth assignment τ to the encoding described next, the set $A_\tau = \{a \in \mathcal{A} \mid \tau(x_a) = 1\}$ corresponds to a σ-assumption set under $\sigma \in \{adm, com, stb\}$. Likewise, for $a \in \mathcal{A}$, $\tau(y_a) = 1$ iff A_τ attacks $\{a\}$, and $\tau(z_a) = 1$ iff

$$B_\tau = A_\tau \cup \{a \in \mathcal{A} \mid \tau(y_a) = 0\}$$

attacks $\{a\}$, that is, $\{a\}$ is not defended by A_τ since it is attacked by an undefeated argument. Given these intended interpretations, we define

$$\varphi_{cf}(F) = \bigwedge_{a \in \mathcal{A}} (x_a \to \neg y_a)\,,$$
$$\varphi_{adm}(F) = \varphi_{cf}(F) \wedge \bigwedge_{a \in \mathcal{A}} (x_a \to \neg z_a)\,,$$
$$\varphi_{com}(F) = \varphi_{adm}(F) \wedge \bigwedge_{a \in \mathcal{A}} (\neg z_a \to x_a)\,,$$
$$\varphi_{stb}(F) = \varphi_{cf}(F) \wedge \bigwedge_{a \in \mathcal{A}} (\neg x_a \to y_a)\,.$$

To enforce the intended meaning of the x_a, y_a and z_a variables, we need to in particular capture the derivation of atoms from sets of assumptions. This necessitates an acyclicity constraint to prohibit atoms cyclically supporting each other. Consider a natural way of attempting to encode in SAT that an atom x is derivable from S: x is derivable from S iff there is an "activated" rule for x, i.e., a rule r such that each $b \in body(r)$ either is an assumption in S or is derived with another activated rule. To see the necessity for further constraints, consider the rules $(x \leftarrow y)$ and $(y \leftarrow x)$ where $x, y \in \mathcal{L} \setminus \mathcal{A}$. A satisfying assignment is allowed to activate both rules and derive both x and y due to cyclicity even though there are no assumptions supporting this.

3.1 Level-based Encoding of Acyclic Derivations

A natural way to rule out cyclic derivations in a SAT encoding is a "level-based" encoding, explicitly encoding sequences of rule applications from a set of assumptions by enforcing that an atom x can be derived on level i only if all body elements of a rule for x are derived on level $i-1$. To encode all derived atoms, we assume an upper bound U on the maximum length of a derivation. For (flat) ABA, we may set $U = |\mathcal{L}| - |\mathcal{A}|$. For each non-assumption $s \in \mathcal{L} \setminus \mathcal{A}$ and each $i = 1, \ldots, U$, we declare variables d^i_s and e^i_s, with the intended meaning that $\tau(d^i_s) = 1$ iff s is derived from A_τ at the ith iteration, and $\tau(e^i_s) = 1$ iff s is derived from B_τ at the ith iteration. Now, let

$$\varphi^{lvl}_{att}(F) = \bigwedge_{a \in \mathcal{A}} \left(y_a \leftrightarrow \psi^{lvl}_{att}(a)\right)$$

where $\psi^{lvl}_{att}(a)$ is $x_{\overline{a}}$ if the contrary $\overline{a}$ is an assumption, and $\bigvee_{i=1}^{U} d^i_{\overline{a}}$ otherwise. This encodes that $\{a\}$ is attacked by A_τ iff its contrary $\overline{a}$ is derivable. Similarly,

$$\varphi^{lvl}_{ndef}(F) = \bigwedge_{a \in \mathcal{A}} \left(z_a \leftrightarrow \psi^{lvl}_{ndef}(a)\right),$$

where $\psi^{lvl}_{ndef}(a)$ is $\neg y_{\overline{a}}$ if the contrary $\overline{a}$ is an assumption and $\bigvee_{i=1}^{U} e^i_{\overline{a}}$ otherwise, states that $\{a\}$ is not defended by A_τ iff its contrary $\overline{a}$ is derivable from B_τ.

For each $s \in \mathcal{L} \setminus \mathcal{A}$ and $i = 1, \ldots, U$, we define $\varphi_A(s, i)$ as

$$d^i_s \leftrightarrow \bigvee_{\substack{r \in \mathcal{R} \\ head(r) = s}} \left(\bigwedge_{\substack{a \in body(r) \\ a \in \mathcal{A}}} x_a \wedge \bigwedge_{\substack{a \in body(r) \\ a \in \mathcal{L} \setminus \mathcal{A}}} d^{i-1}_a \right),$$

where we set $d_s^0 = 0$ for all $s \in \mathcal{L} \setminus \mathcal{A}$. This formula encodes that s is derived from A_τ at iteration i iff there is a rule $r \in \mathcal{R}$ with s as its head such that, for all $a \in body(r)$, x_a is set to true for an assumption, and d_a^{i-1} is set to true for a non-assumption. Similarly, the formula $\varphi_B(s,i)$ defined as

$$e_s^i \leftrightarrow \bigvee_{\substack{r \in \mathcal{R} \\ head(r)=s}} \left(\bigwedge_{\substack{a \in body(r) \\ a \in \mathcal{A}}} \neg y_a \wedge \bigwedge_{\substack{a \in body(r) \\ a \in \mathcal{L} \setminus \mathcal{A}}} e_a^{i-1} \right)$$

states the same for derivations from B_τ.

Proposition 1. *For $\sigma \in \{adm, com, stb\}$, each assignment τ satisfying $\Phi_\sigma^{lvl}(F) = \varphi_\sigma(F) \wedge \varphi_{att}^{lvl}(F) \wedge \varphi_{ndef}^{lvl}(F) \wedge \bigwedge_{s \in \mathcal{L}} \bigwedge_{i=1}^{U} \varphi_A(s,i) \wedge \bigwedge_{s \in \mathcal{L}} \bigwedge_{i=1}^{U} \varphi_B(s,i)$ corresponds to a σ-extension A_τ. Vice versa, each σ-extension A extends to a satisfying assignment τ of $\Phi_\sigma^{lvl}(F)$ by setting $\tau(x_a) = 1$ iff $a \in A$.*

Definitions of $\varphi_{ndef}(F)$ and $\varphi_B(s,i)$ in $\Phi_\sigma^{lvl}(F)$ are not needed for stable semantics as every assumption outside A_τ is attacked.

3.2 Graph-based Acyclicity

Another way to enforce acyclicity of derivations is to enforce that the graph corresponding to activated rules is acyclic. This allows for encoding the existence of acyclic tree derivations. Derivations can be captured by checking the existence of an acyclic subgraph of the graph corresponding to the rules of an ABF.

Proposition 2. *Given an ABF $F = (\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-)$ and set of assumptions $S \subseteq \mathcal{A}$. Let $G = (V, E)$ be a graph with $V = \mathcal{L} \setminus \mathcal{A}$ and $E = \{(s,t) \mid r \in \mathcal{R}, s = head(r), t \in body(r)\}$. An atom $x \in \mathcal{L}$ is derivable from a set of assumptions $S \subseteq \mathcal{A}$ iff there is subgraph $G_S^x = (V', E')$ of G such that (i) $x \in V'$, (ii) G_S^x is acyclic, and (iii) $\forall s \in \mathcal{L}$: $s \in V'$ iff either $s \in S$ or there is a rule $r \in \mathcal{R}$ with $head(r) = s$ and $\forall t \in body(r) : t \in V'$ and $(s,t) \in E'$.*

We describe an encoding of acyclic derivations based on Proposition 2, thereby avoiding explicitly expressing an indexed sequence of rule applications. This encoding requires an additional way of enforcing the acyclicity of a graph. We later discuss options for this: explicitly encoding acyclicity in SAT (with vertex elimination), and using specialized user propagators for SAT.

Similarly as in the level-based encoding, we use the x_a, y_a, and z_a variables for each assumption $a \in \mathcal{A}$, as well as the $\varphi_\sigma(F)$ formulas encoding the semantics. We declare variables d_a and e_a for each non-assumption $a \in \mathcal{L} \setminus \mathcal{A}$, and define

$$\varphi_{att}(F) = \bigwedge_{a \in \mathcal{A}} (y_a \leftrightarrow \psi_{att}(a)),$$

where $\psi_{att}(a) = x_{\overline{a}}$ if $\overline{a}$ is an assumption, and $\psi_{att}(a) = d_{\overline{a}}$ otherwise, as well as

$$\varphi_{ndef}(F) = \bigwedge_{a \in \mathcal{A}} (z_a \leftrightarrow \psi_{ndef}(a)),$$

where $\psi_{ndef}(a) = \neg y_{\overline{a}}$ if $\overline{a}$ is an assumption, and $\psi_{ndef}(a) = e_{\overline{a}}$ otherwise. These formulas state that $\{a\}$ is attacked by A_τ and B_τ, respectively. In addition, we declare variables $r_{s,t}^d$ and $r_{s,t}^e$ for each $s, t \in \mathcal{L} \setminus \mathcal{A}$ for which there exists a rule $r \in \mathcal{R}$ with $head(r) = s$ and $t \in body(r)$. These variables will be used to enforce that derivations must be acyclic. Now, for $s \in \mathcal{L} \setminus \mathcal{A}$ the formula $\varphi_A^{acyc}(s)$ defined as

$$\left(\bigvee_{\substack{r \in \mathcal{R} \\ head(r)=s}} \left(\bigwedge_{\substack{a \in body(r) \\ a \in \mathcal{A}}} x_a \wedge \bigwedge_{\substack{a \in body(r) \\ a \in \mathcal{L} \setminus \mathcal{A}}} d_a \right) \rightarrow d_s \right) \wedge$$

$$\left(d_s \rightarrow \bigvee_{\substack{r \in \mathcal{R} \\ head(r)=s}} \left(\bigwedge_{\substack{a \in body(r) \\ a \in \mathcal{A}}} x_a \wedge \bigwedge_{\substack{a \in body(r) \\ a \in \mathcal{L} \setminus \mathcal{A}}} (d_a \wedge r_{a,s}^d) \right) \right)$$

states that, if there exists a rule with s as its head so that its body is satisfied, then s is derived from A_τ. Further, if s is derived, we require in addition all variables $r_{a,s}^d$ for $a \in body(r) \cap (\mathcal{L} \setminus \mathcal{A})$ to be true. We state the same for derivations from B_τ via $\varphi_B^{acyc}(s)$ defined as

$$\left(\bigvee_{\substack{r \in \mathcal{R} \\ head(r)=s}} \left(\bigwedge_{\substack{a \in body(r) \\ a \in \mathcal{A}}} x_a \wedge \bigwedge_{\substack{a \in body(r) \\ a \in \mathcal{L} \setminus \mathcal{A}}} e_a \right) \rightarrow e_s \right) \wedge$$

$$\left(e_s \rightarrow \bigvee_{\substack{r \in \mathcal{R} \\ head(r)=s}} \left(\bigwedge_{\substack{a \in body(r) \\ a \in \mathcal{A}}} x_a \wedge \bigwedge_{\substack{a \in body(r) \\ a \in \mathcal{L} \setminus \mathcal{A}}} (e_a \wedge r_{a,s}^e) \right) \right).$$

For a satisfying assignment τ, we consider two directed graphs $G_\tau^d = (\mathcal{L} \setminus \mathcal{A}, E_\tau^d)$ and $G_\tau^e = (\mathcal{L} \setminus \mathcal{A}, E_\tau^e)$, where

$$E_\tau^d = \{(s,t) \mid s, t \in \mathcal{L} \setminus \mathcal{A}, \tau(r_{s,t}^d) = 1\},$$
$$E_\tau^e = \{(s,t) \mid s, t \in \mathcal{L} \setminus \mathcal{A}, \tau(r_{s,t}^e) = 1\}.$$

If we ensure that these graphs are acyclic, satisfying truth assignments correspond to σ-extensions and vice versa. To limit redundancy, we further add

$$\varphi_G = \bigwedge_{\substack{s,t \in \mathcal{L} \setminus \mathcal{A} \\ \exists r \in \mathcal{R} :\, head(r)=s, t \in body(r)}} (r_{s,t}^d \rightarrow r_{s,t}^e)$$

which ensures that G_τ^d is a subgraph of G_τ^e, intuitively meaning that the derivations from $A\tau$ are reused for $B\tau$ (note that $A\tau \subseteq B\tau$). For $\sigma = stb$, we set $G_\tau^\sigma = G_\tau^d$, and for $\sigma \in \{adm, com\}$, we set $G_\tau^\sigma = G_\tau^e$.

Proposition 3. *For any $\sigma \in \{adm, com, stb\}$, each assignment τ satisfying $\Phi_\sigma^{acyc}(F) = \varphi_\sigma(F) \wedge \varphi_{att}(F) \wedge \varphi_{ndef}(F) \wedge \bigwedge_{s \in \mathcal{L} \setminus \mathcal{A}} \varphi_A^{acyc}(s) \wedge \bigwedge_{s \in \mathcal{L} \setminus \mathcal{A}} \varphi_B^{acyc}(s)$, for which the graph G_τ^σ is acyclic, corresponds to a σ-extension A_τ. Likewise, a σ-extension A extends to a satisfying assignment τ of $\Phi_\sigma^{acyc}(F)$ for which G_τ^σ is acyclic by setting $\tau(x_a) = 1$ iff $a \in A$.*

Note that the formulas $\varphi_{ndef}(F)$ and $\varphi_B^{acyc}(s)$ are not required for stable semantics.

Graph Acyclicity by Vertex Elimination. A compact approach to encoding acyclicity of the graphs G_τ^d and G_τ^e is based on an encoding simulating vertex elimination. The

concept of vertex elimination, originally introduced in the context of sparse Gaussian elimination by Rose, Tarjan, and Lueker (1976), was adapted to propositional acyclicity encodings by Rankooh and Rintanen (2022).

Let Φ denote any SAT with acyclicity encoding in the form of $\Phi_\sigma^{acyc}(F)$, $\sigma \in \{stb, adm, com\}$ explained above. Formula Φ can be regarded as a formula with an underlying directed graph $G = (\mathcal{L} \setminus \mathcal{A}, E)$, where E is the set of edges from s to t iff there exists a rule $r \in \mathcal{R}$ with $head(r) = s$ and $t \in body(r)$. The graph G_τ^σ is a subgraph of G, for which the edge (s,t) is present if $\tau(r_{s,t}^d) = 1$ for $\sigma = stb$, and $\tau(r_{s,t}^e) = 1$ for $\sigma \in \{adm, com\}$. We say that τ is acyclic if G_τ^σ is acyclic. We want to find Φ' such that τ is a satisfying assignment for $\Phi \wedge \Phi'$ iff τ is an acyclic satisfying assignment of Φ. Concretely, for each $v \in \mathcal{L} \setminus \mathcal{A}$, define the fill-in

$$F(v) = \big\{\langle x, y\rangle \mid \langle x, v\rangle \in E,\ \langle v, y\rangle \in E,\ x \neq y\big\},$$

and eliminate v to produce $G(v)$ by

$$G(v) = \big((\mathcal{L} \setminus \mathcal{A}) \setminus \{v\},\ E(v) \cup F(v)\big),$$
$$E(v) = \{\langle x, y\rangle \in E \mid x \neq v \neq y\}.$$

Let $\alpha\colon \{1, \ldots, n\} \to \mathcal{L}$ be an elimination order. Starting from $G_0 = G$, we iterate for $i = 1, \ldots, n-1$ and eliminate the vertex $\alpha(i)$ from G_{i-1}. This yields the graph sequence $G = G_0, G_1, \ldots, G_{n-1}$ with cumulative fill-in

$$F_\alpha(G) = \bigcup_{i=1}^{n-1} F_{i-1}\big(\alpha(i)\big),$$

and the vertex-elimination graph $G_\alpha^* = \big(\mathcal{L},\ E \cup F_\alpha(G)\big)$. An important property of G_α^* is that, regardless of ordering α, a cycle in G induces a 2-cycle in G_α^*.

Based on this property, we construct Φ' so that it simulates vertex elimination for any satisfying assignment of Φ. We define atoms $e_{s,t}$ to represent the edges of G_α^*, $e_{s,t}^*$. To reduce symmetry, we consider $e_{s,t}$ in our encoding only if $\alpha(s) \leq \alpha(t)$. To do this, we use symbol $e_{s,t}^*$ to denote $e_{s,t}$ if $\alpha(s) \leq \alpha(t)$, and $\neg e_{s,t}$ if $\alpha(s) > \alpha(t)$. For $\sigma = stb$, we add the constraint $r_{s,t}^d \rightarrow e_{s,t}^*$ to enforce the edges of G to be also edges of G^*; for $\sigma \in \{adm, com\}$ we similarly add the constraint $r_{s,t}^e \rightarrow e_{s,t}^*$. Further, for each $i = 1, \ldots, n-1$ and $\langle s, t\rangle \in F_{i-1}(\alpha(i))$, we include $e_{s,\alpha(i)}^* \wedge e_{\alpha(i),t}^* \rightarrow e_{s,t}^*$ to simulate vertex elimination.

With Φ denoting $\Phi_\sigma^{acyc}(F)$ for $\sigma \in \{stb, adm, com\}$, and Φ' as constructed above, the soundness and completeness of vertex elimination based encoding of acyclicity (Rankooh and Rintanen 2022) guarantee that a satisfying assignment to $\Phi \wedge \Phi'$ corresponds to a σ-extension, and vice versa.

4 SAT with User Propagators for ABA

We turn to an alternative approach to directly enforcing acyclicity in the SAT encoding: via implementing specialized user propagators (Gebser, Janhunen, and Rintanen 2014) in a SAT solver. Namely we propose to use a graph-based acyclicity propagator or an unfounded set based propagator making use of the recent IPASIR-UP interface.

4.1 Graph-based Acyclicity Propagator

The solver is initialized with the formula $\Phi_\sigma^{acyc}(F)$ (recall Section 3.2). For $\sigma = stb$, all $r_{s,t}^d$ variables are set as observed, and for $\sigma \in \{adm, com\}$, the $r_{s,t}^e$ variables are set as observed. The acyclicity propagator then keeps track of the corresponding graph, where all edges (s,t) are labeled either *possible*, *enabled*, or *disabled*. When an observed variable is set to true (resp. false), the corresponding edge is enabled (resp. disabled). We also record its decision level. Upon backtracking, all enabled and disabled edges with decision level higher than the new level are set back to *possible*.

For propagating, we check every edge (s,t) which has been enabled since the last time the propagator was called. First, we perform DFS starting from the node t following every enabled edge in the graph. If the node s is reached, we have found a cycle C, and inform the solver about the external clause $\bigvee_{(u,v)\in C} \neg r_{u,v}^d$ (resp. $\bigvee_{(u,v)\in C} \neg r_{u,v}^e$). Otherwise, we perform DFS starting from node s, following enabled edges in reverse. Letting N_1 denote the set of nodes which can be reached from t by the first DFS, N_2 denote the set of nodes which can be reversely reached from s by the second DFS, and P be the set of edges labeled possible, we propagate $\neg r_{u,v}$ for every $(u,v) \in P \cap (N_1 \times N_2)$. The corresponding reason clause is $\bigvee_{(u',v')\in C'} \neg r_{u',v'}^d$ (resp. $\bigvee_{(u',v')\in C'} \neg r_{u',v'}^e$) where C' is the "almost-cycle" formed by edges used to reach u from t and v from s.

4.2 Unfounded Set based Propagator

Motivated by the success of ASP for ABA, we investigate using *unfounded set propagation*, a core technique used in state-of-the-art ASP solvers to determine non-circular supports for atoms (Gebser, Kaufmann, and Schaub 2012), for SAT-based ABA reasoning. Our implementation of unfounded set propagation follows the source pointer based approach of CLASP, which we implement using IPASIR-UP. The solver is initialized with a simpler encoding which, intuitively, encodes both sides of the equivalence for derivations in a similar manner. Namely, for $s \in \mathcal{L} \setminus \mathcal{A}$,

$$\varphi_A^{ufs}(s) = d_s \leftrightarrow \bigvee_{\substack{r\in\mathcal{R}\\ head(r)=s}} \left(\bigwedge_{\substack{a\in body(r)\\ a\in\mathcal{A}}} x_a \wedge \bigwedge_{\substack{a\in body(r)\\ a\in a\in\mathcal{L}\setminus\mathcal{A}}} d_a \right)$$

states that s is derived from A_τ if and only if there is a corresponding rule which is activated. Similarly for derivations from B_τ, we define

$$\varphi_B^{ufs}(s) = e_s \leftrightarrow \bigvee_{\substack{r\in\mathcal{R}\\ head(r)=s}} \left(\bigwedge_{\substack{a\in body(r)\\ a\in\mathcal{A}}} y_a \wedge \bigwedge_{\substack{a\in body(r)\\ a\in a\in\mathcal{L}\setminus\mathcal{A}}} e_a \right).$$

We define $\Phi_\sigma^{ufs}(F) = \varphi_\sigma(F) \wedge \varphi_{att}(F) \wedge \varphi_{ndef}(F) \wedge \bigwedge_{s\in\mathcal{L}\setminus\mathcal{A}} \varphi_A^{ufs}(s) \wedge \bigwedge_{s\in\mathcal{L}\setminus\mathcal{A}} \varphi_B^{ufs}(s)$, where $\varphi_\sigma(F)$ encodes the base semantics (recall Section 3), $\varphi_{att}(F)$ and $\varphi_{ndef}(F)$ encodes the interpretation of the y_a and z_a variables (recall

Section 3.2). For $\sigma = stb$, again, the subformulas $\varphi_{ndef}(F)$ and $\varphi_B^{ufs}(F)$ are not required.

It suffices to ensure that the corresponding derivations do not contain unfounded atoms. For this, all d_s as well as e_s variables for $s \in \mathcal{L} \setminus \mathcal{A}$ are set as observed. We keep track of assignments to these variables whenever the solver notifies about an assignment or backtracks to a previous decision level. For each $s \in \mathcal{L} \setminus \mathcal{A}$, we keep track of two source pointers, $source_d(s)$ and $source_e(s)$. When propagating, as long as $\tau(d_s) \neq 0$ (resp. $\tau(e_s) \neq 0$), we attempt to set $source_d(s)$ (resp. $source_e(s)$) to a rule $r \in \mathcal{R}$ which can provide non-circular support for deriving s. The set of atoms U for which a source pointer cannot be established is unfounded. In this case, we propagate $\neg d_s$ (resp. $\neg e_s$), with the reason clause stating that there is no rule which can provide external support for U. For more details on unfounded set propagation, we refer the reader to (Gebser, Kaufmann, and Schaub 2012).

5 Empirical Evaluation

We present an empirical evaluation of the runtime of our SAT-based approaches for reasoning in ABA. We focus on the following central problems in the evaluation: credulous reasoning under complete semantics (DC-CO), credulous and skeptical reasoning under stable semantics (DC-ST and DS-ST), skeptical reasoning under preferred (DS-PR) and computing the ideal extension (SE-ID). The experiments were run on 2.50-GHz Intel Xeon Gold 6248 CPUs with a per-instance 15-min time and 16-GB memory limit.

5.1 Implementation

Our implementation of the SAT-based approaches uses the CaDiCaL SAT solver (Biere et al. 2024) v2.1.3. The acyclicity and unfounded set propagators were implemented via the IPASIR-UP interface of CaDiCaL (Fazekas et al. 2024). We will refer to the implementations as SAT-LEVEL (SAT encoding with level-based acyclicity encoding), SAT-VE (SAT encoding with vertex elimination for acyclicity) SAT-ACYC (SAT with graph-based acyclicity propagator), and SAT-UFS (SAT with unfounded set propagator). The implementations, benchmark generators and data are available in open source via https://bitbucket.org/coreo-group/100ba.

For SE-ID and DS-PR, we adapt the iterative SAT-based algorithms implemented in the abstract argumentation solver μ-TOKSIA (Niskanen and Järvisalo 2020) to ABA reasoning. The algorithms for DS-PR and SE-ID in μ-TOKSIA initialize a SAT solver with complete semantics as a base abstraction and perform incremental calls to the solver. For DS-PR μ-TOKSIA implements a SAT-based counterexample-guided abstraction refinement (CEGAR) approach to iteratively find candidate complete extensions not containing the query argument, and checks for counterexample supersets of the candidate extension (Dvořák et al. 2014). For SE-ID, μ-TOKSIA implements an iterative SAT approach which first finds all arguments attacked by a complete extension, and then finds the unique subset-maximal complete extension in the complement (Niskanen and Järvisalo 2020). As both algorithms are based on complete semantics and the ABA semantics are analogous with abstract argumentation semantics, the algorithmic concept extend to ABA by using our CaDiCaL-based approach for complete semantics in ABA as the abstraction solver.

We compare our SAT-based approaches to ASP-FORABA (Lehtonen, Wallner, and Järvisalo 2021a) which was the best-performing system in the ABA track of the most recent argumentation solver competition ICCMA 2023 (Järvisalo, Lehtonen, and Niskanen 2025). ASP-FORABA employs ASP encodings for ABA reasoning tasks, using the state-of-the-art ASP solver CLINGO (Gebser et al. 2011; Gebser et al. 2016) v5.6.2 for obtaining a solution, either in one shot (for problems in NP), using CLINGO in its cautious reasoning mode for computing the ideal extension, or iteratively in the style of CEGAR (Lehtonen, Wallner, and Järvisalo 2021b) for skeptical acceptance under preferred.

5.2 Benchmarks

We consider two benchmark sets in the evaluation: ICCMA 2023 (Järvisalo, Lehtonen, and Niskanen 2025) ABA Track benchmark set (ICCMA23 for short) and benchmarks generated by a new CLUSTERED generator which we propose based on ideas from the existing abstract argumentation benchmark generator StableGenerator (Cerutti et al. 2014; Thimm and Villata 2017).

The ICCMA23 set consists of 400 ABFs. We use the index-1 sentence as the query. There are 80 instances with 25, 100, 500, 2000 and 5000 atoms in them, respectively. In half of the instances 10% of the atoms are assumptions and in the other half 30%. For each number of sentences, there are 10 ABFs for each combination $(5, 5)$, $(5, 10)$, $(10, 5)$ and $(10, 10)$ of (maximum number of rules per atom, the maximum body size of rules). Importantly, we observed that there are limitations to basing a comprehensive runtime evaluation on the ICCMA23 set: a significant portion of the instances (216 for DC-ST, 289 for DS-ST and 154 for DC-CO and by extension DS-PR) are unsatisfiable without even considering the acyclicity constraints outlined in Section 3. It suffices to simply call a SAT solver on the base SAT encoding *without enforcing acyclicity at all* to solve these instances. Thus, as the acyclicity constraint is a very significant part of the problems, we filtered out all such ICCMA23 instances from the evaluation (apart from SE-ID as the ideal extension always exists and thus there are no unsatisfiable instances).

We construct a complementary benchmark set CLUSTERED, using a scheme that ensures that the ABFs admit multiple stable assumption sets. We partition the assumptions of the ABF being constructed and let the assumptions of each partition attack the majority of assumptions outside of it. Our aim is to have many stable (and thus complete and preferred) assumption sets while having acyclicity in the rules. The generator selects the number of partitions P and the numbers of assumptions $P_{\mathcal{A}}$ and non-assumption atoms $P_{\mathcal{L}}$ in each partition, the maximum number of rules per atom mra, the maximum rule size mrs, and a contrary probability C. For each i in $1, \ldots, P$, let $\mathcal{A}_i$ be a set of $P_{\mathcal{A}}$ fresh assumptions for partition i, let each assumption in $\mathcal{A}_i$ have a fresh contrary

		#solved (mean runtime)							
		(mra, mrs)							
Problem	Solver	$(10,10)$		$(10,5)$		$(5,10)$		$(5,5)$	
DC-CO	ASPFORABA	65	(3.8)	31	(0.9)	76	(0.1)	54	(11.4)
	SAT-LEVEL	62	(19.2)	24	(51.9)	76	(2.3)	48	(8.8)
	SAT-VE	**72**	**(13.5)**	31	(85.9)	**76**	**(0.0)**	**59**	**(29.9)**
	SAT-ACYC	66	(5.2)	31	(21.3)	**76**	**(0.0)**	53	(21.2)
	SAT-UFS	65	(0.8)	**31**	**(0.5)**	**76**	**(0.0)**	56	(5.6)
DC-ST	ASPFORABA	46	(0.1)	19	(0.5)	59	(0.1)	43	(23.8)
	SAT-LEVEL	46	(7.2)	15	(10.3)	59	(1.2)	37	(6.8)
	SAT-VE	**55**	**(14.8)**	19	(50.8)	**59**	**(0.0)**	**46**	**(34.8)**
	SAT-ACYC	46	(0.0)	19	(31.7)	**59**	**(0.0)**	41	(10.1)
	SAT-UFS	47	(9.6)	**19**	**(0.1)**	**59**	**(0.0)**	43	(17.1)
DS-ST	ASPFORABA	13	(1.0)	49	(0.6)	16	(0.1)	17	(3.5)
	SAT-LEVEL	11	(3.1)	36	(121.1)	16	(1.1)	13	(0.7)
	SAT-VE	**19**	**(35.3)**	49	(87.6)	**16**	**(0.0)**	**23**	**(57.7)**
	SAT-ACYC	13	(1.9)	39	(36.8)	**16**	**(0.0)**	16	(8.3)
	SAT-UFS	13	(1.1)	**49**	**(0.1)**	**16**	**(0.0)**	17	(2.4)
DS-PR	ASPFORABA	65	(8.3)	**31**	**(1.4)**	76	(0.1)	53	(11.3)
	SAT-LEVEL	62	(19.9)	22	(30.3)	76	(2.4)	47	(10.9)
	SAT-VE	62	(0.5)	25	(19.8)	**76**	**(0.0)**	48	(6.3)
	SAT-ACYC	65	(9.2)	31	(30.0)	**76**	**(0.0)**	52	(33.7)
	SAT-UFS	**65**	**(4.3)**	31	(1.9)	**76**	**(0.0)**	**54**	**(13.3)**
SE-ID	ASPFORABA	89	(11.9)	99	(4.3)	100	(0.1)	84	(16.5)
	SAT-LEVEL	80	(30.4)	66	(112.7)	100	(2.2)	75	(17.4)
	SAT-VE	80	(0.5)	62	(28.1)	**100**	**(0.0)**	79	(37.0)
	SAT-ACYC	89	(14.5)	87	(38.2)	**100**	**(0.0)**	85	(35.1)
	SAT-UFS	**89**	**(4.4)**	**100**	**(1.5)**	**100**	**(0.0)**	**88**	**(21.6)**

Table 1: Number of solved instances and mean runtime over solved on the ICCMA23 benchmark set.

atom, and let $\mathcal{L}_i$ be a set of $P_\mathcal{L}$ fresh non-assumption atoms. For each $x \in \mathcal{L}_i$, generate $r \in [1, mra]$ rules for x of size $s \in [1, mrs]$ (r and s selected uniformly at random) such that the rule body consists of s atoms from $\mathcal{L}_i \cup \mathcal{A}_i$. Finally, for each assumption $a \in \mathcal{A} \setminus \mathcal{A}_i$, generate a rule $(\overline{a} \leftarrow x)$ for a randomly selected $x \in \mathcal{L}_i$ with probability C. As a result, the CLUSTERED ABFs cannot have a strongly connected component (SCC) in the underlying rule graph larger than $P_\mathcal{L}$, but many of the atoms within each cluster will belong to the same SCC. There are no attacks within the partitions. With C close to 1, the assumptions in each partition attack most assumptions outside the partition. This intuitively results in many stable extensions: in particular, for $C = 1$, each partition by itself is a stable extension (if all atoms are derivable). We found that with $mra = 3$ and $mrs = 2$ and $P_\mathcal{A}$ not much smaller than $P_\mathcal{L}$ virtually all atoms within each partition are derivable, and the size of SCCs typically equal roughly half or more $P_\mathcal{L}$. To limit the number of varying parameters when generating the CLUSTERED benchmark set, we fixed $mra = 3$ and $mrs = 2$ and generated two separate sets of benchmarks, one with small number of large partitions and one with large number of small partitions. Namely, we let each P, $P_\mathcal{A}$ and $P_\mathcal{L}$ take a value from $[25, 30, 35]$ in one set and $P \in [5, 7]$, and $P_\mathcal{A}$ and $P_\mathcal{L}$ take a value from $[100, 150, 200]$ in the

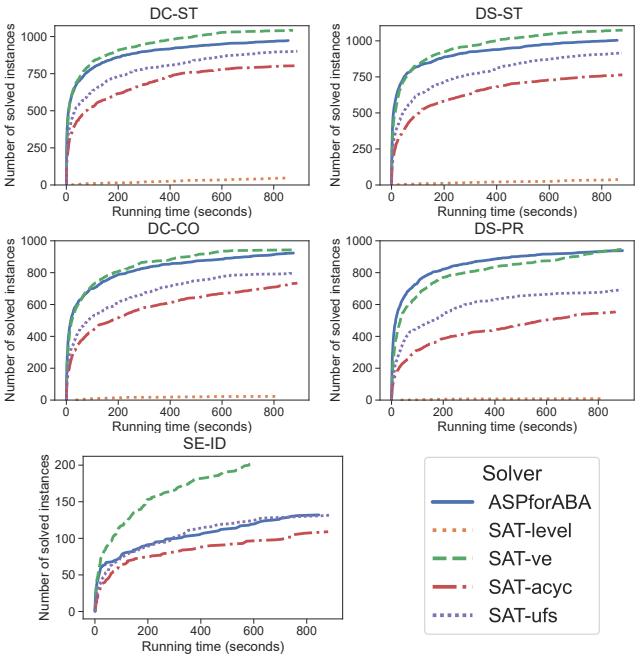

Figure 1: SAT-VE, ASPFORABA and SAT-UFS on DC-CO, DC-ST, and DS-ST on the CLUSTERED instances.

other. We generated five ABFs for each combination of P, $P_\mathcal{L}$, and $P_\mathcal{A}$ and $C = 0.8$ and $C = 0.9$, for a total of 450 ABFs. For acceptance problems, for each ABF we perform three separate queries: one assumption, one contrary atom, and one atom that is neither an assumption nor a contrary, for a total of 1350 instances. The reason for three separate queries is that these different atom types might behave differently due to the structure of the ABFs: contrary atoms are derivable from multiple clusters while other atoms only from a single cluster.

5.3 Results

An overview of solver performance on the ICCMA23 benchmarks is shown in Table 1. We report the number of solved instances and mean runtime over solved instances for each parameter pair (mra, mrs) (i.e. max rules per atom and max rule body size). Instances for the combination $(5, 10)$ turn out to be trivial for all solvers. We observe that SAT-VE and SAT-UFS perform particularly well, both outperforming the state-of-the-art solver ASPFORABA on many problems. SAT-UFS outperforms the others especially on SE-ID and SAT-VE performs well on the one-shot problems DC-CO, DC-ST and DS-ST. Among the other SAT-based approaches, SAT-LEVEL is the least performant overall, as can be expected.

The relatively small performance differences further motivate experiments on the CLUSTERED benchmark set which turns out to reveal more significant and complementary differences. Figure 1 shows for each problem the number of instances solved (y-axis) under different per-instance time limits (x-axis), i.e, the higher the line for a solver, the better the solver performs. SAT-LEVEL memouts on most instances,

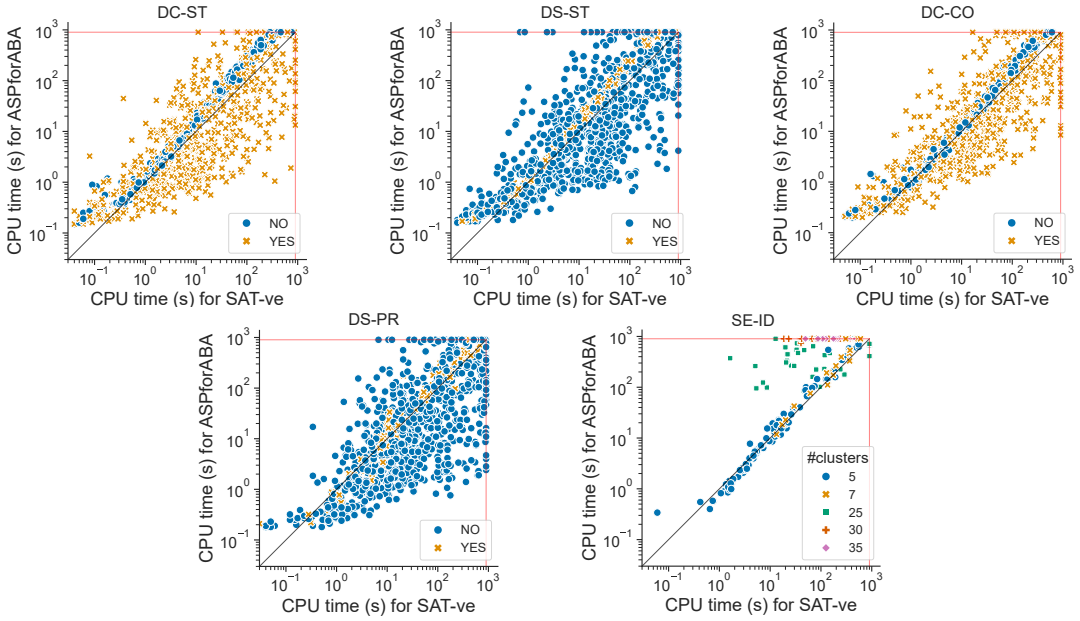

Figure 2: Per-instance runtime comparison of ASPFORABA and SAT-VE on the CLUSTERED benchmark set for DC-ST, DS-ST, DC-CO, DS-PR, and SE-ID.

underlining the need for more sophisticated SAT techniques. On this benchmark set, SAT-VE solves the most instances on each problem. SAT-VE also performs best in terms of runtimes on all problems except DS-PR, where ASPFORABA solves more instances within any given runtime less than roughly 800 seconds, after which SAT-VE manages to solve a few more instances than ASPFORABA. Aside from SAT-LEVEL, SAT-ACYC is the least performant approach overall. On the beyond-NP problem SE-ID, SAT-VE excels by solving ≈ 50 % more instances than ASPFORABA.

For a closer look at the relative performance of SAT-VE and ASPFORABA, the two best-performing approaches on CLUSTERED, Figure 2 shows a per-instance runtime comparison for each of the problems DC-ST, DS-ST, DC-CO, and DS-PR, distinguishing between NO and YES benchmark instances. On DC-ST (Figure 2, top left), SAT-VE consistently outperforms ASPFORABA on NO-instances, which may be due to CaDiCaL being more effective in proving unsatisfiability. On YES-instances the relative performance of the solvers varies more, with SAT-VE still narrowly solving more instances than ASPFORABA: 663 vs 655. Dually, on DS-ST (top center) where the YES-instances require proving unsatisfiability, we observe very similar relative performance. DC-CO (top right) and DS-PR (bottom left) follow the same pattern. Finally, bottom right of Figure 2 provides a more fine-grained view of SAT-VE vs ASPFORABA on SE-ID wrt. the number of clusters in the individual CLUSTERED benchmarks. On instances with small number (5 or 7) of larger clusters, the approaches exhibit similar runtimes. However, on instances with a larger number (25, 30 or 35) of clusters SAT-VE clearly outperforms ASPFORABA. Notably, ASPFORABA was able to solve only one instance with 30 clusters and no instances with 35 clusters.

Regarding the size of the encodings (see more details in the supplement), for the ICCMA23 instance which exhibits the average number of clauses for SAT-VE, the number of variables and clauses, resp., for the encodings are 167k and 32M for SAT-VE, and 8M and 33M for SAT-LEVEL. Analogously, for CLUSTERED these numbers are 7k and 45K for SAT-VE, and 8M and 75M for SAT-LEVEL. This highlights the structure-based compactness of SAT-VE. The ratio of number of clauses in SAT-VE and number of clauses in the "base" encoding without acyclicity constraints over solved instances is: for ICCMA23 min 1.12, max 2833.5, and average 224.9; and for CLUSTERED min 1.03, max 1.74, average 1.14. The UFS propagator took a maximum of 4 % of total runtime (average < 1 %) on the CLUSTERED set, while on ICCMA23 UFS propagation took a maximum of 81 % of total runtime (average < 4 %).

6 Conclusions

We addressed the challenge of developing SAT-based approaches to reasoning in the structured argumentation formalism of ABA. Extending a base encoding for central ABA semantics, we studied the applicability of both recent advances in SAT encodings and two types of domain-specific propagators for enforcing acyclic derivations towards scaling up SAT-based reasoning in ABA. The resulting SAT-based ABA reasoners support a range of central argumentation semantics and both NP-complete and beyond-NP reasoning tasks. Overall, the SAT-based approach, especially enforcing acyclicity by vertex elimination or UFS propagation, turned out to scale even beyond the current state-of-the-art approach to reasoning in ABA, based on ASP solving. We also provided a new ABA benchmark generator.

Acknowledgements

The work is financially support by Research Council of Finland under grants 347588 (AN) and 356046 (MFR, MJ), and Helsinki Institute for Information Technology HIIT (TL, MFR). The authors wish to thank the Finnish Computing Competence Infrastructure (FCCI) for supporting this project with computational and data storage resources.

References

Atkinson, K.; Baroni, P.; Giacomin, M.; Hunter, A.; Prakken, H.; Reed, C.; Simari, G. R.; Thimm, M.; and Villata, S. 2017. Towards artificial argumentation. *AI Magazine* 38(3):25–36.

Bao, Z.; Čyras, K.; and Toni, F. 2017. ABAplus: Attack reversal in abstract and structured argumentation with preferences. In *PRIMA*, volume 10621 of *LNCS*, 420–437. Springer.

Baroni, P.; Gabbay, D.; Giacomin, M.; and van der Torre, L., eds. 2018. *Handbook of Formal Argumentation*. College Publications.

Bench-Capon, T. J. M., and Dunne, P. E. 2007. Argumentation in artificial intelligence. *Artif. Intell.* 171(10-15):619–641.

Besnard, P., and Doutre, S. 2004. Checking the acceptability of a set of arguments. In *NMR*, 59–64.

Besnard, P., and Hunter, A. 2018. A review of argumentation based on deductive arguments. In *Handbook of Formal Argumentation*. College Publications. 437–484.

Besnard, P.; García, A. J.; Hunter, A.; Modgil, S.; Prakken, H.; Simari, G. R.; and Toni, F. 2014. Introduction to structured argumentation. *Argument Comput.* 5(1):1–4.

Biere, A.; Faller, T.; Fazekas, K.; Fleury, M.; Froleyks, N.; and Pollitt, F. 2024. CaDiCaL 2.0. In *CAV*, volume 14681 of *LNCS*, 133–152. Springer.

Biere, A. 2009. Bounded model checking. In *Handbook of Satisfiability*, volume 185 of *FAIA*. IOS Press. 457–481.

Bistarelli, S.; Kotthoff, L.; Lagniez, J.; Lonca, E.; Mailly, J.; Rossit, J.; Santini, F.; and Taticchi, C. 2025. The third and fourth international competitions on computational models of argumentation: Design, results and analysis. *Argument Comput.* 16(2):236–299.

Bondarenko, A.; Dung, P. M.; Kowalski, R. A.; and Toni, F. 1997. An abstract, argumentation-theoretic approach to default reasoning. *Artif. Intell.* 93:63–101.

Borg, A., and Odekerken, D. 2022. PyArg for solving and explaining argumentation in python: Demonstration. In *COMMA*, volume 353 of *FAIA*, 349–350. IOS Press.

Brewka, G., and Woltran, S. 2010. Abstract dialectical frameworks. In *KR*, 102–111. AAAI Press.

Brewka, G.; Polberg, S.; and Woltran, S. 2014. Generalizations of dung frameworks and their role in formal argumentation. *IEEE Intell. Syst.* 29(1):30–38.

Calegari, R.; Omicini, A.; Pisano, G.; and Sartor, G. 2022. Arg2P: an argumentation framework for explainable intelligent systems. *J. Log. Comput.* 32(2):369–401.

Cerutti, F.; Oren, N.; Strass, H.; Thimm, M.; and Vallati, M. 2014. A benchmark framework for a computational argumentation competition. In *COMMA*, volume 266 of *FAIA*, 459–460. IOS Press.

Cerutti, F.; Gaggl, S. A.; Thimm, M.; and Wallner, J. P. 2018. Foundations of implementations for formal argumentation. In *Handbook of Formal Argumentation*. College Publications. 688–767.

Craven, R., and Toni, F. 2016. Argument graphs and assumption-based argumentation. *Artif. Intell.* 233:1–59.

Čyras, K.; Fan, X.; Schulz, C.; and Toni, F. 2018. Assumption-based argumentation: Disputes, explanations, preferences. In *Handbook of Formal Argumentation*. College Publications. 365–408.

Cyras, K.; Heinrich, Q.; and Toni, F. 2021. Computational complexity of flat and generic assumption-based argumentation, with and without probabilities. *Artificial Intelligence* 293:103449.

Diller, M.; Gaggl, S. A.; and Gorczyca, P. 2021. Flexible dispute derivations with forward and backward arguments for assumption-based argumentation. In *CLAR*, volume 13040 of *LNCS*, 147–168. Springer.

Diller, M.; Gaggl, S. A.; and Gorczyca, P. 2022. Strategies in flexible dispute derivations for assumption-based argumentation. In *SAFA*, volume 3236 of *CEUR Workshop Proceedings*, 59–72. CEUR-WS.org.

Dimopoulos, Y.; Nebel, B.; and Toni, F. 2002. On the computational complexity of assumption-based argumentation for default reasoning. *Artif. Intell.* 141(1/2):57–78.

Dung, P. M.; Mancarella, P.; and Toni, F. 2007. Computing ideal sceptical argumentation. *Artif. Intell.* 171(10-15):642–674.

Dung, P. M. 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artif. Intell.* 77(2):321–358.

Dunne, P. E., and Wooldridge, M. 2009. Complexity of abstract argumentation. In *Argumentation in Artificial Intelligence*. Springer. 85–104.

Dunne, P. E. 2009. The computational complexity of ideal semantics. *Artif. Intell.* 173(18):1559–1591.

Dvořák, W., and Dunne, P. E. 2018. Computational problems in formal argumentation and their complexity. In *Handbook of Formal Argumentation*. College Publications. 631–687.

Dvořák, W.; Järvisalo, M.; Wallner, J. P.; and Woltran, S. 2014. Complexity-sensitive decision procedures for abstract argumentation. *Artif. Intell.* 206:53–78.

Fazekas, K.; Niemetz, A.; Preiner, M.; Kirchweger, M.; Szeider, S.; and Biere, A. 2024. Satisfiability modulo user propagators. *J. Artif. Intell. Res.* 81:989–1017.

Gaggl, S. A.; Linsbichler, T.; Maratea, M.; and Woltran, S. 2020. Design and results of the second international competition on computational models of argumentation. *Artif. Intell.* 279.

García, A. J., and Simari, G. R. 2004. Defeasible logic programming: An argumentative approach. *Theory Pract. Log. Program.* 4(1-2):95–138.

García, A. J., and Simari, G. R. 2014. Defeasible logic programming: DeLP-servers, contextual queries, and explanations for answers. *Argument Comput.* 5(1):63–88.

Gebser, M.; Kaufmann, B.; Kaminski, R.; Ostrowski, M.; Schaub, T.; and Schneider, M. T. 2011. Potassco: The Potsdam answer set solving collection. *AI Commun.* 24(2):107–124.

Gebser, M.; Kaminski, R.; Kaufmann, B.; Ostrowski, M.; Schaub, T.; and Wanko, P. 2016. Theory solving made easy with clingo 5. In *ICLP 2016 TCs*, volume 52 of *OASIcs*, 2:1–2:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Gebser, M.; Janhunen, T.; and Rintanen, J. 2014. SAT modulo graphs: Acyclicity. In *JELIA*, volume 8761 of *LNCS*, 137–151. Springer.

Gebser, M.; Kaufmann, B.; and Schaub, T. 2012. Conflict-driven answer set solving: From theory to practice. *Artif. Intell.* 187:52–89.

Gelfond, M., and Lifschitz, V. 1988. The stable model semantics for logic programming. In *ICLP/SLP*, 1070–1080. MIT Press.

Jean-Marie Lagniez, E. L., and Mailly, J.-G. 2023. Crustabri, the evolution of CoQuiAAS. In *Solver and Benchmark Descriptions of ICCMA 2023: 5th International Competition on Computational Models of Argumentation*, Department of Computer Science Series of Publications B, 20–21. University of Helsinki.

Järvisalo, M.; Lehtonen, T.; and Niskanen, A. 2025. ICCMA 2023: 5th international competition on computational models of argumentation. *Artif. Intell.* 342:104311.

Lagniez, J.; Lonca, E.; and Mailly, J. 2015. Coquiaas: A constraint-based quick abstract argumentation solver. In *ICTAI*, 928–935. IEEE Computer Society.

Lehtonen, T.; Rapberger, A.; Ulbricht, M.; and Wallner, J. P. 2023. Argumentation frameworks induced by assumption-based argumentation: Relating size and complexity. In *KR*, 440–450.

Lehtonen, T.; Odekerken, D.; Wallner, J. P.; and Järvisalo, M. 2024a. Complexity results and algorithms for preferential argumentative reasoning in ASPIC+. In *KR*.

Lehtonen, T.; Rapberger, A.; Toni, F.; Ulbricht, M.; and Wallner, J. P. 2024b. Instantiations and computational aspects of non-flat assumption-based argumentation. In *IJCAI*, 3457–3465. ijcai.org.

Lehtonen, T.; Wallner, J. P.; and Järvisalo, M. 2017. From structured to abstract argumentation: Assumption-based acceptance via AF reasoning. In *ECSQARU*, volume 10369 of *LNCS*, 57–68. Springer.

Lehtonen, T.; Wallner, J. P.; and Järvisalo, M. 2021a. Declarative algorithms and complexity results for assumption-based argumentation. *J. Artif. Intell. Res.* 71:265–318.

Lehtonen, T.; Wallner, J. P.; and Järvisalo, M. 2021b. Harnessing incremental answer set solving for reasoning in assumption-based argumentation. *Theory Pract. Log. Program.* 21(6):717–734.

Lehtonen, T.; Wallner, J. P.; and Järvisalo, M. 2022. Computing stable conclusions under the weakest-link principle in the ASPIC+ argumentation formalism. In *KR*, 215–225.

Niemelä, I. 1999. Logic programs with stable model semantics as a constraint programming paradigm. *Ann. Math. Artif. Intell.* 25(3-4):241–273.

Niskanen, A., and Järvisalo, M. 2020. μ-toksia: An efficient abstract argumentation reasoner. In *KR*, 800–804.

Popescu, A., and Wallner, J. P. 2023. Reasoning in assumption-based argumentation using tree-decompositions. In *JELIA*, volume 14281 of *LNCS*, 192–208. Springer.

Prakken, H. 2010. An abstract framework for argumentation with structured arguments. *Argument Comput.* 1(2):93–124.

Rankooh, M. F., and Rintanen, J. 2022. Propositional encodings of acyclicity and reachability by using vertex elimination. In *AAAI*, 5861–5868. AAAI Press.

Rose, D. J.; Tarjan, R. E.; and Lueker, G. S. 1976. Algorithmic aspects of vertex elimination on graphs. *SIAM J. Comput.* 5(2):266–283.

Strass, H.; Wyner, A.; and Diller, M. 2019. *EMIL*: Extracting meaning from inconsistent language: Towards argumentation using a controlled natural language interface. *Int. J. Approx. Reason.* 112:55–84.

Thimm, M., and Villata, S. 2017. The first international competition on computational models of argumentation: Results and analysis. *Artificial Intelligence* 252:267–294.

Thimm, M.; Cerutti, F.; and Vallati, M. 2021. Skeptical reasoning with preferred semantics in abstract argumentation without computing preferred extensions. In *IJCAI*, 2069–2075. ijcai.org.

Toni, F. 2014. A tutorial on assumption-based argumentation. *Argument Comput.* 5(1):89–117.