Computational Models of Argument
F. Toni et al. (Eds.)

doi:10.3233/FAIA220156

Algorithms for Reasoning in a Default Logic Instantiation of Assumption-Based Argumentation[1]

Tuomo LEHTONEN [a], Johannes P. WALLNER [b] and Matti JÄRVISALO [a]
[a] *University of Helsinki, Finland*
[b] *Graz University of Technology, Austria*
ORCiD ID: Tuomo Lehtonen https://orcid.org/0000-0001-6117-4854, Johannes P. Wallner https://orcid.org/0000-0002-3051-1966, Matti Järvisalo https://orcid.org/0000-0003-2572-063X

Abstract. Assumption-based argumentation (ABA) is one of the most-studied formalisms for structured argumentation. While ABA is a general formalism that can be instantiated with various different logics, most attention from the computational perspective has been focused on the logic programming (LP) instantiation of ABA. Going beyond the LP-instantiation, we develop an algorithmic approach to reasoning in the propositional default logic (DL) instantiation of ABA. Our approach is based on iterative applications of Boolean satisfiability (SAT) solvers as a natural choice for implementing derivations as entailment checks in DL. We instantiate the approach for deciding acceptance and for assumption-set enumeration in the DL-instantiation of ABA under several central argumentation semantics, and empirically evaluate an implementation of the approach.

Keywords. structured argumentation, assumption-based argumentation, default logic, decision procedures, SAT, counterexample-guided abstraction refinement

1. Introduction

Assumption-based argumentation (ABA) [1,2] is a central approach to structured argumentation [3,4,5]. ABA captures and generalizes different approaches default reasoning, constituting a general-purpose framework which can be instantiated for any formal logic to support various different application settings. Derivations of conclusions from assumptions via inference rules in the logic of choice give rise to arguments.

Arguably the most-studied ABA instantiation is the logic programming (LP) fragment of ABA [1] in which arguments are derived with logic programming rules. From the computational perspective, development of algorithmic approaches to central reasoning problems, such as acceptance, in ABA has focused on the LP-instantiation [6,7,8,9,10]. In the LP-instantiation, derivations are computable in polynomial time, which implies

[1]This work has been financially supported in part by Academy of Finland grant 322869 and the University of Helsinki Doctoral Programme in Computer Science DoCS. The authors wish to thank the Finnish Computing Competence Infrastructure (FCCI) for supporting this project with computational and data storage resources.

that deciding acceptance in ABA is a computational problem contained in NP for various argumentation semantics [11]. However, algorithmic approaches to reasoning in ABA beyond the LP-instantiation, i.e., ABA instantiated with more expressive logics, and in particular ones for which derivations may be hard to compute, are scarce. Addressing this challenge, in this work we develop algorithms for reasoning in ABA instantiated for propositional default logic (DL) [12,13], which we refer to as DL-ABA.

In DL-ABA, derivations of arguments require checking entailment (coNP). This implies that reasoning about acceptance is beyond the first level of the polynomial hierarchy [11]. The need for entailment checking suggests using Boolean satisfiability (SAT) solvers [14] as a basis for developing approaches to reasoning in DL-ABA. Indeed, we employ incremental SAT solving for developing an approach that allows for deciding acceptance and the enumeration of assumption-sets under several central argumentation semantics. We provide an implementation of the approach and a first empirical evaluation of its scalability. Although there have been earlier developments for computing extensions in DL [15,16,17,18] which in particular corresponds to computing extensions under stable semantics in DL-ABA, to our best understanding our approach presented is the first in its generality for DL-ABA.

2. Preliminaries

We review background on assumption-based argumentation (ABA) [1,2] with propositional default logic (DL) [12] as the underlying deductive system.

The first ingredient for ABA is a deductive system $(\mathscr{L},\mathscr{R})$. For our purposes $\mathscr{L}$ is a set of propositional formulas and $\mathscr{R}$ a set of rules of the form $r = a_0 \leftarrow a_1,\ldots,a_n$ with $a_i \in \mathscr{L}$. We say that a_0 is the head of the rule ($head(r) = a_0$) and the set $\{a_1,\ldots,a_n\}$ is the body ($body(r) = \{a_1,\ldots,a_n\}$). A sentence $a \in \mathscr{L}$ is derivable from $A \subseteq \mathscr{L}$ (in symbols $A \vdash_{\mathscr{R}} a$) if either $a \in A$, or there is a sequence of rules $(r_1,\ldots,r_n)$ from $\mathscr{R}$ s.t. $head(r_n) = a$ and $body(r_i) \subseteq \bigcup_{j<i} head(r_j) \cup A$ for $1 \leq i \leq n$; that is, each body element of rules must be present either in A or as heads of previous rules in the sequence. We omit subscript $\mathscr{R}$ when clear from context. We assume that $(\mathscr{L}_p,\mathscr{R}_p)$ is a deductive system for propositional logic, i.e., $\mathscr{L}_p$ is the set of all propositional formulas and $A \vdash_{\mathscr{R}_p} a$ iff $A \models a$ (i.e., there is a derivation via $\mathscr{R}_p$ iff classical semantic entailment holds; one may select any sound and complete inference system for classical propositional logic as $\mathscr{R}_p$).

A propositional default theory is a pair $T = (W,D)$, where $W \subseteq \mathscr{L}_p$ and D is a set of default rules of the form $r = c \leftarrow a, Mb_1,\ldots,Mb_n$ with $c,a,b_1,\ldots,b_n \in \mathscr{L}_p$ and $Mb_i \notin \mathscr{L}_p$ (Mb_i are not propositional formulas). We refer to c as the conclusion, to a as the prerequisite, and to $\{Mb_1,\ldots,Mb_m\}$ as the justifications of the default rule r. We use the shorthands $M(r) = \{Mb_1,\ldots,Mb_n\}$, $prereq(r) = a$ and $conc(r) = c$. Intuitively, Mb is interpreted as $\neg b$ can not be proven and thus it is consistent to assume b.

We directly state ABA instantiated with propositional default logic.

Definition 1. *Let (W,D) be a propositional default theory. The assumption-based argumentation framework (ABF) corresponding to (W,D) is $F = (\mathscr{L},\mathscr{R},W,\mathscr{A},^-)$ with*

- $\mathscr{L} = \mathscr{L}_p \cup \{M\alpha \mid \alpha \in \mathscr{L}_p\}$,
- $\mathscr{R} = \mathscr{R}_p \cup D$,
- $\mathscr{A} = \{Mb \mid Mb$ *occurs in some default rule in* D, $b \in \mathscr{L}_p\}$, *and*

- $^-$ *a function mapping* $\mathscr{A}$ *to* $\mathscr{L}$ *defined by* $\overline{Mb} = \neg b$ *for all* $Mb \in \mathscr{A}$.

An $Mb \in \mathscr{A}$ is an assumption (in a given ABF). For brevity, as (W,D) uniquely determines the corresponding ABF and as we focus on ABFs corresponding to propositional default theories, we identify (W,D) with the corresponding ABF F and write $F = (W,D)$ referring to the ABF corresponding to (W,D).

Given an ABF, derivability from a set of assumptions A is defined by $W \cup A \vdash_{\mathscr{R}} a$, i.e., a is derivable from W and the assumption set (note that $\mathscr{R}$ includes default rules from the default theory). A set of assumptions $A \subseteq \mathscr{A}$ attacks a set of assumptions $B \subseteq \mathscr{A}$ iff $W \cup A \vdash_{\mathscr{R}} \overline{Mb}$ for some $b \in B$, or equivalently, $W \cup A \vdash_{\mathscr{R}} \neg b$; that is, a set of assumptions is a set of justifications (which are "assumed"), and A attacks B, if one can derive the negation of some justification in B from the propositional theory W with all default rules whose justifications are met by A. For a singleton $B = \{b\}$ we say that A attacks b. Note that an atom b may be entailed although Mb is not, as assumptions are not part of the propositional vocabulary. In ABA terminology, an ABF is flat if assumptions do not occur as heads of rules, as is the case for DL-ABA.

A set of assumptions A is conflict-free if A does not attack A. A defends the assumption set B if A attacks each assumption set C that attacks B.

Definition 2. *For a given ABF* $F = (W,D)$ *and a conflict-free set of assumptions* $A \subseteq \mathscr{A}$, *we say* A *is*

- *admissible (in* F*) if* A *defends itself,*
- *complete (in* F*) if* A *is admissible and contains every assumption set it defends,*
- *grounded (in* F*) if* A *is* $\subseteq$*-minimally complete, and*
- *stable (in* F*) if* A *is conflict-free and attacks every assumption* $a \in \mathscr{A} \setminus A$.

A useful equivalent characterization for grounded semantics is by utilizing $\mathscr{F}_F$ for an ABF $F = (W,D)$, defined by $\mathscr{F}_F(S) = \{Mb \in \mathscr{A} \mid \{Mb\}$ defended by $S\}$ for an $S \subseteq \mathscr{A}$. The grounded assumption set is then the (unique) least fixed point of $\mathscr{F}_F$ [1]. There is a direct correspondence between Reiter's default extensions [12] and stable assumption sets: E is a default extension of (W,D) iff E' is a stable assumption set of $F = (W,D)$ with $E = \{a \mid W \cup E' \vdash a\} \cap \mathscr{L}_p$ [1]. An atom $x \in \mathscr{L}$ is credulously (resp., skeptically) accepted under semantics $\sigma \in \{adm, com, grd, stb\}$ (for admissible, complete, grounded, stable) if x is derivable from some (resp., all) σ-assumption set(s).

Example 1. *Let* $W = (\neg b \vee \neg a) \wedge (\neg b \vee \neg c)$ *and* D *contain four default rules:* $(r_1 = a \leftarrow Ma)$, $(r_2 = b \leftarrow Mb)$, $(r_3 = c \leftarrow Mc)$, *and* $(r_4 = d \leftarrow a \wedge b, Md)$. *For the corresponding ABF,* $\mathscr{L}$ *and* $\mathscr{R}$ *are together an extension of a propositional deductive system with defaults and the assumptions* $\mathscr{A} = \{Ma, Mb, Mc, Md\}$. *Contrariness is given by* $\overline{Mx} = \neg x$ *for* $x \in \{a,b,c,d\}$. *The four singleton assumption sets* $\{Mx\}$ *are all conflict-free. For instance,* $\neg a$ *is not derivable from* W *and* Ma, *i.e.,* $W \cup \{Ma\} \nvdash \neg a$. *Using only rules from* $\mathscr{R}_p$ *(representing classical propositional entailment),* $\neg a$ *cannot be derived from* $W \wedge Ma$. *However,* Ma *can be derived and thereby via* r_1 *the atom* a. *Thus* $W \cup \{Ma\} \vdash a$. *Moreover,* $\neg b$ *is derived via the first clause in* W. *Thus* $\{Ma\}$ *attacks* $\{Mb\}$. *Symmetrically,* $\{Mb\}$ *attacks* $\{Ma\}$. *Only assumption sets including* Mb *derive* $\neg a$. *Thus* $\{Ma\}$ *attacks all assumption sets that attack* $\{Ma\}$, *indicating that* $\{Ma\}$ *is admissible.* $\{Md\}$ *does not attack any other set and is not attacked, as* d *does not occur anywhere outside* r_4. *As all other sets are attacked,* $\{Md\}$ *is the (unique) grounded assumption set. A com-*

plete and stable assumption set is $\{Ma, Md\}$. It holds that a is credulously accepted under complete semantics, since the complete assumption set $\{Ma, Md\}$ derives a. It holds that $\{Mb, Md\}$ (another complete assumption set) does not derive a. Thus a is not skeptically accepted. Finally, rule r_4 does not trigger: its prerequisite is never entailed by a conflict-free assumption set ($a \wedge b$ is unsatisfiable with W).

The complexity of ABA instantiated with propositional default logic was investigated in [13,11,19]. For flat ABA instantiated with a deductive system whose derivation complexity is in coNP, we have Σ_2^P-completeness for credulous reasoning under admissible and stable semantics (and since credulous acceptance under admissible and complete semantics coincides [2], also for complete semantics). Skeptical acceptance under admissibility is coNP-complete and Π_P^2-complete under stable semantics. The complexity of reasoning under grounded semantics (and skeptical acceptance for complete semantics) is in Δ_P^2 (i.e., decidable via a deterministic polynomial time algorithm with access to an NP oracle). Furthermore, the complexity of credulous and skeptical reasoning in propositional default logic (i.e., deciding whether a formula is entailed by one or all extensions according to Reiter [12]) coincides with the complexity of stable semantics [13].

3. A SAT-Based Approach to Deciding DL-ABA

Our SAT-based approach to deciding DL-ABA under different argumentation semantics is based on counterexample-guided abstraction refinement [20] which has earlier proven successful in the realm of abstract argumentation (see, e.g., [21,22]). For high-level intuition, our algorithms work by iteratively first "guessing" with a SAT solver a candidate assumption set A, and then employing a SAT solver to determine the set C of conclusions of default rules in D that are applicable by A. After this, we employ a SAT solver to check if A conforms to the semantics of interest (together with a query, in case of acceptance problems) based on C. If this is the case, the search terminates. If not, a counterexample witnessing this fact is obtained, and we proceed to the next iteration to guess another candidate assumption set A. The counterexamples obtained are used for further restricting (or "refining") the set of candidate assumption sets that will be considered in the forthcoming iterations, depending on the semantics at hand.

We will continue by describing the approach in more detail, including how the candidate assumption sets are guessed, conclusions determined, counterexamples obtained, and refinements made. As convenient, we will treat a set of propositional formulas interchangeably as the conjunction of the formulas the set contains.

The following observation is central to our approach. For a given ABF $F = (W, D)$, we define a function that, given a subset $S \subseteq \mathscr{A}$ of assumptions, iteratively constructs the set of conclusions of defaults in D that are applicable by S. Let $derived_S(X) = \{c \mid (c \leftarrow a, Mb_1, \ldots, Mb_n) \in D$ where $Mb_1, \ldots, Mb_n \in S$ and $W \cup \{\bigwedge_{c' \in X} c'\} \models a\}$. For a given S, it holds that $derived_S$ is $\subseteq$-monotone; this follows by monotonicity of classical propositional entailment $\models$. Further, a unique least fixed-point exists and can be computed by iteratively applying $derived_S$ starting with $X = \emptyset$, i.e., $derived_S^i(\emptyset)$ for some $i \geq 0$ reaches the least fixed point. This fact can be directly inferred since D is assumed to be finite: $derived_S^i(\emptyset)$ reaches some fixed-point after some number of iterations, and the result of each iteration must be a subset of each fixed-point of $derived_S$ (e.g., it holds that $derived_S(\emptyset)$ is part of each fixed-point, and then also $derived_S^1(\emptyset)$ must be, and so on).

Proposition 1. *Suppose an ABA framework $F = (\mathscr{L}, \mathscr{R}, W, \mathscr{A}, ^-)$, a set of assumptions $A \subseteq \mathscr{A}$ and a sentence $x \in \mathscr{L}$. Let X be the least fixed point of $derived_A$ and $C = \bigwedge_{c \in X} c$. It holds that x is derivable from A in F if and only if either $x \in A$ or the propositional formula $W \wedge C \wedge \neg x$ is not satisfiable.*

Proof. We first show that $y \in X$ implies that y is derivable from A in F via induction on $derived_A^i(\emptyset)$ with $i \geq 1$. For the base case $i = 1$, since $y \in derived_A(\emptyset)$, there is a rule $y \leftarrow a, Mb_1, \ldots, Mb_n$ with $\{Mb_1, \ldots, Mb_n\} \subseteq A$ and $W \models a$. Thus there is a derivation of a from W in F. For the induction step, assume that $y \in derived_A^i(\emptyset)$ implies y is derivable from A in F. We need to show that $z \in derived_A^{i+1}(\emptyset)$ implies z is derivable from A in F. Then there is, again, a rule $z \leftarrow a, Mb_1, \ldots, Mb_n$ with $\{Mb_1, \ldots, Mb_n\} \subseteq A$. It must hold that a is entailed by $W \wedge \bigwedge_{c \in derived_A^i(\emptyset)} c$, implying that there is a derivation in F for a from A and, in turn, also for z. Thus, $y \in X$ implies y is derivable from A in F. Now assume that $W \wedge C \wedge \neg x$ is not satisfiable, and thus $W \wedge C \models x$. Each conjunct c of C is a subset of X and thus, as shown, derivable from A. Therefore x is derivable from A in F.

Suppose x is derivable from A in F. Then there is a sequence of rules in $\mathscr{R}$ s.t. each body element of each rule is either in A or the head of a rule previously in the sequence. We can assume that the derivation is of finite length, since there are only finitely many default rules, and a derivation in a sound and complete inference system in propositional logic can be assumed to be only requiring finitely many steps. Consider again an inductive line of reasoning on the length of the derivation. The base case is straightforward: the body of the first rule is in A or W, and, in turn, the head is entailed by $W \wedge C$. Assume that up to i each head of preceding rules is entailed by $W \wedge C$. Independently of the rule being a default, the body of the rule is either entailed by $W \wedge C$ (from previous rules and induction hypothesis) or in A, implying the claim. □

We continue by detailing our algorithmic approach. Apart from W, central to the approach is a propositional formula ϕ over all of the assumptions Ma of the framework in question, used to guess candidate assumption sets. Initially ϕ is the empty formula, and ϕ is expanded at each refinement step conjunctively with further propositional clauses. Following Proposition 1, W is used to decide derivations and thus attacks from any given assumption set. In the following, for a truth assignment I we let $I_{\mathscr{A}} = \{Ma \mid Ma \in \mathscr{A} \cap I\}$ to denote the set of assumptions that are assigned to *true* by I.

Algorithm 1 gives the skeleton for credulous reasoning under stable, admissible and complete semantics. Recall that ϕ is a conjunction of *refinement clauses* ruling out provably incorrect candidate solutions (initially the empty formula). A candidate assumption

Algorithm 1 Credulous reasoning: skeleton for stable, admissible and complete

Require: ABA framework (W, D) and a query $q \in \mathscr{L}_p$
Ensure: return YES if q is credulously justified under given semantics, NO otherwise

```
Let φ be an empty propositional formula over 𝒜
while I ← Sat(φ) do
    C ← Concluded_via_Defaults(I_𝒜)
    if CF_Derive_Query(I_𝒜, C, q) then
        if ¬Counterexample(I, C) then return YES
        φ ← Refine(I)
return NO
```

Algorithm 2 Concluded_via_Defaults(A)

Require: $A \subseteq \mathscr{A}$
Ensure: return the conclusions of the default rules that are applicable by A.
1: $X \leftarrow \{r \in X \mid M(r) \subseteq A\}$
2: $C \leftarrow \top$
3: *changes* $\leftarrow$ **true**
4: **while** *changes* **do**
5: *changes* $\leftarrow$ **false**
6: $X' \leftarrow X$
7: **while** $X' \neq \emptyset$ **do**
8: *testrule* $\leftarrow pop(X')$
9: **if** $I \leftarrow \text{Sat}(W \wedge C \wedge \neg prereq(testrule))$ **then** $X' \leftarrow X' \setminus \{r \in X \mid \neg prereq(r) \in I\}$
10: **else**
11: $C \leftarrow C \wedge conc(testrule)$
12: Remove *testrule* from X
13: *changes* $\leftarrow$ **true**
14: **break**
15: **return** C

set $I_{\mathscr{A}}$ is guessed by constraining the solution space with the refinement clauses. As explained further in what follows, the following subroutines are used in the algorithms. Here C is the least fixed point of $derived_A$ for assumption set A (Algorithm 2).

- notDerivable(C,q): invokes a SAT solver on $W \wedge C \wedge (\neg q)$, true iff q is not derivable from A.
- ExistUndefeated(B,C): invokes a SAT solver on $W \wedge C \wedge \bigvee_{Ma \in B} a$, true iff there is an assumption in B not attacked by A.
- Attacked(B,C): returns the set of assumptions in B attacked by A (Algorithm 4).

On Line 3 of Algorithm 1, the rules applicable by $I_{\mathscr{A}}$ are determined, and the conclusions of those rules are conjoined as C. Line 4 checks if $I_{\mathscr{A}}$ is conflict-free and the query is derived from $I_{\mathscr{A}}$; both checks use W with every element of C enforced to hold. If those checks succeed, the existence of a counterexample is checked on Line 5. The form of the counterexample depends on the semantics as detailed later on. If a counterexample is not found, $I_{\mathscr{A}}$ is a σ-assumption set that derives q. Otherwise a refinement is added, again depending on semantics; see below.

The details of the subroutines Concluded_via_Defaults, CF_Derive_Query, and Attacked are provided as Algorithms 2, 3 and 4, respectively. Given an assumption set A, Algorithm 2 computes the least fixed-point of $derived_A$ (recall Proposition 1). The rules whose justifications is a subset of A are collected to X (Line 1). C is a conjunction over

Algorithm 3 CF_Derive_Query(I,C,q)

Ensure: return YES if $I_{\mathscr{A}}$ is conflict-free and derives q
1: **if** Attacked$(I_{\mathscr{A}},C) \neq \emptyset$ **or** notDerivable(C,q) **then**
2: $\phi \leftarrow$ Refine(I)
3: **return** NO
4: **return** YES

Algorithm 4 Attacked(B,C)

Ensure: return the elements of B that are attacked by A
1: $X \leftarrow \{x \mid Mx \in B\}$
2: **while** X not empty **do**
3: **if** $I \leftarrow \mathrm{Sat}(W \wedge C \wedge \bigvee_{x \in X} x)$ **then** $X \leftarrow X \setminus I$ **else return** $\{Mx \mid x \in X\}$
4: **return** $\{Mx \mid x \in X\}$

the set of conclusions of rules in X that are in the deductive closure; initially C is empty (Line 2). In the loop starting on Line 4 it is iteratively checked for rules in X if their prerequisite is entailed by $W \wedge C$. If not, all rules whose prerequisite occurs negatively in the found assignment I are removed from consideration for this loop, because none of those prerequisites are entailed (Line 9). If a rule is entailed, the conclusion of this rule is added to C (Line 11). In this case the rule is removed from further consideration as it is already determined to be applicable (Line 12). This rule being applicable changes what $W \wedge C$ entails, and thus the algorithm will not terminate (Line 13). Instead, each rule not already found to be applicable is considered in the next iteration. This is continued until no more prerequisites of rules in X are entailed.

Algorithm 3 checks if the candidate is conflict-free and derives the query. Concretely it returns NO and applies the appropriate refinements if there is an attack from the candidate to itself, checked with Algorithm 4, or the query is not entailed by $W \wedge C$. Given assumption sets A and B, and the conclusions of default rules applicable by A, Algorithm 4 considers the contents of assumptions in B (Line 1), and iteratively removes from consideration those contents of B whose negation $W \wedge C$ does not entail, as this implies that A does not derive them. The rest of the assumptions are attacked by A. If the call on Line 3 is unsatisfiable, then the set of attacked assumptions returned on Line 3 is not empty, otherwise the empty set is returned on Line 4.

3.1. Counterexamples

Counterexample(I,C) is defined as follows for the individual semantics. These subroutines determine if a conflict-free assumption set A is a σ-assumption set under stable, admissible, or complete semantics, respectively. They employ W and C (the least fixed point of $derived_A$) as computed in Algorithm 2.

- Stable: return ExistUndefeated$(\mathscr{A} \setminus I_{\mathscr{A}},C)$. That is, it is checked whether there is an assumption that is not in the current candidate A and not defeated by it.
- Admissibility: return true if Attacked$(I_{\mathscr{A}},C') \neq \emptyset$, where $U = \mathscr{A} \setminus$ Attacked$(\mathscr{A} \setminus I_{\mathscr{A}},C)$, and $C' =$ Concluded_via_Defaults(U). That is, collect all assumptions U not defeated by A, and then check that U does not defeat A.
- Complete: with U and C' as above, return true if either Attacked$(I_{\mathscr{A}},C') \neq \emptyset$ or ExistUndefeated$(\mathscr{A} \setminus I_{\mathscr{A}},C')$ is true. That is, A is not complete if it is not admissible or there is an assumption outside A that is not attacked by the assumptions that A does not attack.

3.2. Refinements

The details of the refinement step made when a counterexample is found depend on the semantics at hand. In particular, we make use of the observations detailed in Proposition 2

to obtain in cases stronger refinements—ruling out more than one candidate from further consideration at forthcoming iterations of Algorithm 1 than what is be achieved by the so-called trivial refinement, consisting of ruling out only the particular candidate for which a counterexample is obtained from further consideration.

Proposition 2. *Given an ABA framework* $F = (\mathscr{L}, \mathscr{R}, W, \mathscr{A}, {}^{-})$ *and an assumption set* $A \subseteq \mathscr{A}$*, it holds that*

1. *if* A *is not conflict-free, then no* $A' \supseteq A$ *is conflict-free,*
2. *if* A *is conflict-free but not stable, then no* $A' \subseteq A$ *is stable,*
3. *if* A *does not derive* x*, then no* $A' \subseteq A$ *derives* x*, and an* A' *that derives* x *must cover the justifications of at least one default rule not covered by* A*,*
4. *if* A *derives* x*, then all* $A' \supseteq A$ *derive* x*, and*
5. *if* A *is conflict-free but not admissible, then no superset of the assumptions not defeated by* A *is admissible.*

Proof. Item 1: the same self-attack is present in $A' \supseteq A$ as in A. For Item 2, suppose that A is conflict-free but not stable, implying that there is an $a \in \mathscr{A} \setminus A$ that A does not attack. Derivability is $\subseteq$-monotone, implying that a is not attacked by any subset of A either. The first part of Item 3 follows from the same observation for an A that does not derive x. The second part follows from the fact that to increase what A derives, one needs to apply additional default rules. Item 4: the same derivation for x exists from any $A' \supseteq A$ than from A. Lastly, for Item 5 assume that A is conflict-free but not admissible, implying that there is an attack from the set U of assumptions not defeated by A to A. Since A is conflict-free, $A \subseteq U$. Thus U is self-defeating and the claim follows from Item 1. □

Refine(I) is defined as follows for the different cases of a candidate being discarded, based on Proposition 2. Here U is the set of assumptions not defeated by $I_{\mathscr{A}}$.

Refinement	Required	Optional
A is not conflict-free	$\bigvee_{Ma \in I_{\mathscr{A}}} \neg Ma$	
Query not deriv. from A (cred.)	$\bigvee_{Ma \in \mathscr{A} \setminus I_{\mathscr{A}}} Ma$	$\bigvee_{r \in D,\ M(r) \not\subseteq I} \bigwedge_{Mb_i \in M(r)} Mb_i$
Query deriv. from A (skept.)	$\bigvee_{Ma \in I_{\mathscr{A}}} \neg Ma$	
A is not stable	$\bigvee_{Ma \in \mathscr{A} \setminus I_{\mathscr{A}}} Ma$	
A is not admissible	$\bigvee_{Ma \in \mathscr{A} \setminus I_{\mathscr{A}}} Ma \vee \bigvee_{Ma \in I_{\mathscr{A}}} \neg Ma$	$\bigvee_{Ma \in U} \neg Ma$
A is not complete	$\bigvee_{Ma \in \mathscr{A} \setminus I_{\mathscr{A}}} Ma \vee \bigvee_{Ma \in I_{\mathscr{A}}} \neg Ma$	

The required clauses for admissible and complete simply rule out the current candidate. We also list optional clauses for a query not being derived from the candidate for credulous reasoning and the candidate not being admissible. The optional clauses are not neceessary for correctness, but are potentially useful for restricting the remaining candidate space further towards earlier termination.

Algorithm 5 Acceptance under grounded

Require: ABA framework $F = (W, D)$ and query $q \in \mathscr{L}_p$
Ensure: return YES if q is credulously justified in F under grounded, NO otherwise
1: $S \leftarrow \emptyset$
2: **while true do**
3: $C \leftarrow$ Concluded_via_Defaults(S)
4: $U \leftarrow \mathscr{A} \setminus$ Attacked$(\mathscr{A}, C)$
5: $C' \leftarrow$ Concluded_via_Defaults(U)
6: *defended* $\leftarrow \mathscr{A} \setminus$ Attacked$(\mathscr{A} \setminus S, C')$
7: **if** *defended* is empty **then break else** $S \leftarrow S \cup$ *defended*
8: **return** $\neg$notDerivable(C, q)

3.3. The Special Case of Grounded Semantics

Algorithm 5 for grounded semantics differs from the other semantics. Here we rely on the fact that DL-ABA frameworks are flat and that the grounded assumption set is the least fixed point of $\mathscr{F}_F(S) = \{Mb \in \mathscr{A} \mid \{Mb\}$ defended by $S\}$ for an $S \subseteq \mathscr{A}$. In other words, the grounded assumption set can be iteratively build in the set S, initially empty, by iteratively adding to it all assumptions defended by it. More concretely, on Line 3 the conclusions of rules applicable by A are identified, and on Line 4 the set of assumptions U not defeated by S is identified. Similarly, on Lines 5 and 6 the conclusions of rules applicable by U and assumptions defended by S are identified; any assumption that is not attacked by U is defended by S, because S counters all attacks not originating from U. If S defends no assumptions outside of itself, S is the least fixed point of the defense-operator, and thus the grounded assumption set (Line 7). Finally, q is accepted under grounded semantics if and only if it is derivable from S (Line 8).

3.4. Skeptical Reasoning and Enumeration

Algorithm 1 requires only minor modifications for deciding skeptical acceptance by checking for the existence of a counterexample instead of set deriving the query. In particular, negate the entailment check in Algorithm 3 and invert the answer given by Algorithm 1. Then the subroutine returns NO if the query is entailed by $W \wedge C$, since then the candidate derives the query and thus can not constitute a counterexample. To enumerate all σ-assumption-sets, remove the entailment check of Algorithm 3 and when a solution is found, instead of terminating collect the solution and continue by calling Refine(I).

To query for an assumption Mb instead of member of $\mathscr{L}_p$, it suffices to initialize ϕ with the assertion that Mb holds and omit the entailment check in Algorithm 3.

As there is guaranteed to be a unique grounded assumption set in a flat ABA framework (including any DL-ABA framework), Algorithm 5 answers skeptical acceptance as such. To enumerate (more precisely, return the single) grounded assumption set, S should be returned on Line 8 instead of performing the entailment check.

4. Experiments

We provide first empirical results on the runtime performance of our SAT-based approach to DL-ABA, focusing on credulous and skeptical reasoning. We implemented the algo-

rithms (available at https://bitbucket.org/coreo-group/satfordl-aba) using the PySAT Python interface [23] and Glucose 3 [24] as the SAT solver. We were unable to run earlier systems [15,16,17,18] for finding extensions in DL (specifically for finding stable extensions in DL-ABA) for a comparison due to unavailability and compilation issues. The experiments were run on 2.60-GHz Intel Xeon E5-2670 8-core 64-GB machines with CentOS 7 under a per-instance 600-s time and 16-GB memory limit.

For benchmarks, we randomly sampled a set of 400 CNF formulas from a large set of real-world SAT instances originally used for benchmarking iterative SAT-based algorithms for the beyond-NP problem of backbone computation [25]. These CNF formulas can be considered suitably challenging for the purpose of our evaluation. For each CNF formula, we chose one literal uniformly at random from the set of positive and negative instances of all variables occuring in the formula, and added default rules to obtain ABFs. Specifically, for each CNF formula, we generated 12 ABFs for a total of 4800 instances, with $a \in \{20, 50, 100, 200\}$ assumptions, each of whose content was randomly selected from the literals in the CNF, and $d \in \{100, 200, 400\}$ default rules per framework. Each rule had one literal selected uniformly at random as its prerequisite and conclusion, respectively, and from one to five assumptions as its justifications, with both the amount and identities of the assumptions selected uniformly at random.

Table 1 gives the number of timeouts and mean runtimes over solved instances wrt. the number of assumptions and rules for the different reasoning tasks for credulous reasoning under stable, admissible, complete and grounded semantics, and for skeptical reasoning under stable semantics.[2] Note that these results are obtained without using the optional refinements for credulous reasoning and admissible semantics (recall Section 3.2). Acceptance under grounded semantics appears the easiest to solve, likely due to the fact that in this algorithm, unlike for the other semantics, we do not need to "guess" assumption sets, but instead build the grounded set iteratively. Acceptance under stable semantics appears harder to solve compared to the other second-level problems, which fits the intuition that there are fewer stable sets, making it harder to find a suitable one. Performance on skeptical acceptance under stable semantics is very similar to credulous acceptance, and likewise for performance under admissible and complete semantics for credulous reasoning. For the semantics other than grounded, there is a clear increase in difficulty of the instances as the number of assumptions and defeasible rules, respectively, increases. For example, for all parameters combinations with at least 100 assumptions, our current implementation timed out on over half of the instances, suggesting that there would be room for further runtime improvements as well as for developing further understanding on what makes individual instances hard to solve.

Figure 1 shows the impact of the number of assumptions on runtime performance and (in the left side plot) the effect of the optional credulous reasoning refinement under stable semantics. Interestingly, the optional refinement clauses appear to degrade runtime performance, especially for instances with 50 assumptions. For a possible explanation, note that while the number of candidates that do not derive the given query is much lower when using the optional refinement, there are in turn more candidates that fail the other criteria (conflict-freeness and stability). Hence the number of iterations may even increase when using the optional refinements. On the other hand, the optional refinement

[2] Skeptical reasoning under admissible semantics is relatively easy as it reduces to checking if the empty assumption set derives the query. Skeptical reasoning under complete semantics coincides with acceptance under grounded semantics. Credulous reasoning under admissible and complete semantics also coincide.

Table 1. Detailed runtime results. There are a total of 400 instances per each combination of $|\mathscr{A}|$ and $|D|$.

		#timeouts (mean runtime over solved instances (s))									
$\|\mathscr{A}\|$	$\|D\|$	*stb* cred		*adm* cred		*com* cred		*grd* accept		*stb* skept	
20	100	20	(55.5)	29	(50.8)	28	(52.6)	9	(11.0)	22	(54.6)
	200	65	(67.9)	66	(55.1)	65	(59.4)	12	(15.7)	66	(62.2)
	400	97	(77.9)	88	(66.3)	92	(60.2)	10	(15.9)	96	(71.3)
50	100	115	(113.0)	108	(83.5)	108	(84.1)	13	(14.7)	118	(117.3)
	200	200	(107.8)	180	(57.2)	180	(58.6)	8	(17.0)	197	(111.4)
	400	264	(48.8)	223	(26.9)	225	(21.6)	13	(19.5)	263	(56.5)
100	100	240	(101.6)	201	(51.7)	201	(55.4)	12	(16.4)	238	(108.8)
	200	285	(61.2)	226	(27.5)	228	(24.4)	14	(18.8)	281	(61.9)
	400	299	(32.3)	235	(15.3)	235	(16.9)	12	(24.1)	301	(32.2)
200	100	293	(45.3)	229	(25.4)	231	(24.2)	13	(21.5)	297	(37.6)
	200	312	(8.4)	239	(7.0)	239	(7.9)	16	(20.7)	310	(24.8)
	400	312	(13.5)	239	(11.0)	240	(12.9)	16	(20.9)	316	(16.7)

clauses for admissibility seem to slightly improve performance on admissible and complete semantics, allowing for solving 1 and 3 more instances, respectively. Figure 1(right) shows that instances with more default rules are generally harder to solve under complete and stable semantics (we omit here admissible due to similar performance to complete and grounded due to the relative easiness of the task).

5. Conclusions

We developed an algorithmic approach, based on iterative applications of Boolean satisfiability (SAT) solvers, to reasoning in ABA instantiated with propositional default

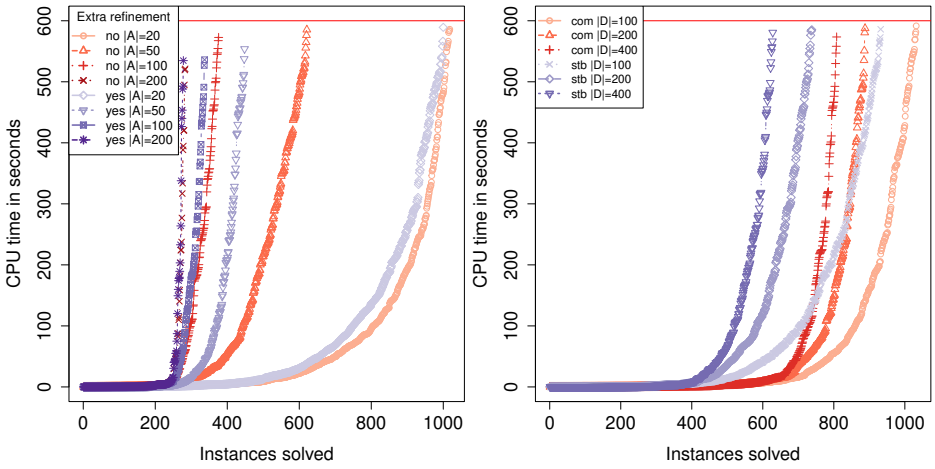

Figure 1. Left: credulous reasoning under stable semantics with and without the optional refinement clauses. Right: credulous reasoning under complete and stable semantics. 1200 instances for each variant.

logic (DL). We detailed instantiations of the approach for deciding acceptance and for assumption-set enumeration in the DL-instantiation of ABA under several central argumentation semantics. We also provided an implementation of the approach which to the best of our understanding is the first system in its generality targeted at the DL instantiation of ABA, and empirically evaluated its potential.

References

[1] Bondarenko A, Dung PM, Kowalski RA, Toni F. An Abstract, Argumentation-Theoretic Approach to Default Reasoning. Artif Intell. 1997;93:63-101.

[2] Čyras K, Fan X, Schulz C, Toni F. Assumption-Based Argumentation: Disputes, Explanations, Preferences. In: Handbook of Formal Argumentation. College Publications; 2018. p. 365-408.

[3] Modgil S, Prakken H. A general account of argumentation with preferences. Artif Intell. 2013;195:361-97.

[4] Besnard P, Hunter A. Elements of Argumentation. MIT Press; 2008.

[5] García AJ, Simari GR. Defeasible Logic Programming: An Argumentative Approach. Theory Pract Log Program. 2004;4(1-2):95-138.

[6] Gaertner D, Toni F. CaSAPI: A system for credulous and sceptical argumentation. In: Proc. NMR; 2007. p. 80-95.

[7] Thimm M. Tweety: A Comprehensive Collection of Java Libraries for Logical Aspects of Artificial Intelligence and Knowledge Representation. In: Proc. KR. AAAI Press; 2014. p. 528-37.

[8] Craven R, Toni F. Argument graphs and assumption-based argumentation. Artif Intell. 2016;233:1-59.

[9] Lehtonen T, Wallner JP, Järvisalo M. Declarative Algorithms and Complexity Results for Assumption-Based Argumentation. J Artif Intell Res. 2021;71:265-318.

[10] Lehtonen T, Wallner JP, Järvisalo M. Harnessing Incremental Answer Set Solving for Reasoning in Assumption-Based Argumentation. Theory Pract Log Program. 2021;21(6):717-34.

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

[12] Reiter R. A Logic for Default Reasoning. Artif Intell. 1980;13(1-2):81-132.

[13] Gottlob G. Complexity Results for Nonmonotonic Logics. J Log Comput. 1992;2(3):397-425.

[14] Biere A, Heule M, van Maaren H, Walsh T, editors. Handbook of Satisfiability, 2nd edition. vol. 336 of Frontiers in Artificial Intelligence and Applications. IOS Press; 2021.

[15] Schaub T, Brüning S. Prolog Technology for Default Reasoning: Proof Theory and Compilation Techniques. Artif Intell. 1998;106(1):1-75.

[16] Cholewinski P, Marek VW, Truszczynski M. Default Reasoning System DeReS. In: Proc. KR. Morgan Kaufmann; 1996. p. 518-28.

[17] Nicolas P, Saubion F, Stéphan I. GADEL: a Genetic Algorithm to Compute Default Logic Extensions. In: Proc. ECAI. IOS Press; 2000. p. 484-90.

[18] Chen Y, Wan H, Zhang Y, Zhou Y. dl2asp: Implementing Default Logic via Answer Set Programming. In: Proc. JELIA. vol. 6341 of LNCS. Springer; 2010. p. 104-16.

[19] Cyras K, Heinrich Q, Toni F. Computational complexity of flat and generic Assumption-Based Argumentation, with and without probabilities. Artif Intell. 2021;293:103449.

[20] Clarke EM, Gupta A, Strichman O. SAT-based counterexample-guided abstraction refinement. IEEE Trans Comput Aided Des Integr Circuits Syst. 2004;23(7):1113-23.

[21] Dvořák W, Järvisalo M, Wallner JP, Woltran S. Complexity-sensitive decision procedures for abstract argumentation. Artif Intell. 2014;206:53-78.

[22] Wallner JP, Niskanen A, Järvisalo M. Complexity Results and Algorithms for Extension Enforcement in Abstract Argumentation. J Artif Intell Res. 2017;60:1-40.

[23] Ignatiev A, Morgado A, Marques-Silva J. PySAT: A Python Toolkit for Prototyping with SAT Oracles. In: Proc. SAT. vol. 10929 of LNCS. Springer; 2018. p. 428-37.

[24] Audemard G, Simon L. On the Glucose SAT Solver. Int J Artif Intell Tools. 2018;27(1):1840001:1-1840001:25.

[25] Janota M, Lynce I, Marques-Silva J. Algorithms for computing backbones of propositional formulae. AI Commun. 2015;28(2):161-77.