TPLP **21** (6): 717–734, 2021.

doi:10.1017/S1471068421000296

Harnessing Incremental Answer Set Solving for Reasoning in Assumption-Based Argumentation[*]

TUOMO LEHTONEN
University of Helsinki, Helsinki, Finland
(*e-mail:* tuomo.lehtonen@helsinki.fi)

JOHANNES P. WALLNER
Graz University of Technology, Graz, Austria
(*e-mail:* wallner@ist.tugraz.at)

MATTI JÄRVISALO
University of Helsinki, Helsinki, Finland
(*e-mail:* matti.jarvisalo@helsinki.fi)

revised 10 August 2021; accepted 22 August 2021

Abstract

Assumption-based argumentation (ABA) is a central structured argumentation formalism. As shown recently, answer set programming (ASP) enables efficiently solving NP-hard reasoning tasks of ABA in practice, in particular in the commonly studied logic programming fragment of ABA. In this work, we harness recent advances in incremental ASP solving for developing effective algorithms for reasoning tasks in the logic programming fragment of ABA that are presumably hard for the second level of the polynomial hierarchy, including skeptical reasoning under preferred semantics as well as preferential reasoning. In particular, we develop non-trivial counterexample-guided abstraction refinement procedures based on incremental ASP solving for these tasks. We also show empirically that the procedures are significantly more effective than previously proposed algorithms for the tasks.

KEYWORDS: answer set programming, incremental answer set solving, assumption-based argumentation, structured argumentation, algorithms, experimental evaluation

1 Introduction

Argumentation, and in particular the study of computational models of argument, constitutes a core research area in artificial intelligence, and knowledge representation and nonmonotonic reasoning in particular (Baroni *et al.* 2018). Computational models for structured argumentation, as opposed to abstract argumentation, make the internal structure of arguments explicit, supporting the view that arguments are most often made explicit

* Work financially supported by Academy of Finland (grant 322869), University of Helsinki Doctoral Programme in Computer Science DoCS, and the Austrian Science Fund (FWF): P30168-N31 and I2854.

through derivations from more basic structures, thereby having an intrinsic structure. Various structured argumentation formalisms have been proposed, each with their own features and applications (Čyras *et al.* 2018; Modgil and Prakken 2018; Besnard and Hunter 2018; García and Simari 2018).

In this work we focus on the commonly studied logic programming fragment of assumption-based argumentation (ABA) (Čyras *et al.* 2018) and its extension, ABA$^+$, equipped with preferences (Čyras and Toni 2016a). These central structured argumentation formalisms have found applications for example, in decision making in a multi-agent context (Fan *et al.* 2014), game theory (Fan and Toni 2016), and in choosing treatment recommendations based on clinical guidelines and preferential information given by patients (Čyras and Oliveira 2019). In addition to applications, the challenge of developing efficient systems for ABA reasoning is also highlighted by the 2021 ICCMA argumentation system competition, which for the first time called for ABA reasoning systems for several NP-hard problems.

Among the algorithmic approaches proposed for reasoning in ABA (Dung *et al.* 2006; Gaertner and Toni 2007; Toni 2013; Craven *et al.* 2013; Craven and Toni 2016; Lehtonen *et al.* 2017; 2021a), in terms of scalability arguably the currently most efficient practical approach is based on encoding ABA reasoning tasks declaratively using answer set programming (ASP) (Gelfond and Lifschitz 1988; Niemelä 1999), and invoking off-the-shelf ASP solvers for the reasoning part (Lehtonen *et al.* 2021a; Caminada and Schulz 2017). While this approach is noticeably more efficient than other competing ABA reasoning systems on NP-complete variants of ABA reasoning, the approach was not directly extended to cover all *beyond-NP* variants of ABA reasoning, that is, reasoning tasks which are presumably hard for the second-level of the polynomial hierarchy. In particular, skeptical acceptance in ABA under preferred semantics was treated resorting to the so-called Asprin approach, although a direct treatment would be viable. Further, credulous reasoning in ABA$^+$ under admissible and complete semantics was not covered.

Motivated by the success of ASP-based ABA reasoning, in this work we harness very recent advances in incremental ASP solving (Kaminski *et al.* 2020; Gebser *et al.* 2011) for developing counterexample-guided abstraction refinement (CEGAR) (Clarke *et al.* 2003; 2004) style algorithms for skeptical reasoning in ABA under preferred semantics, as well as credulous reasoning in ABA$^+$ under admissible and complete semantics. Compared to the currently existing ABA reasoning systems supporting these tasks, in particular the Asprin-based (Brewka *et al.* 2015) approach to reasoning in ABA under preferred semantics (Lehtonen *et al.* 2021a), and the ABAplus system (Bao *et al.* 2017) for enumerating admissible and complete assumption sets in ABA$^+$, our approach provides significant performance improvements in practice and allows for directly reasoning about credulous acceptance in ABA$^+$. Our implementation is available at https://bitbucket.org/coreo-group/aspforaba.

2 Assumption-based argumentation

We recall ABA (Bondarenko *et al.* 1997; Toni 2014; Čyras *et al.* 2018) and ABA$^+$ (Čyras and Toni 2016a;b; Bao *et al.* 2017; Čyras 2017) which extends ABA with preferences over assumptions. We define ABA$^+$ frameworks, as ABA$^+$ is a generalization of ABA.

We focus on the commonly studied logic programming fragment of ABA and ABA$^+$. In particular, we assume a deductive system $(\mathcal{L}, \mathcal{R})$ with $\mathcal{L}$ a set of atoms and $\mathcal{R}$ a set of inference rules over $\mathcal{L}$ with a rule $r \in \mathcal{R}$ having the form $a_0 \leftarrow a_1, \ldots, a_n$ with $a_i \in \mathcal{L}$. To distinguish ABA atoms from atoms in ASP, we refer to the former as sentences. We denote the head of rule r by $\mathit{head}(r) = \{a_0\}$ and the (possibly empty) body of r by $\mathit{body}(r) = \{a_1, \ldots, a_n\}$.

An ABA$^+$ framework is a tuple $F = (\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-, \leq)$ with $(\mathcal{L}, \mathcal{R})$ a deductive system, a set of assumptions $\mathcal{A} \subseteq \mathcal{L}$, a function $^-$ mapping assumptions $\mathcal{A}$ to sentences $\mathcal{L}$, and a preorder $\leq$ on $\mathcal{A}$. The strict counterpart $<$ of $\leq$ is defined as usual by $a < b$ iff $a \leq b$ and $b \not\leq a$, for $a, b \in \mathcal{A}$. An ABA framework, that is ABA$^+$ without preferences, is an ABA$^+$ framework with $\leq\, = \emptyset$, denoted by $(\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-)$. In this paper, we focus on so-called flat ABA$^+$ frameworks where assumptions cannot be derived, that is, do not occur in heads of rules. We assume that each set in an ABA$^+$ is finite.

There are two notions of derivations for ABA: tree-derivations and forward-derivations. ABA$^+$ is in general defined through tree-derivations. We briefly recall both notions. A sentence $s \in \mathcal{L}$ is tree-derivable from a set of assumptions $X \subseteq \mathcal{A}$ and rules $R \subseteq \mathcal{R}$, denoted by $X \models_R s$, if there is a finite tree with the root labeled by s, the leaves labeled by elements of X, and for each internal node there is a rule $r \in R$ such that the node itself is labeled by $\mathit{head}(r)$ and the set of labels of the children of this node is $\mathit{body}(r)$. For each rule $r \in R$ there is a node labeled in this way. For brevity, R can be left unspecified and be assumed to be some suitable subset of $\mathcal{R}$. A sentence $a \in \mathcal{L}$ is forward-derivable from a set $X \subseteq \mathcal{A}$ via rules $\mathcal{R}$, denoted by $X \vdash_{\mathcal{R}} a$, if there is a sequence of rules $(r_1, \ldots, r_n)$ such that $\mathit{head}(r_n) = a$, for each rule r_i we have $r_i \in \mathcal{R}$, and each sentence in the body of r_i is derived from rules earlier in the sequence or is in X, that is, $\mathit{body}(r_i) \subseteq X \cup \bigcup_{j<i} \mathit{head}(r_j)$. The deductive closure for an assumption set X w.r.t. rules $\mathcal{R}$ is given by $\mathit{Th}_{\mathcal{R}}(X) = \{a \mid X \vdash_{\mathcal{R}} a\}$.

Definition 1
Let $(\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-, \leq)$ be an ABA$^+$ framework, and $A, B \subseteq \mathcal{A}$ be two sets of assumptions. A $<$-attacks B if

- $A' \models_R \overline{b}$ for some $A' \subseteq A$, $b \in B$, and $\nexists a' \in A'$ with $a' < b$, or
- $B' \models_R \overline{a}$ for some $a \in A$ and $B' \subseteq B$ s.t. $\exists b' \in B'$ with $b' < a$.

In words, set A attacks B if (i) from a subset A' of A, one can tree-derive a contrary of an assumption $b \in B$ and no member in A' is strictly less preferred than b, or (ii) from B, via subset B' one can tree-derive a contrary of an assumption $a \in A$ and some member of B' is strictly less preferred than a. Attacks of type (i) are *normal* $<$-attacks and those of type (ii) *reverse* $<$-attacks, with the intuition that the (non-preference based) conflict in (i) succeeds and in case of (ii) is countered and reversed by the preference relation. For brevity, we omit set notation when A $<$-attacks a singleton $\{b\}$ (then we say A $<$-attacks b).

Definition 2
Let $F = (\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-, \leq)$ be an ABA$^+$ framework. An assumption set $A \subseteq \mathcal{A}$ is called conflict-free if A does not $<$-attack itself. Set A defends assumption set $B \subseteq \mathcal{A}$ if for all $C \subseteq \mathcal{A}$ that $<$-attack B it holds that A $<$-attacks C.

Definition 3
Let $F = (\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-, \leq)$ be an ABA$^+$ framework. Further, let $A \subseteq \mathcal{A}$ be a conflict-free set of assumptions in F. Set A is

- $<$-*admissible* in F if A defends itself;
- $<$-*complete* in F if A is admissible in F and contains every assumption set defended by A;
- $<$-*preferred* in F if A is $<$-admissible and there is no $<$-admissible set of assumptions B in F with $A \subset B$.

We use the term $<$-σ *assumption set* for an assumption set under a semantics $\sigma \in \{adm, com, prf\}$, that is, $<$-admissible, $<$-complete, and $<$-preferred assumption set, respectively.

For ABA, we refer to the corresponding semantics without the preference relation $<$ (e.g. complete semantics instead of $<$-complete semantics). Attacks ($\emptyset$-attacks) in ABA frameworks simplify to attacks from A to B when $A \vdash_{\mathcal{R}} \overline{b}$ for $b \in B$.

Main reasoning tasks on ABA$^+$ are the following.

Definition 4
Let $F = (\mathcal{L}, \mathcal{R}, \mathcal{A}, {}^-, \leq)$ be an ABA$^+$ framework and $<$-σ a semantics. A sentence $s \in \mathcal{L}$ is

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

The tasks for ABA are analogous (disregarding $<$).

Example 1
Let F be an ABA$^+$ framework with $\mathcal{A} = \{a, b, c\}$, $\overline{b} = x$, $\overline{c} = y$, $a < c$, and $\mathcal{R} = \{(x \leftarrow a), (y \leftarrow a)\}$. We have $\{a\}$ normally $<$-attacks b and $\{c\}$ reversely $<$-attacks a. The $<$-admissible sets of F are $\emptyset$, $\{c\}$, and $\{b, c\}$, and the framework has the unique $<$-complete set $\{b, c\}$.

We focus on computationally hard reasoning tasks in ABA and ABA$^+$. Deciding skeptical acceptance under preferred semantics in ABA is Π_2^P-complete (Dimopoulos *et al.* 2002). In ABA$^+$, credulous acceptance under $<$-admissible semantics is Σ_2^P-complete, and checking whether a set is $<$-admissible and $<$-complete is coNP-complete and coNP-hard, respectively (Lehtonen *et al.* 2021a).

3 Algorithms

We present ASP-based CEGAR algorithms for beyond-NP reasoning tasks in ABA and ABA$^+$. The CEGAR-based algorithms follow the iterative schema of considering an NP-abstraction of the solution space (containing spurious solutions), and drawing candidates from this space. At each iteration, a candidate solution is obtained (if one remains) by calling an ASP solver. If no further candidate solutions can be obtained, the search terminates. If a candidate is obtained, one checks with another ASP solver call whether the candidate is an actual solution. If it is, the search terminates. If not, a counterexample is obtained, and the abstraction is refined (solution space is reduced) by ruling out from further consideration at least the candidate solution.

We briefly recap basic ASP concepts. An answer set program π consists of rules r of the form $h \leftarrow b_1, \ldots, b_k, not\, b_{k+1}, \ldots,\ not\, b_m$, where h and each b_i is an atom. A literal is an atom or a default negated (not) atom. A rule is positive if $k = m$, a fact if $m = 0$, and a constraint if there is no head h (then a shorthand for the same rule with a fresh atom a in the head and default negated in the body). An atom b_i has the form $p(t_1, \ldots, t_n)$ with p a predicate and with each t_j a constant or a variable. An answer set program, a rule, and an atom, respectively, is ground if it is free of variables. For a non-ground program, GP is the set of rules obtained by applying all possible substitutions from the variables to the set of constants appearing in the program. An interpretation I, that is, a subset of all the ground atoms, satisfies a positive rule $r = h \leftarrow b_1, \ldots, b_k$ iff all positive body elements $b_1, \ldots, b_k$ are in I implies that the head atom is in I. For a program π consisting only of positive rules, let $Cl(\pi)$ be the uniquely determined interpretation I that satisfies all rules in π and no subset of I satisfies all rules in π. Interpretation I is an answer set of a ground program π if $I = Cl(\pi^I)$ where $\pi^I = \{(h \leftarrow b_1, \ldots, b_k) \mid (h \leftarrow b_1, \ldots, b_k, not\, b_{k+1}, \ldots, not\, b_m) \in \pi, \{b_{k+1}, \ldots, b_m\} \cap I = \emptyset\}\}$ is the reduct; and of a non-ground program π if I is an answer set of GP of π. A program π is satisfiable iff there is an answer set of π.

We make use of the following shorthands. Let I be an interpretation (set of ASP atoms), $\mathbf{p}$ be some ASP predicate of arity one, and $M = \{l_1, \ldots, l_n\}$ a set of ASP literals. We define $\mathbf{p}(I) = \{\mathbf{p}(x) \mid \mathbf{p}(x) \in I\}$, and $constr(M) = \leftarrow l_1, \ldots, l_n$. That is, $\mathbf{p}(I)$ is a set of atoms $\mathbf{p}(x)$ which are contained in the interpretation, and $constr(M)$ is an ASP constraint containing M as its body. For reasons of convenience, if l is an ASP literal, we also define the shorthand $constr(l) = \leftarrow l$ (i.e. allowing M to be a set or a single literal).

In the following we refer to forward-derivations when talking about derivations. For ABA, tree and forward-derivations are equivalent for the problems considered here (Dung *et al.* 2006; 2010). While forward-derivations are not directly applicable for ABA$^+$, for our approach to $<$-admissible and $<$-complete semantics we employ previous results (Lehtonen *et al.* 2021a) together with new ones (Section 3.2), which allow for avoiding naive application of tree-derivations which would require explicitly constructing arguments. We present one algorithm per semantics here with a brief note on how to extend to other reasoning tasks; full algorithms for other reasoning tasks can be found online (Lehtonen *et al.* 2021b).

Algorithm 1 Skeptical acceptance under preferred

Require: ABA framework $F = (\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-)$, $s \in \mathcal{L}$

Ensure: return YES if s is skeptically accepted under preferred semantics in F, NO otherwise

```
1: $\pi := \mathtt{ABA}(F) \cup \pi_{com}$
2: **while** $\pi \cup \{constr(\mathbf{supported}(s))\}$ is satisfiable **do**
3:     Let $I$ be the found answer set
4:     $\pi := \pi \cup \{constr(\mathbf{out}(I))\}$
5:     **while** $\pi \cup \{constr(\mathbf{supported}(s))\} \cup \mathbf{in}(I)$ is satisfiable **do**
6:         Let $I$ be the found answer set
7:         $\pi := \pi \cup \{constr(\mathbf{out}(I))\}$
8:     **if** $\pi \cup \mathbf{in}(I)$ is unsatisfiable **then return** NO
9: **return** YES
```

Listing 1. *Module* π_{com}

```
in(X) ← assumption(X), not out(X).
out(X) ← assumption(X), not in(X).
supported(X) ← assumption(X), in(X).
supported(X) ← head(R,X), triggered_by_in(R).
triggered_by_in(R) ← head(R,_), supported(X) : body(R,X).
← in(X), contrary(X,Y), supported(Y).
defeated(X) ← supported(Y), contrary(X,Y).
derived_from_undefeated(X) ← assumption(X), not defeated(X).
derived_from_undefeated(X) ← head(R,X), triggered_by_undefeated(R).
triggered_by_undefeated(R) ← head(R,_), derived_from_undefeated(X) : body(R,X).
attacked_by_undefeated(X) ← contrary(X,Y), derived_from_undefeated(Y).
← in(X), attacked_by_undefeated(X).
← out(X), not attacked_by_undefeated(X).
```

3.1 Skeptical acceptance under preferred semantics

We begin with skeptical reasoning under preferred semantics in ABA, which is a Π_2^P-complete problem. Following successful schemes for the same reasoning task on abstract argumentation frameworks (AFs) (Cerutti *et al.* 2018), we present Algorithm 1 for deciding skeptical acceptance of sentences in an ABA framework. We encode the given ABA framework $F = (\mathcal{L}, \mathcal{R}, \mathcal{A}, ^{-})$ as ASP: assign each rule a unique identifier ($\mathcal{R} = \{r_1, ..., r_n\}$) and let

$$\begin{aligned}\texttt{ABA}(F) =&\{\mathbf{assumption}(a).\mid a \in \mathcal{A}\} \cup \\ &\{\mathbf{head}(i,b).\mid r_i \in \mathcal{R}, b \in \mathit{head}(r_i)\} \cup \\ &\{\mathbf{body}(i,b).\mid r_i \in \mathcal{R}, b \in \mathit{body}(r_i)\} \cup \\ &\{\mathbf{contrary}(a,x).\mid x = \bar{a}, a \in \mathcal{A}\}.\end{aligned}$$

Listing 1 presents module π_{com} for finding complete assumption sets, including a possible queried sentence (Lehtonen *et al.* 2021a). Now I is an answer set of $\texttt{ABA}(F) \cup \pi_{com}$ iff $\{a \mid \mathbf{in}(a) \in I\}$ is a complete assumption set of F, and one can derive the sentence x from this complete assumption set iff $\mathbf{supported}(x) \in I$. Further, $\mathbf{out}(I)$ contains the assumptions of F that are not part of $\mathbf{in}(I)$.

Algorithm 1 decides skeptical acceptance under preferred semantics for ABA frameworks by first generating a complete assumption set (one can also use admissible sets instead of complete sets in this algorithm) within the framework that does not derive the queried sentence s (Line 2). If there is no answer set found in the first application of the while loop in Line 2, then all complete assumption sets of F derive s, and the algorithm terminates. Otherwise, we add to the ASP encoding π the constraint ruling out the complete set encoded in $\mathbf{in}(I)$ as a solution: we add $\mathit{constr}(\mathbf{out}(I))$, which states that at least one atom in $\mathbf{out}(I)$ must not be present in an answer set from now on (excluding $\mathbf{in}(I)$ and its subsets). Subsequently, we iteratively generate proper supersets of a currently found complete assumption set not deriving s (loop starting in Line 5). When this inner loop terminates, we found a complete assumption set that is $\subseteq$-maximal among all complete assumption sets that do not derive s, and π currently contains the last constraint ruling out this particular complete assumption set and its subsets. In Line 8, we check with an ASP solver call whether $\pi \cup \mathbf{in}(I) := \pi \cup \{\mathbf{in}(a). \mid \mathbf{in}(a) \in I\}$ is satisfiable. If it is, there is a complete assumption set that is a proper *superset* of the assumptions encoded in $\mathbf{in}(I)$ and that derives s. In this case $\mathbf{in}(I)$ is not a preferred assumption set and thus not a counterexample to s being skeptically accepted under preferred semantics, so the algorithm proceeds to searching for a new candidate (Line 2). Importantly, π still contains the constraints ruling out any subset of $\mathbf{in}(I)$. On the other hand, if in Line 8 the ASP solver reports unsatisfiability, $\mathbf{in}(I)$ represents a preferred assumption set not deriving s, constituting a counterexample to s being skeptically accepted under preferred semantics. To enumerate all preferred assumption sets, it suffices to omit the query and Line 8, and collect each answer after exiting the inner loop.

The following proposition states the correctness of the approach; correctness follows by the previous discussion on the details of the algorithm and the employed encodings.

Proposition 1
Algorithm 1 decides skeptical acceptance under preferred semantics for ABA frameworks, that is, for a given ABA framework $F = (\mathcal{L}, \mathcal{R}, \mathcal{A}, ^{-})$ and sentence $s \in \mathcal{L}$, Algorithm 1 returns YES if s is skeptically accepted under preferred semantics in F, and NO otherwise.

3.2 ABA$^+$ properties

We move on to ABA$^+$, in this section first giving an alternative characterization of $<$-admissible and $<$-complete semantics to better suit our algorithmic setting, and then showing complexity membership result for $<$-complete semantics. In contrast to ABA (and AFs), credulous acceptance under $<$-admissible and $<$-complete semantics does not coincide (Čyras 2017, Example 3.6). Further, $<$-attacks differ from (non-preference-based) attacks, requiring more complex computation (Lehtonen *et al.* 2021a). We begin with stating conditions for an assumption set to be $<$-admissible or $<$-complete, on which we base our algorithms. Define for an assumption set A the set of assumptions U not individually $<$-attacked by A by $U = \{a \in \mathcal{A} \mid A \text{ does not } <\text{-attack } a\}$.

Proposition 2
Given an ABA$^+$ framework F, a conflict-free set of assumptions A in F, and the set of assumptions U that A does not individually $<$-attack, it holds that

- A is $<$-admissible iff there is no set $B \subseteq U$ such that A does not $<$-attack B and B $<$-attacks A, and
- A is $<$-complete iff A is $<$-admissible and for all $a \in \mathcal{A} \setminus A$ it holds that a is $<$-attacked by some $B \subseteq U$ such that A does not $<$-attack B.

Proof
For the first item, assume that A is $<$-admissible. It follows that if a set B of assumptions $<$-attacks A we have A $<$-attacks B (A defends itself due to admissibility). Thus, there is no B s.t. B $<$-attacks A and A does not $<$-attack B. For the other direction, assume that there is no $B \subseteq U$ such that A does not $<$-attack B and B $<$-attacks A. Suppose there is a set C of assumptions that $<$-attacks A. If $C \cap (\mathcal{A} \setminus U) \neq \emptyset$ (C contains an assumption outside U), then A $<$-attacks C (on a particular assumption, and, due to subset monotonicity of $<$-attacks, also C). Consider the case that $C \cap (\mathcal{A} \setminus U) = \emptyset$. Then $C \subseteq U$ (since $U \subseteq \mathcal{A}$). If A does not $<$-attack C, then we arrive at a contradiction

(contradicts our assumption of the right hand side of the formal statement). Thus, A $<$-attacks C. It follows that A defends itself against all $C \subseteq \mathcal{A}$ that $<$-attack A. Since A is assumed to be conflict-free, it follows that A is $<$-admissible.

For the second item, assume that A is $<$-complete. Then A is $<$-admissible by definition. Let $a \in \mathcal{A} \setminus A$. It follows from definition that A does not defend $\{a\}$. This implies that there is a set B that $<$-attacks $\{a\}$ and A does not $<$-attack B. Since A $<$-attacks any set C with $C \cap U \neq \emptyset$, we have $B \subseteq U$. For the other direction, assume that A is $<$-admissible, and for all $a \in \mathcal{A} \setminus A$ there is a set $B \subseteq U$ such that B $<$-attacks $\{a\}$ and A does not $<$-attack B. Suppose A is not $<$-complete: then there is a set C such that A defends C and $C \nsubseteq A$. Let $c \in C \setminus A$, implying that $c \in \mathcal{A} \setminus A$. Then, by assumption, there is a $B \subseteq U$ such that B $<$-attacks $\{c\}$ and A does not $<$-attack B. This is a contradiction to $\{c\}$ being defended by A. Thus, A is $<$-complete. $\square$

The first item implies that we can focus on assumption sets among U for checking defense. From the second item, it follows that given a $<$-complete assumption set A, for every $a \in U$ (and even for every $a \in \mathcal{A}$) it holds that either $a \in A$, or U $<$-attacks a. This fact can be used to prune candidates for $<$-complete assumption sets.

Complementing earlier results, we show a complexity membership result for credulous acceptance under $<$-complete semantics. The proof uses Proposition 2 and an earlier result (Lehtonen *et al*. 2021a, Proposition 11): after guessing a set of assumptions, checking $<$-admissibility amounts to verifying whether each $<$-attacker is $<$-attacked (in coNP), and checking whether the set contains all defended sets amounts to checking for each individual assumption outside A whether this assumption is not defended (each check in NP).

Theorem 3
Credulous acceptance under $<$-complete semantics in ABA$^+$ is in Σ_2^P.

Proof
Non-deterministically construct a set of assumptions A. Now check whether (i) the queried sentence is derivable from A, (ii) A is $<$-admissible, and (iii) A is $<$-complete. Checking (i) and conflict-freeness can be done in polynomial time. Construct $U = \{u \in \mathcal{A} \mid A \text{ does not } <\text{-attack } u\}$, which is doable in polynomial time (Lehtonen *et al*. 2021a, Proposition 11, item 1). For checking further conditions of $<$-admissibility, check for each set of assumptions $B \subseteq U$ whether B $<$-attacks A without A $<$-attacking B. Checking for two concrete sets of assumptions whether one $<$-attacks the other is doable in polynomial time (Lehtonen *et al*. 2021a, Proposition 11, items 1 and 2). Thus, one can check whether A defends itself via a check in coNP. For checking whether A is also $<$-complete, check for each $a \in \mathcal{A} \setminus A$ whether there is some set $B \subseteq U$ such that B $<$-attacks a and A does not $<$-attack B. This is in NP. These checks establish whether A is $<$-complete, by Proposition 2, and whether the queried sentence is derivable from A, satisfying the definition of credulous acceptance. Overall, this gives a non-deterministic polynomial time algorithm that accesses an NP oracle, showing membership in Σ_2^P for credulous acceptance under $<$-complete semantics. $\square$

Algorithm 2 Credulous acceptance under <-admissible semantics

```
Require: ABA$^+$ framework $F = (\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-, \leq)$
Ensure: return YES if $s$ is credulously accepted under <-admissible semantics in $F$,
    NO otherwise
1: $\pi_{cand} := \texttt{ABA}^+(F) \cup \pi_{cf} \cup \pi^+_{undefeated} \cup \{constr(not\, \textbf{supported}(s))\}$
2: $\pi_{check} := \texttt{ABA}^+(F) \cup \pi^+_{defended} \cup \pi^+_{suspect-defeat}$
3: while $\pi_{cand}$ is satisfiable do
4:     Let $I$ be the found answer set
5:     if $\pi_{check} \cup \textbf{undefeated}(I) \cup \textbf{in}(I)$ is unsatisfiable then return YES
6:     $\pi_{cand} := \pi_{cand} \cup \{constr(\textbf{out}(I) \cup \textbf{in}(I))\}$
7: return NO
```

3.3 <-Admissible semantics

We proceed to algorithms for ABA$^+$, starting with Algorithm 2 for deciding credulous acceptance under <-admissible semantics in a given ABA$^+$ framework F. This algorithm can be straightforwardly extended to cover enumeration of all <-admissible sets. We first give details of the algorithm, and subsequently explanations of the underlying ASP encodings.

We represent a given ABA$^+$framework $F = (\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-, \leq)$ in ASP as $\texttt{ABA}^+(F) = \texttt{ABA}(F) \cup \{\textbf{preferred}(x, y). \mid y \leq x\} \cup \pi^+_{preferences}$. Listing 2 shows $\pi^+_{preferences}$. Algorithm 2 employs the ASP modules $\pi_{cand} = \texttt{ABA}^+(F) \cup \pi^+_{cf} \cup \{constr(not\, \textbf{supported}(s))\}$ and $\pi_{check} = \texttt{ABA}^+(F) \cup \pi^+_{defended} \cup \pi^+_{suspect-defeat}$. The former, π_{cand}, encodes the abstraction (candidate search space) by considering conflict-free sets of assumptions that contain the queried sentence s in $\textbf{in}(I)$, for an answer set I of the encoding, and additionally computes all singleton assumptions not <-attacked by $\textbf{in}(I)$, in the ASP atoms $\textbf{undefeated}(I)$. In Line 5 of Algorithm 2 we check, based on Proposition 2, whether $\textbf{in}(I)$ corresponds to an <-admissible set in F: the ASP encoding is satisfiable iff there is a subset of $\textbf{undefeated}(I)$ that is not <-attacked by $\textbf{in}(I)$ but that <-attacks $\textbf{in}(I)$ (via either normal or reverse <-attacks). If $\textbf{in}(I)$ does correspond to an <-admissible set, this is a witness to s being credulously accepted. Otherwise, we exclude this assumption set via the constraint $constr(\textbf{out}(I) \cup \textbf{in}(I))$.

Listing 2. *Module* $\pi^+_{preferences}$

$\textbf{preferred}(X,Z) \leftarrow \textbf{preferred}(X,Y), \textbf{preferred}(Y,Z).$
$\textbf{less\_preferred}(X,Y) \leftarrow \textbf{preferred}(Y,X), \textit{not}\, \textbf{preferred}(X,Y).$
$\textbf{no\_less\_preferred}(X,Y) \leftarrow \textbf{assumption}(X), \textbf{assumption}(Y), \textit{not}\, \textbf{less\_preferred}(X,Y).$

Algorithm 2 is extended to cover enumeration of <-admissible assumption sets by reporting all found <-admissible sets and not terminating until there are no more candidates.

Listing 3. *Module* π^+_{cf}

$\textbf{in}(X) \leftarrow \textbf{assumption}(X), \textit{not}\, \textbf{out}(X).$
$\textbf{out}(X) \leftarrow \textbf{assumption}(X), \textit{not}\, \textbf{in}(X).$
$\textbf{supported}(X) \leftarrow \textbf{assumption}(X), \textbf{in}(X).$
$\textbf{supported}(X) \leftarrow \textbf{head}(R,X), \textbf{triggered\_by\_in}(R).$
$\textbf{triggered\_by\_in}(R) \leftarrow \textbf{head}(R,\_), \textbf{supported}(X) : \textbf{body}(R,X).$

← **in**(X), **contrary**(X,Y), **supported**(Y).
pref_supported(X,Y) ← **no_less_preferred**(X,Y), **assumption**(X), **in**(X).
pref_supported(X,Y) ← **head**(R,X), **pref_triggered_by_in**(R,Y).
pref_triggered_by_in(R,Y) ← **head**(R,_), **assumption**(Y), **pref_supported**(X,Y):
  body(R,X).
normally_defeated(Y) ← **pref_supported**(X,Y), **contrary**(Y,X).
derivable_from_undefeated(Z,Z) ← **assumption**(Z), *not* **normally_defeated**(Z).
derivable_from_undefeated(Y,Z) ← **head**(R,Y), **triggered_by_undefeated**(R,Z).
triggered_by_undefeated(R,Z) ← **head**(R,_), **assumption**(Z),
  derivable_from_undefeated(Y,Z) : **body**(R,Y).
in_attacked_by_normally_undefeated(X,Z) ← **in**(X),**contrary**(X,Y),
  derivable_from_undefeated(Y,Z).
reversely_defeated(Z) ← **less_preferred**(Z,X),
  in_attacked_by_normally_undefeated(X,Z).
undefeated(X) ← **assumption**(X), *not* **normally_defeated**(X), *not* **reversely_defeated**(X).

Listing 4. *Partial module* $\pi^{+}_{defended}$

suspect(X) ← **undefeated**(X), *not* **other**(X).
other(X) ← **undefeated**(X), *not* **suspect**(X).
pref_supported_by_suspects(X,Y) ← **in**(Y), **no_less_preferred**(X,Y), **assumption**(X),
  suspect(X).
pref_supported_by_suspects(X,Y) ← **in**(Y), **head**(R,X),
  pref_triggered_by_suspects (R,Y).
pref_triggered_by_suspects(R,Y) ← **in**(Y), **head**(R,_),
  pref_supported_by_suspects (X,Y) : **body**(R,X).
in_normally_defeated_by_suspects ← **in**(Y), **pref_supported_by_suspects**(X,Y),
  contrary(Y,X).
supported_by_in(X) ← **assumption**(X), **in**(X).
supported_by_in(X) ← **head**(R,X), **triggered_by_in**(R).
triggered_by_in(R) ← **head**(R,_), **supported_by_in**(X) : **body**(R,X).
reach_in(X,Y) ← **triggered_by_in**(R), **head**(R,Y), **body**(R,X).
reach_in(X,Y) ← **reach_in**(X,Z), **reach_in**(Z,Y).
reach_in(X,X) ← **in**(X).
in_reversely_defeated_by_suspects ← **suspect**(Y), **contrary**(Y,X), **supported_by_in**(X),
  in(Z), **reach_in**(Z,X), **less_preferred**(Z,Y).
← *not* **in_normally_defeated_by_suspects**, *not* **in_reversely_defeated_by_suspects**.

We present π^{+}_{cf} in Listing 3, $\pi^{+}_{defended}$ in Listing 4 and $\pi^{+}_{suspect-defeat}$ in Listing 5. The first six lines of Listing 3 encode conflict-freeness (note that conflict-freeness is independent of preferences (Čyras and Toni 2016b)). In brief, **in**(I) encodes a guess of an assumption set and **supported**(I) which sentences are derivable from this set. For computing assumptions x that are individually $<$-attacked by the assumptions A encoded by **in**(I), we make use of a result proven by (Lehtonen *et al.* 2021a, Lemma 8). Checking whether A normally $<$-attacks an x can directly be encoded by forward-derivations: if from the subset $A' \subseteq A$ which are not less preferred to x one can derive the contrary of x, a normal $<$-attack from A to x exists. For the remaining assumptions, it holds that A reversely $<$-attacks x if from $\{x\}$ one can derive the contrary of an assumption $a \in A$ with $x < a$.

From $\mathbf{in}(I)$ and $\mathbf{undefeated}(I)$ obtained as facts from an earlier ASP call, the encodings in Listings 4 and 5 determine whether $\mathbf{in}(I)$ defends itself against $\mathbf{undefeated}(I)$. In Listing 4, we guess a subset of the undefeated assumptions (called suspects here) and check whether this set $<$-attacks $\mathbf{in}(I)$ but is not $<$-attacked by $\mathbf{in}(I)$ (making $\mathbf{in}(I)$ undefended and not $<$-admissible). Line 1 encodes the guess and normal $<$-attacks as before. Reverse $<$-attacks from a set larger than one are more involved; the idea is taken from Lehtonen *et al.* (2021a, proof of Proposition 11.2). We compute what is supported by $\mathbf{in}(I)$. The set $\mathbf{in}(I)$ is reversely $<$-attacked by the $\mathbf{suspect}(I)$ set if one can tree-derive from $\mathbf{in}(I)$ a contrary of an assumption x in the $\mathbf{suspect}(I)$ set, with the required assumptions among $\mathbf{in}(I)$ having an assumption less preferred than x. To show that there is such a derivation tree from a subset A of the assumptions corresponding to $\mathbf{in}(I)$ that derives a contrary y of x, with one assumption $a \in A$ being less preferred than x, we check whether one can reach y from a via the derivation rules (implying existence of such a tree). In Listing 5, normal and reverse $<$-attacks from $\mathbf{in}(I)$ to $\mathbf{suspect}(I)$ are determined analogously to $<$-attacks from $\mathbf{suspect}(I)$ to $\mathbf{in}(I)$. The final constraints of Listings 4 and 5 ensure that $\mathbf{suspect}(I)$ $<$-attacks $\mathbf{in}(I)$, but not vice versa. If $\mathtt{ABA}^+(F) \cup \pi^+_{defended} \cup \pi^+_{suspect-defeat}$ is unsatisfiable, the assumption set encoded in $\mathbf{in}(I)$ defends itself.

Listing 5. *Module* $\pi^+_{suspect-defeat}$

```
supported_by_in(X,Y) ← suspect(Y), no_less_preferred(X,Y), assumption(X), in(X).
supported_by_in(X,Y) ← suspect(Y), head(R,X), triggered_by_in(R,Y).
triggered_by_in(R,Y) ← suspect(Y), head(R,_), assumption(Y), supported_by_in(X,Y)
    : body(R,X).
suspect_normally_defeated_by_in ← supported_by_in(X,Y), contrary(Y,X).
supported_by_suspects(X) ← assumption(X), suspect(X).
supported_by_suspects(X) ← head(R,X), triggered_by_suspects(R).
triggered_by_suspects(R) ← head(R,_), supported_by_suspects(X) : body(R,X).
reach_suspect(X,Y) ← triggered_by_suspects(R), head(R,Y), body(R,X).
reach_suspect(X,Y) ← reach_suspect(X,Z), reach_suspect(Z,Y).
reach_suspect(X,X) ← suspect(X).
suspect_reversely_defeated_by_in ← in(Y), contrary(Y,X), supported_by_suspects(X),
    suspect(Z), reach_suspect(Z,X), less_preferred(Z,Y).
← suspect_normally_defeated_by_in.
← suspect_reversely_defeated_by_in.
```

The following proposition states the correctness of Algorithm 2 based on Proposition 2 and the previous discussion on the details of the algorithm and the employed encodings.

Algorithm 3 Credulous acceptance under <-complete semantics

Require: ABA$^+$ framework $F = (\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-, \leq)$, $s \in \mathcal{L}$

Ensure: return YES if s is credulously accepted under <-complete semantics in F, NO otherwise

```
1:  $\pi_{cand} := \texttt{ABA}^+(F) \cup \pi_{cf} \cup \pi^+_{undefeated} \cup \pi^+_{prune} \cup \{constr(not\,\textbf{supported}(s))\}$
2:  $\pi_{check1} := \texttt{ABA}^+(F) \cup \pi^+_{defended} \cup \pi^+_{suspect-defeat}$
3:  $\pi_{check2} := \texttt{ABA}^+(F) \cup \pi^+_{com} \cup \pi^+_{suspect-defeat}$
4:  while $\pi_{cand}$ is satisfiable do
5:      Let $I$ be the found answer set; $flag := true$
6:      if $\pi_{check1} \cup \textbf{undefeated}(I) \cup \textbf{in}(I)$ unsatisfiable then
7:        for each $u \in \mathcal{A}$ such that $\textbf{undefeated}(a) \in I$ do
8:          if $\pi_{check2} \cup \textbf{undefeated}(I) \cup \{\textbf{target}(u)\} \cup \textbf{in}(I)$ is
            unsatisfiable then $flag := false$; break
9:        if $flag = true$ then return YES
10:     $\pi_{cand} := \pi_{cand} \cup \{constr(\textbf{out}(I) \cup \textbf{in}(I))\}$
11: return NO
```

Proposition 4
Algorithm 2 decides credulous acceptance under <-admissible semantics, that is, for a given ABA framework $F = (\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-)$ and $s \in \mathcal{L}$, Algorithm 2 returns YES if s is credulously accepted in F, and NO otherwise.

3.4 <-Complete semantics

For deciding credulous acceptance under <-complete semantics, we present Algorithm 3. There are two key differences to Algorithm 2: the abstraction π_{cand} is stronger and verifying whether a candidate assumption set is <-complete is more involved. For the former, in addition to the constraints posed in Algorithm 2, we add that if $\textbf{in}(I)$ corresponds to a conflict-free set of assumptions A and U is the set of all assumptions which are not individually <-attacked by A, then for each $a \in U$ it must hold that either $a \in A$ or a is <-attacked by U. By Proposition 2, this only rules out assumption sets that are not <-complete. As we will see in the experiments, pruning the search space in this manner can significantly speed up computation. For verifying whether a conflict-free set of assumptions is <-complete, Algorithm 3 checks in Lines 7–9 for each $a \in U$ whether a is defended by A, in addition to verifying <-admissibility (Line 6).

Enumeration of <-complete assumption sets can be achieved by reporting all found answers and not terminating until there are no candidates, and finding a <-complete assumption set by omitting the query and reporting $\textbf{in}(I)$ on Line 9.Enumeration also finds the <-grounded assumption set, which is defined as the intersection of all <-complete sets.

In module π^+_{prune} (Listing 6) we compute <-attacks on singleton assumptions. We consider singleton assumptions in U, and whether they are <-attacked by U. The final constraint rules out exactly the condition mentioned after Proposition 2, namely that there is an $a \in \mathcal{A}$ such that $a \notin \textbf{in}(I)$ and a is not <-attacked by U.

Via the encoding in Listing 7 together with Listing 5 we check whether, given a set $\textbf{in}(I)$ and an assumption $\textbf{target}(I)$, $\textbf{in}(I)$ defends $\textbf{target}(I)$. More specifically the module is

Listing 6. *Module* π^+_{prune}

pref_supported_by_undefeated(X,Y) ← **no_less_preferred**(X,Y), **assumption**(X),
    undefeated(X).
pref_supported_by_undefeated(X,Y) ← **head**(R,X), **pref_triggered_by_undefeated**(R,Y).
pref_triggered_by_undefeated(R,Y) ← **pref_supported_by_undefeated**(X,Y) :
    body(R,X), **assumption**(Y), **head**(R,_).
undefeated_normally_defeated_by_undefeated(Y) ← **undefeated**(Y),
    pref_supported_by_undefeated(X,Y), **contrary**(Y,X).
undefeated_reversely_defeated_by_undefeated(Z) ← **less_preferred**(Z,X),
    undefeated(X), **contrary**(X,Y), **derivable_from_undefeated**(Y,Z).
← **out**(Y), **undefeated**(Y), *not* **undefeated_normally_defeated_by_undefeated**(Y), *not*
    undefeated_reversely_defeated_by_undefeated(Y).

Listing 7. *Module* π^+_{com}

suspect(X) ← **assumption**(X), *not* **other**(X).
other(X) ← **assumption**(X), *not* **suspect**(X).
pref_supported_by_suspects(X) ← **target**(Y), **no_less_preferred**(X,Y), **assumption**(X),
    suspect(X).
pref_supported_by_suspects(X) ← **head**(R,X), **pref_triggered_by_suspects**(R).
pref_triggered_by_suspects(R) ← **head**(R,_), **pref_supported_by_suspects**(X) :
    body(R,X).
target_normally_attacked ← **target**(Y), **pref_supported_by_suspects**(X),
    contrary(Y,X).
derivable_from_target(X) ← **target**(X).
derivable_from_target(X) ← **head**(R,X), **triggered_by_target**(R).
triggered_by_target(R) ← **head**(R,_), **derivable_from_target**(X) : **body**(R,X).
suspect_attacked_by_target(X) ← **suspect**(X), **contrary**(X,Y),
    derivable_from_target(Y).
target_reversely_attacked ← **target**(Y),**less_preferred**(Y,X),
    suspect_attacked_by_target(X).
← *not* **target_normally_attacked**, *not* **target_reversely_attacked**.

unsatisfiable if there is no set of assumptions, called suspects here, that attack **target**(I) without **in**(I) attacking the suspect set. In other words, the encoding is unsatisfiable if the **target**(I) is defended by **in**(I). We check if the target is normally or reversely $<$-attacked via **suspect**(I). As before, in Listing 5 we compute normal and reverse $<$-attacks from **in**(I) to the **suspect**(I) set.

The following proposition states the correctness of the approach.

Proposition 5
Algorithm 3 decides credulous acceptance under $<$-complete semantics, that is, for a given ABA framework $F = (\mathcal{L}, \mathcal{R}, \mathcal{A}, {}^{-})$ and $s \in \mathcal{L}$, Algorithm 3 returns YES if s is credulously accepted under $<$-complete semantics in F, and NO otherwise.

4 Empirical evaluation

We implemented the ASP-based CEGAR algorithms using the incremental Python interface of Clingo v5.4.0 (Gebser *et al.* 2016; Kaminski *et al.* 2020). The implementation is available at https://bitbucket.org/coreo-group/aspforaba. We empirically evaluate its performance, comparing it to current state-of-the-art approaches: the Asprin-based (Brewka *et al.* 2015) approach to skeptical acceptance under preferred semantics (Lehtonen *et al.* 2021a) and the ABAplus system which supports computing assumption sets in ABA$^+$ under $<$-admissible and $<$-complete semantics (Bao *et al.* 2017). A direct comparison with ABAplus is only applicable for frameworks which satisfy the so-called WCP property (weak contraposition) due to restrictions in ABAplus.

For comparison with Asprin we use similar benchmarks as (Lehtonen *et al.* 2021a), with $|\mathcal{L}| = 250, 500, 1000, 1500, \ldots, 8000$. For each $|\mathcal{L}|$, we generated 20 frameworks with 15% and 30% of the sentences assumptions each for a total of 680 frameworks. The number of rules per head and body lengths, respectively, were randomly chosen from $[1, 20]$. For comparison with ABAplus, we use the 120 frameworks first used by Lehtonen *et al.* (2021a) that satisfy the WCP property, containing up to 30 sentences. The experiments were run single-threaded on 2.6-GHz Intel Xeon E5-2670 processors using per-instance 600-second time and 16-GB memory limit.

Figure 1 (left) shows a per-instance runtime comparison of the Asprin-based approach and our incremental ASP-based CEGAR algorithm for skeptical ABA reasoning under preferred semantics. The CEGAR approach clearly outperforms Asprin. A comparison of ABAplus and our CEGAR approach is shown in Table 1. The CEGAR approach dominates ABAplus in performance on the task of assumption set enumeration (as supported by ABAplus) under both $<$-admissible and $<$-complete semantics. We conclude that the CEGAR algorithms based on incremental ASP outperform the current state of the art on all of the three reasoning tasks.

Runtimes for Asprin are in its enumeration mode rather than query mode as Asprin is consistently faster on this task using enumeration, as shown in Figure 2 (left). Conversely, Figure 2 (right) shows that using our CEGAR approach, most instances are solved faster via direct skeptical reasoning compared to assumption set enumeration; there is only a handful of instances on which enumeration is faster.

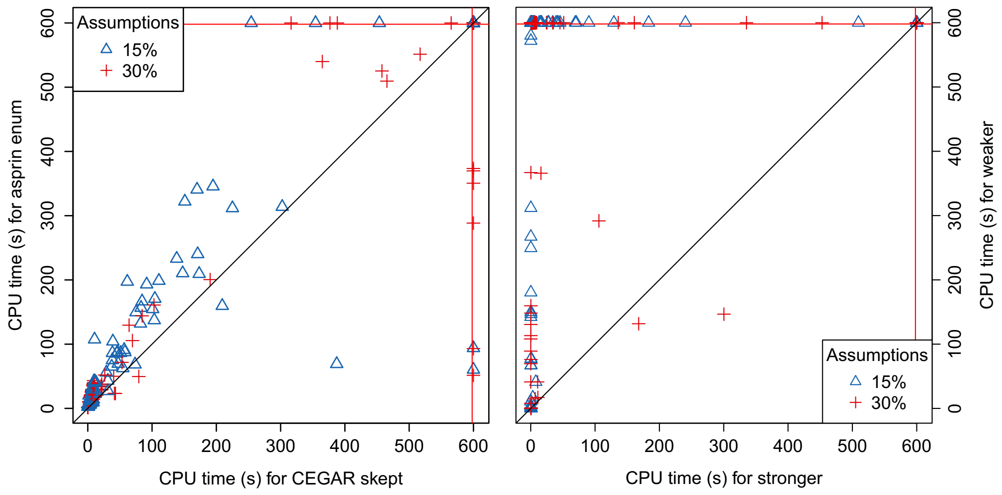

Fig. 1. Left: Runtime comparison of Asprin (enumeration) and incremental ASP on skeptical preferred. Right: Runtime comparison of the incremental ASP approach using the weaker and stronger abstraction for finding an assumption set under $<$-complete semantics.

Table 1. *Runtime comparison. Cumulative runtimes are over solved instances*

Problem	**Approach**	**#timeouts**	**Running times** (s) **mean**	**median**	**cumulative**
ABA$^+$ <-*adm* *enumeration*	CEGAR	**0**	**0.114**	**0.040**	**14**
	ABAplus	9	15.442	0.560	1714
ABA$^+$ <-*com* *enumeration*	CEGAR	**0**	**0.096**	**0.040**	**12**
	ABAplus	9	14.240	0.550	1581

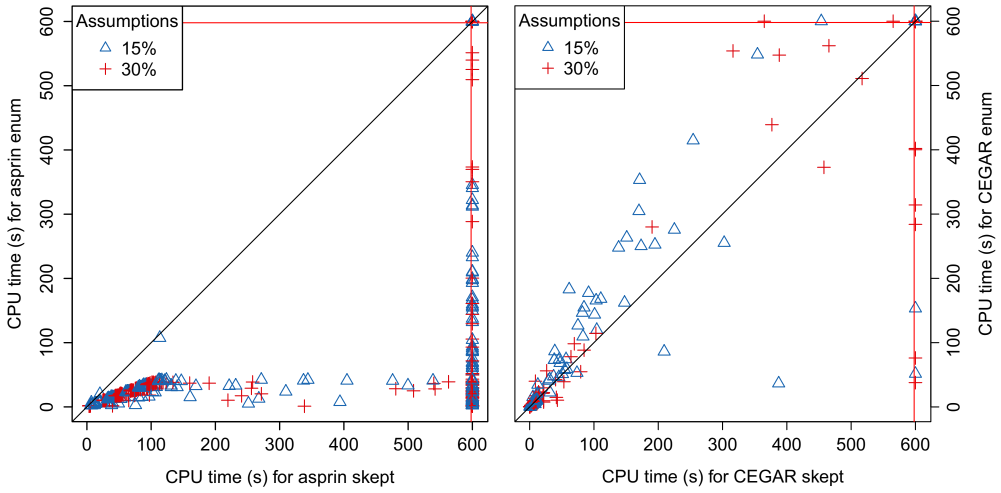

Fig. 2. Runtime comparisons under preferred semantics. Left: Asprin enumeration vs skeptical reasoning. Right: CEGAR enumeration vs skeptical reasoning.

For more insights into our CEGAR approach to ABA$^+$, we generated larger instances with 50-500 sentences. For each $|\mathcal{L}|$, we generated 30 instances with 15% and 30% assumptions each taking as preferences random permutations of the assumptions, with assumption a_i set to be preferred to a_j for $i < j$ in the permutation with probabilities 5%, 15% or 40% (10 instances for each probability). On these instances the stronger abstraction for <-complete semantics yields significant runtime improvements over the weaker abstraction, enabling scaling up to 500 sentences (Figure 1 right). The runtime improvements are at least in part due to the fact that the stronger abstraction results in considerably fewer iterations (essentially number of candidates found). Using the weaker abstraction the algorithm takes on average 1148 iterations, compared to 12 when using the stronger abstraction. Figure 3 (left) shows the iterations taken to solve each instance. Further, the runtimes of the CEGAR approach using the stronger abstraction under <-complete semantics are similar between the task of finding an assumption set without a query and credulous reasoning on both unsatisfiable and satisfiable instances; Figure 3 (right) shows the overall runtime results for credulous reasoning under <-complete semantics. We also observe that with a larger number of sentences being assumptions (30% vs 15%) instances tend to become harder to solve for all the considered problems.

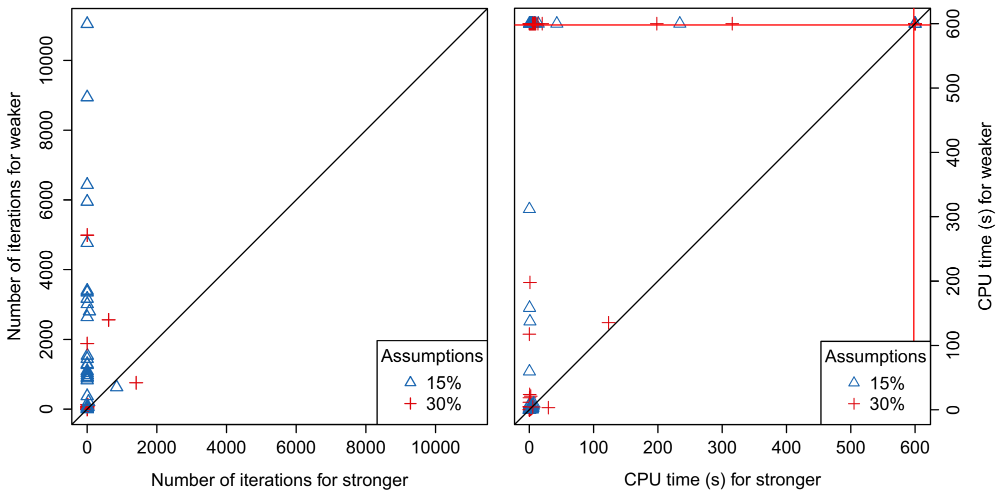

Fig. 3. Comparisons for <-complete semantics using the CEGAR approach. Left: Comparison of iterations needed using the weaker and stronger abstraction for finding a <-complete assumption set. Right: Runtime comparison between the weaker and stronger abstraction for answering credulous acceptance.

5 Conclusions

We developed an approach to beyond-NP reasoning in ABA frameworks based on recent advances in incremental answer set solving. In particular, we detailed ASP-based CEGAR procedures for skeptical acceptance under preferred semantics in ABA and credulous reasoning under <-admissible and <-complete semantics in ABA^+, and assumption set enumeration for all of these. Our implementation of the approach empirically outperforms previous algorithmic solutions to these reasoning tasks. We developed a stricter abstraction for <-complete semantics, speeding up solving in practice, and obtained complexity upper bounds for credulous reasoning in ABA^+ under <-complete semantics. A promising direction for further work is to extend the CEGAR approach considered in this work to other beyond-NP reasoning problems in ABA, such as reasoning over general (i.e. possibly non-flat) ABA frameworks.

Competing interests: The authors declare none.

Supplementary material

To view supplementary material for this article, please visit http://dx.doi.org/10.1017/S1471068421000296.

References

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

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

Besnard, P. and Hunter, A. 2018. A review of argumentation based on deductive arguments. In *Handbook of Formal Argumentation*. College Publications, Chapter 9, 437–484.

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

Brewka, G., Delgrande, J. P., Romero, J. and Schaub, T. 2015. asprin: Customizing answer set preferences without a headache. In *Proc. AAAI*. AAAI Press, 1467–1474.

Caminada, M. and Schulz, C. 2017. On the equivalence between assumption-based argumentation and logic programming. *Journal of Artificial Intelligence Research 60*, 779–825.

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, Chapter 15, 688–767.

Clarke, E. M., Grumberg, O., Jha, S., Lu, Y. and Veith, H. 2003. Counterexample-guided abstraction refinement for symbolic model checking. *Journal of the ACM 50,* 5, 752–794.

Clarke, E. M., Gupta, A. and Strichman, O. 2004. SAT-based counterexample-guided abstraction refinement. *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems 23,* 7, 1113–1123.

Craven, R. and Toni, F. 2016. Argument graphs and assumption-based argumentation. *Artificial Intelligence 233*, 1–59.

Craven, R., Toni, F. and Williams, M. 2013. Graph-based dispute derivations in assumption-based argumentation. In *TAFA 2013 Revised Selected Papers*. LNCS, vol. 8306. Springer, 46–62.

Čyras, K. 2017. ABA+: Assumption-based argumentation with preferences. Ph.D. thesis, Imperial College London, UK.

Čyras, K., Fan, X., Schulz, C. and Toni, F. 2018. Assumption-based argumentation: Disputes, explanations, preferences. In *Handbook of Formal Argumentation*. College Publications, Chapter 7, 365–408.

Čyras, K. and Oliveira, T. 2019. Resolving conflicts in clinical guidelines using argumentation. In *Proc. AAMAS*. IFAAMAS, 1731–1739.

Čyras, K. and Toni, F. 2016a. ABA+: Assumption-based argumentation with preferences. In *Proc. KR*. AAAI Press, 553–556.

Čyras, K. and Toni, F. 2016b. Properties of ABA+ for non-monotonic reasoning. In *Proc. NMR*. 25–34.

Dimopoulos, Y., Nebel, B. and Toni, F. 2002. On the computational complexity of assumption-based argumentation for default reasoning. *Artificial Intelligence 141,* 1/2, 57–78.

Dung, P. M., Kowalski, R. A. and Toni, F. 2006. Dialectic proof procedures for assumption-based, admissible argumentation. *Artificial Intelligence 170,* 2, 114–159.

Dung, P. M., Toni, F. and Mancarella, P. 2010. Some design guidelines for practical argumentation systems. In *Proc. COMMA*. FAIA, vol. 216. IOS Press, 183–194.

Fan, X. and Toni, F. 2016. On the interplay between games, argumentation and dialogues. In *Proc. AAMAS*. ACM, 260–268.

Fan, X., Toni, F., Mocanu, A. and Williams, M. 2014. Dialogical two-agent decision making with assumption-based argumentation. In *Proc. AAMAS*. IFAAMAS/ACM, 533–540.

Gaertner, D. and Toni, F. 2007. CaSAPI: A system for credulous and sceptical argumentation. In *Proc. NMR*. 80–95.

García, A. J. and Simari, G. R. 2018. Argumentation based on logic programming. In *Handbook of Formal Argumentation*. College Publications, Chapter 8, 409–435.

Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T. and Wanko, P. 2016. Theory solving made easy with Clingo 5. In *Technical Communications of ICLP*. OASICS. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2:1–2:15.

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

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

Kaminski, R., Romero, J., Schaub, T. and Wanko, P. 2020. How to build your own ASP-based system?! *CoRR abs/2008.06692*.

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

Lehtonen, T., Wallner, J. P. and Järvisalo, M. 2021a. Declarative algorithms and complexity results for assumption-based argumentation. *Journal of Artificial Intelligence Research 71*, 265–318.

Lehtonen, T., Wallner, J. P. and Järvisalo, M. 2021b. Harnessing incremental answer set solving for reasoning in assumption-based argumentation. *CoRR abs/2108.04192*.

Modgil, S. and Prakken, H. 2018. Abstract rule-based argumentation. In *Handbook of Formal Argumentation*. College Publications, Chapter 6, 287–364.

Niemelä, I. 1999. Logic programs with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence 25,* 3–4, 241–273.

Toni, F. 2013. A generalised framework for dispute derivations in assumption-based argumentation. *Artificial Intelligence 195*, 1–43.

Toni, F. 2014. A tutorial on assumption-based argumentation. *Argument & Computation 5,* 1, 89–117.