

MUS-Based Generation of Arguments and Counter-arguments

Philippe Besnard

IRIT-CNRS, Université de Toulouse

118 route de Narbonne, F-31065 Toulouse cedex, France

besnard@irit.fr

Éric Grégoire Cédric Piette Badran Raddaoui

Université Lille-Nord de France, Artois

CRIL-CNRS UMR 8188

F-62307 Lens Cedex, France

{gregoire, piette, raddaoui}@cril.fr

Abstract

Most of the approaches of computational argumentation define an argument as a pair consisting of premises and a conclusion, where the latter is entailed by the former. However, the matter of computing arguments and counter-arguments remains largely unsettled.

We propose here a method to compute arguments and counter-arguments in the context of propositional logic, by using the concept of a MUS (Minimally Unsatisfiable Subset). The idea relies on the fact that reduction ad absurdum is valid in propositional logic: $\langle \Phi, \alpha \rangle$ is an argument induced from a knowledge base Δ iff $\Phi \cup \{\neg\alpha\}$ is inconsistent. Therefore, if $\Phi \cup \{\neg\alpha\}$ is a MUS of $\Delta \cup \{\neg\alpha\}$ that contains $\neg\alpha$ then $\langle \Phi, \alpha \rangle$ is an argument from Δ . Not only do we present an algorithm that generates arguments, we also present an algorithm generating the complete argumentation tree induced by a given argument. We include a report on computational experimentations with both algorithms.

Keywords: Argumentation, MUS.

1 Introduction

Argumentation is an appealing way to reason from defective (whether incomplete, contradictory, etc.) premises. It is quite popular in a number of domains, be it law [8] [9], medicine [2], negotiation [7], decision making [4] and so on.

Various approaches have been proposed as computational models for argumentation. Most of them deal with e.g. establishing arguments for or against

a conclusion, identifying counter-arguments, defining an attack relation between arguments, etc. As to the latter, a fair amount of work has been devoted to methods for selecting preferred arguments à la Dung [10]. However, generating arguments in the first place is quite a different matter, for instance in an argumentative model where arguments are pairs (*premises, conclusion*) such that *conclusion* is a logical formula classically entailed by *premises*, a non-ordered collection of logical formulae. There has been recently some work on how to generate arguments in such an approach. Most notably, Efstathiou and Hunter [3] introduce a technique to compute such arguments whose conclusion consists of a single literal. Connection graphs are used where vertices are clauses and edges connect two vertices of which one has some disjunct that is the negation of a disjunct in the other (the usual way to word it is that two such vertices have complementary literals).

In this paper, another technique is presented in order to compute such arguments. It is mostly based [5] on the usual notion of minimal unsatisfiable set of formulae (the set is unsatisfiable but all its proper subsets are satisfiable). Given a knowledge base Δ , $\langle \Phi, \alpha \rangle$ is an argument for α iff the set $\Phi \cup \{\neg\alpha\}$ is a minimal unsatisfiable set.

The content of the paper is as follows. The next section is devoted to the argumentative model due to Besnard and Hunter [1]. Section 3 is about the basics of the notion of a MUS, that are to be used in Section 4 in order to generate arguments and counter-arguments as well as argumentation trees. These can be computed by means of the two algorithms that we provide in the last section. The conclusion of the paper includes some prospects for further work.

2 Argumentation

Logical formulae are denoted $\alpha, \beta, \gamma, \dots$ and sets thereof are denoted $\Phi, \Psi, \Theta, \dots$. The symbols \wedge, \vee, \neg and \perp denote conjunction, disjunction, negation and contradiction. Lastly, \vdash denotes entailment in propositional logic.

We consider a knowledge base Δ consisting of a finite set of clauses. Please note that Δ is fixed in the sequel. Moreover, we assume a distinguished enumeration for every subset of Δ , its *canonical enumeration*. Importantly, it only serves to provide the order in which the clauses of any subset of Δ are to be conjoined to yield a formula logically equivalent to this subset. No other constraint is imposed on Δ , particularly Δ needs not to be consistent.

We focus on the argumentative model developed by Besnard and Hunter [1] that adopts an intuitive notion of an argument: essentially, an argument is a set of formulae entailing another formula viewed as the conclusion of the argument.

Definition 1. An argument A is a pair $\langle \Phi, \alpha \rangle$ s.t.:

1. $\Phi \subseteq \Delta$
2. $\Phi \not\vdash \perp$
3. $\Phi \vdash \alpha$
4. $\forall \Phi' \subset \Phi, \Phi' \not\vdash \alpha$

A is said to be an argument for α . The sets Φ and α denote the *support* and the *conclusion* of A , respectively. Please note that all formulae in Φ are clauses but α does not need to be a clause.

Example 1. Let $\Delta = \{p, \neg p \vee q, \neg q \vee r, \neg p \vee \neg r\}$. In view of Δ , some arguments are:

$$\begin{aligned} &\langle \{p, \neg p \vee q, \neg q \vee r\}, r \rangle. \\ &\quad \langle \{p\}, p \rangle. \\ &\quad \langle \{p, \neg p \vee q\}, p \wedge q \rangle. \\ &\quad \langle \{\neg q \vee r\}, \neg(q \wedge \neg r) \rangle. \end{aligned}$$

The next definition introduces a notion of subsumption among arguments, in the sense of an argument implicitly containing another.

Definition 2. An argument $\langle \Phi, \alpha \rangle$ is *more conservative* than an argument $\langle \Psi, \beta \rangle$ iff $\Phi \subset \Psi$ et $\beta \vdash \alpha$.

Example 2. The argument $\langle \{p\}, p \vee q \rangle$ is more conservative than the argument $\langle \{p, \neg p \vee q\}, q \rangle$.

It may happen that an argument rebuts part of the support of another argument. This underlies the notion of attack, a major component of an argumentative model. In their approach, Besnard and Hunter capture a relation of attack between arguments by means of the following definition.

Definition 3. An *undercut* of an argument $\langle \Phi, \alpha \rangle$ is an argument $\langle \Psi, \neg(\varphi_1 \wedge \dots \wedge \varphi_n) \rangle$ s.t. $\{\varphi_1, \dots, \varphi_n\} \subseteq \Phi$.

Example 3. Let $\Delta = \{p, \neg p \vee q, r, \neg r \vee \neg p\}$. Consider the argument $\langle \{p, \neg p \vee q\}, q \rangle$. Among its undercuts is the argument $\langle \{r, \neg r \vee \neg p\}, \neg(p \wedge (\neg p \vee q)) \rangle$. Another of its undercuts is the argument $\langle \{r, \neg r \vee \neg p\}, \neg p \rangle$ which turns out to be less conservative.

Out of these definitions arises the notion of a maximal conservative counter-argument in the form of the notion of a *maximal conservative undercut*.

Definition 4. $\langle \Psi, \beta \rangle$ is a *maximal conservative undercut* of an argument $\langle \Phi, \alpha \rangle$ iff $\langle \Psi, \beta \rangle$ is an undercut of $\langle \Phi, \alpha \rangle$ such that no other undercut for $\langle \Phi, \alpha \rangle$ is strictly more conservative than $\langle \Psi, \beta \rangle$.

Stated otherwise, being a maximal conservative undercut $\langle \Psi, \beta \rangle$ of $\langle \Phi, \alpha \rangle$ means that for each undercut $\langle \Psi', \beta' \rangle$ of $\langle \Phi, \alpha \rangle$, if $\Psi' \subseteq \Psi$ et $\beta \vdash \beta'$ then $\Psi \subseteq \Psi'$ et $\beta' \vdash \beta$.

Example 4. Let us return to Example 3. The argument $\langle \{r, \neg r \vee \neg p\}, \neg(p \wedge (\neg p \vee q)) \rangle$ is a maximal conservative undercut of $\langle \{p, \neg p \vee q\}, q \rangle$.

Another notion of a counter-argument then arises which is the notion of a *canonical undercut*. These arguments are identified by Besnard and Hunter as of relevant focus, e.g. with respect to collating arguments and counter-arguments.

Definition 5. $\langle \Psi, \neg(\varphi_1 \wedge \dots \wedge \varphi_n) \rangle$ is a canonical undercut of $\langle \Phi, \alpha \rangle$ iff $\langle \Psi, \neg(\varphi_1, \dots, \varphi_n) \rangle$ is a maximal conservative undercut of $\langle \Phi, \alpha \rangle$ and $\langle \varphi_1, \dots, \varphi_n \rangle$ is the canonical enumeration of Φ .

Proposition 1. (Besnard & Hunter) The argument $\langle \Psi, \neg(\varphi_1 \wedge \dots \wedge \varphi_n) \rangle$ is a canonical undercut of $\langle \Phi, \alpha \rangle$ iff $\langle \Psi, \neg(\varphi_1, \dots, \varphi_n) \rangle$ is an undercut of $\langle \Phi, \alpha \rangle$ and $\langle \varphi_1, \dots, \varphi_n \rangle$ is the canonical enumeration of Φ .

In order to obtain a structure gathering arguments and counter-arguments for/against a specific conclusion, Besnard and Hunter define the so-called argumentation trees that collate such arguments and counter-arguments.

Definition 6. An *argumentation tree* for α is a tree whose nodes are such that:

1. The root is an argument for α
2. For every node $\langle \Psi, \beta \rangle$ whose ancestor nodes are $\langle \Psi_1, \beta_1 \rangle, \dots, \langle \Psi_n, \beta_n \rangle$, there exists $\gamma \in \Psi$ such that for $1 \leq i \leq n, \gamma \notin \Psi_i$

- Each child node is a canonical undercut of its parent node.

An argument tree aims at capturing the way counter-arguments can take place as the dispute develops. These trees have noticeable properties. Condition 2 insists that each counter-argument involves extra information thereby precluding cycles. As Δ is a finite set of formulae, it can be proved [1] that there are only finitely many argumentation trees for α and each of them is finite.

Definition 7. A complete argumentation tree for α is an argumentation tree for α such that the children nodes of a node A consist of all the canonical undercuts of A that satisfy 2.

Notation For the sake of readability, the conclusion of a canonical undercut is denoted \diamond (obviously, there is no ambiguity as to what formula it stands for).

Example 5. Let $\Delta = \{\neg p \vee \neg s, p \vee s, q, \neg q \vee p, r, \neg q, \neg r, s, \neg s\}$. Below is a complete argumentation tree for p .

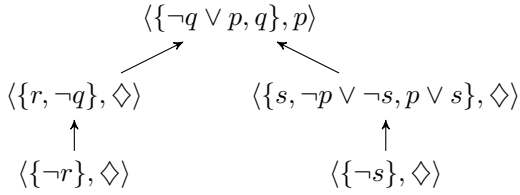


Figure 1. Argumentation tree for p

3 Minimal Unsatisfiable Set: MUS

A MUS (Minimal Unsatisfiable Set) is an unsatisfiable set of formulae which becomes satisfiable if deprived of any member. In other words, all proper subsets of a MUS are satisfiable although it is itself unsatisfiable.

Various methods exist towards computing MUSes of a set of clauses, e.g. Liffiton and Sakallah [6], or, more recently, Grégoire, Mazure and Piette [5].

Definition 8. Let Θ be a set of clauses. A Minimal Unsatisfiable Set for Θ is a set of clauses Γ such that:

- $\Gamma \subseteq \Theta$
- $\Gamma \vdash \perp$
- $\forall \Sigma \subset \Gamma, \Sigma \not\vdash \perp$

Notation $MUS(\Theta)$ denotes the set of all MUSes for Θ .

Example 6. Let $\Delta = \{p, \neg r, \neg q \vee \neg p, q, \neg q \vee r\}$. Δ has two MUSes, namely $\{p, q, \neg q \vee \neg p\}$ and $\{\neg r, q, \neg q \vee r\}$.

We will resort to the notion of a MUS in order to generate arguments for a formula. Furthermore, we will use the notion of a MUS as a way of computing all the canonical undercuts of an argument and therefore as a way to compute argumentation trees.

4 Generating (counter-)arguments

In general, argumentation involves a number of arguments and counter-arguments for/against some conclusion. Generating an argument for this conclusion requires extracting a minimal consistent subset entailing this conclusion.

Given Δ , producing an argument for α consists in finding a minimal subset of Δ that entails α . Accordingly, generating all arguments for α amounts to considering every subset Φ of Δ that satisfies both conditions $\Phi \not\vdash \perp$ and $\Phi \vdash \alpha$, discarding Φ if a proper subset of it also satisfies both conditions.

If $\langle \Phi, \alpha \rangle$ is an argument, then $\Phi \cup \{\neg \alpha\}$ then $\Phi \cup \{\neg \alpha\}$ is minimal inconsistent and $\Phi \cup \{\neg \alpha\}$ is a MUS of $\Delta \cup \{\neg \alpha\}$. As the converse is true as well, producing an argument for α amounts to find a MUS of $\Delta \cup \{\neg \alpha\}$ that contains $\neg \alpha$.

Proposition 2. (Besnard & Hunter) $\langle \Phi, \alpha \rangle$ is an argument iff $\Phi \cup \{\neg \alpha\}$ is a MUS of $\Delta \cup \{\neg \alpha\}$.

Example 7. Let Δ be as in Example 6.

- $\langle \{q\}, q \rangle$ is an argument as $\{q, \neg q\}$ is a MUS of $\Delta \cup \{\neg q\}$.
- $\langle \{q, \neg q \vee \neg p\}, \neg p \rangle$ is an argument as $\{q, \neg q \vee \neg p, \neg p\}$ is a MUS of $\Delta \cup \{\neg p\}$.
- $\langle \{q, \neg q \vee r\}, r \rangle$ is an argument as $\{q, \neg q \vee r, \neg r\}$ is a MUS of $\Delta \cup \{\neg r\}$.

Proposition 2 suggests that computing arguments is equivalent with computing MUSes so that algorithms that often prove efficient in finding MUSes can be taken advantage of. Note that these algorithms are dealing with checking satisfiability issues in the general propositional setting. Accordingly, unless $P=NP$, they can only be exponential in the worst case. Also, the number of MUSes can itself be exponential.

However, such algorithms do not work with arbitrary formulae but clauses (or, for a few algorithms, other syntactical restrictions). Unfortunately, Proposition 2 does not extend to clauses for example. Hence the need for Theorem 1 below, which means that generating all arguments for α is equivalent to computing every MUS of $\Delta \cup \{\neg \alpha\}$ that contains at

least one clause of $\neg\alpha$, provided a specific condition (to be detailed right after the statement of Theorem 1) is verified.

Notation Let $\bar{\alpha}$ denote a set of clauses logically equivalent with $\neg\alpha$.

Theorem 1. Let Φ be a subset of Δ . Let α be a formula. $\langle\Phi, \alpha\rangle$ is an argument iff $\Phi = \Gamma \setminus \bar{\alpha}$ for some MUS Γ of $\Delta \cup \bar{\alpha}$ such that $\Gamma \cap \bar{\alpha} \neq \emptyset$ but there is no MUS Γ' of $\Delta \cup \bar{\alpha}$ satisfying $\Gamma' \cup \bar{\alpha} \subset \Gamma \cup \bar{\alpha}$.

Proof. (\longrightarrow) Let $\langle\Phi, \alpha\rangle$ be an argument. By the definition, $\Phi \vdash \alpha$. Then, $\Phi \cup \bar{\alpha}$ is inconsistent. Assume now that there exist $\Phi' \subset \Phi$ and $\Theta \subseteq \bar{\alpha}$ such that $\Phi' \cup \Theta$ would be minimal inconsistent. Then, $\Phi' \cup \bar{\alpha}$ would be inconsistent and it would happen that $\Phi' \vdash \alpha$, contradicting $\langle\Phi, \alpha\rangle$ being an argument. For $\Phi' \cup \Theta$ to be minimal inconsistent, it thus must be the case that $\Phi' = \Phi$. Yet, $\Phi \cup \bar{\alpha}$ being inconsistent, it must have a minimal inconsistent subset: some Φ' and Θ must exist. Due to Φ being consistent, $\Phi' = \Phi$ makes Θ to be non-empty. Referring to the existence of a MUS as in the statement of the theorem, take Γ to be $\Phi \cup \Theta$ (observe that $\Gamma \cap \bar{\alpha} \neq \emptyset$ because Θ is a non-empty subset of $\bar{\alpha}$). Lastly, let Γ' be a MUS of $\Delta \cup \bar{\alpha}$. Letting $\Phi' = \Gamma' \setminus \bar{\alpha}$, it is not the case that $\Phi' \subset \Phi$ (otherwise Φ would not be minimal with respect to inferring α). Therefore, $\Gamma' \cup \bar{\alpha} \not\subset \Gamma \cup \bar{\alpha}$. (\longleftarrow) Let Γ be a MUS of $\Delta \cup \bar{\alpha}$ such as in the statement of the theorem. As Γ is a MUS, it is inconsistent and so is $\Phi \cup \bar{\alpha}$ hence $\Phi \vdash \alpha$. Should $\Phi' \vdash \alpha$ for some $\Phi' \subset \Phi$, $\Phi' \cup \bar{\alpha}$ would be inconsistent. A MUS Γ' of $\Phi' \cup \bar{\alpha}$ would then exist. By the definition of a MUS, $\Gamma' \subseteq \Phi' \cup \bar{\alpha}$ and $\Gamma' \setminus \bar{\alpha} \subseteq \Phi'$ would immediately ensue. As a consequence, $\Gamma' \setminus \bar{\alpha} \subseteq \Phi' \subset \Phi = \Gamma \setminus \bar{\alpha}$ would hold. This is to say that there would exist some MUS Γ' of $\Delta \cup \bar{\alpha}$ satisfying $\Gamma' \cup \bar{\alpha} \subset \Gamma \cup \bar{\alpha}$, a contradiction. Summing it up, Φ is minimal with respect to inferring α . Lastly, $\Phi = \Gamma \setminus \bar{\alpha}$ yields both $\Phi \subseteq \Delta$ (because $\Gamma \subseteq \Delta \cup \bar{\alpha}$) and $\Phi \not\vdash \perp$ (because Γ is a MUS and $\Gamma \cap \bar{\alpha} \neq \emptyset$). ■

The next example illustrates why Proposition 2 does not work when $\neg\alpha$ is to be converted into clauses, hence the need for Theorem 1.

Example 8. Let $\Delta = \{p, \neg p \vee q\}$. In search of arguments for $p \vee q$, consider the MUSes of $\Delta \cup \{\neg p, \neg q\}$. They are $\Gamma_1 = \{p, \neg p\}$ and $\Gamma_2 = \{p, \neg p \vee q, \neg q\}$. Observe that $\Gamma_1 \cap \{\neg p, \neg q\} \neq \emptyset$ and $\Gamma_2 \cap \{\neg p, \neg q\} \neq \emptyset$ as well. However, $\langle\Gamma_1 \setminus \{\neg p, \neg q\}, p \vee q\rangle$ is an argument

$$\langle\Gamma_1 \setminus \{\neg p, \neg q\}, p \vee q\rangle = \langle\{p\}, p \vee q\rangle$$

but $\langle\Gamma_2 \setminus \{\neg p, \neg q\}, p \vee q\rangle$ is not an argument

$$\langle\Gamma_2 \setminus \{\neg p, \neg q\}, p \vee q\rangle = \langle\{p, \neg p \vee q\}, p \vee q\rangle$$

because $\Gamma_1 \setminus \{\neg p, \neg q\} \subset \Gamma_2 \setminus \{\neg p, \neg q\}$ which means that $\Gamma_2 \setminus \{\neg p, \neg q\}$ is not minimal w.r.t. inferring $p \vee q$. Also, please note that $\Gamma_1 \setminus \{\neg p, \neg q\} \subset \Gamma_2 \setminus \{\neg p, \neg q\}$ implies $\Gamma_1 \cup \{\neg p, \neg q\} \subset \Gamma_2 \cup \{\neg p, \neg q\}$ which makes Γ_2 fail the last condition in Theorem 1.

Due to argumentation trees, a case of utmost importance is that of computing canonical undercuts. Dealing with these turns out to be simpler because only MUSes of Δ need be considered as shown by Corollary 1.

Corollary 1. $\langle\Psi, \diamond\rangle$ is a canonical undercut for $\langle\Phi, \alpha\rangle$ iff $\Psi = \Gamma \setminus \Phi$ for some MUS Γ of Δ such that $\Gamma \cap \Phi \neq \emptyset$ and there exist no MUS Γ' of Δ satisfying $\Gamma' \cup \Phi \subset \Gamma \cup \Phi$.

Example 9. Let $\Delta = \{p, \neg p, q, p \vee \neg q\}$. The MUSes for Δ are $\Gamma_1 = \{p, \neg p\}$ and $\Gamma_2 = \{p \vee \neg q, \neg p, q\}$. Consider the argument $\langle\{p, q\}, p \wedge q\rangle$. What are its canonical undercuts? In the notation of Corollary 1, $\Phi = \{p, q\}$ and it happens that both $\Gamma_1 \cap \Phi \neq \emptyset$ and $\Gamma_2 \cap \Phi \neq \emptyset$. Now, $\langle\Gamma_1 \setminus \Phi, \diamond\rangle$ is a canonical undercut

$$\langle\Gamma_1 \setminus \Phi, \diamond\rangle = \langle\{\neg p\}, \neg(p \wedge q)\rangle$$

but $\langle\Gamma_2 \setminus \Phi, \diamond\rangle$ is not an argument

$$\langle\Gamma_2 \setminus \Phi, \diamond\rangle = \langle\{\neg p, p \vee \neg q\}, \neg(p \wedge q)\rangle$$

because $\Gamma_1 \setminus \Phi \subset \Gamma_2 \setminus \Phi$ hence $\Gamma_2 \setminus \Phi$ is not minimal with respect to inferring $\neg(p \wedge q)$. Please notice that $\Gamma_1 \setminus \Phi \subset \Gamma_2 \setminus \Phi$ entails $\Gamma_1 \cup \Phi \subset \Gamma_2 \cup \Phi$ which makes Γ_2 to fail the last condition in Corollary 1.

Theorem 1 yields a way out in computing arguments through MUSes. The condition $\Gamma' \cup \bar{\alpha} \subset \Gamma \cup \bar{\alpha}$ suggests that computing all the MUSes of $\Delta \cup \bar{\alpha}$ need only be extended by supplementing any such MUS with $\bar{\alpha}$. As such, it is an attractive solution but a weakness now shows up: computing a MUS in the first place involves some check for minimality and so does the last condition in Theorem 1, presumably causing redundant tests. This can fortunately be overcome by using a solver that computes MUSes out of MSSes (Maximal Satisfiable Subsets) by appropriately picking a clause in each MSS: for our purposes, all clauses from $\bar{\alpha}$ would be regarded as picked whenever any one of them is. The next section is devoted to such a solver, HYCAM [5].

5 Algorithms

We now present two algorithms to generate arguments and counter-arguments as just discussed.

The first algorithm, called BA, generates all arguments for a specific conclusion. Given a knowledge base Δ in clausal form and a formula α , this algorithm first converts $\neg\alpha$ into clausal form. The next step consists in extracting the MUSes from $\Delta \cup \{\neg\alpha\}$ that contain at least one clause of $\neg\alpha$. This is achieved by means of the HYCAM function, a variant of the HYCAM algorithm (HYbridization for Computing All Muses) [5].

Algorithm 1: BA

Input: Δ a set of clauses, α a formula

Output: a set of arguments for α

```

1  $\alpha' \leftarrow \text{CNF}(\neg\alpha);$  /*  $\alpha' = c_1 \wedge \dots \wedge c_n$  */
2  $\Theta \leftarrow \text{HYCAM}(\Delta \cup \{\alpha'\});$ 
3  $A(\alpha) \leftarrow \emptyset;$ 
4 forall  $\Gamma \in \Theta$  do
5   forall  $c_i \in \alpha'$  do
6     if  $((c_i \cap \Gamma) \neq \emptyset)$  then
7        $A(\alpha) \leftarrow A(\alpha) \cup \{\langle \Gamma \setminus \{c_i\}, \alpha \rangle\};$ 
8 return  $A(\alpha)$ 

```

Unless computation blown-up in worst-cases, HYCAM finds all MUSes from a given set of clauses, using their MSSes. For our purposes, it is enough to modify HYCAM through the *Camus-mus* procedure so that all MSSes of $\Delta \cup \{\neg\alpha\}$ containing a clause of $\neg\alpha$ are generated, and the desired MUSes are extracted. The second algorithm, called BT, generates the complete argumentation tree for a given argument which is the root of the tree. The other nodes are recorded in a stack as they are found, popping the last item as its canonical undercuts are obtained. For every MUS Γ of Δ , $\Gamma \cap \Phi \neq \emptyset$ is checked as well as $\Gamma \cup \Phi \not\subseteq \Gamma' \cup \Phi$ for every other MUS Γ' . Then, it is verified that $\Gamma \setminus \Phi$ is a subset of the set-theoretic union of the supports of the ancestor nodes of the item just popped.

The latter check is part of the definition of an argumentation tree, it requires the counter-argument being computed to have its support to include some new piece of information thus precluding any cycles. If the checks are ok, then the algorithm adds a node $\langle \Gamma \setminus \Phi, \neg(\phi_1 \wedge \dots \wedge \phi_n) \rangle$ as a child of the node $\langle \Phi, \alpha \rangle$.

6 Computational results

The BA algorithm has been implemented in C. First, it was run on structured instances selected from

Algorithm 2: BT

Input: Δ a set of clauses, α a formula, A an argument

Output: an argumentation tree for α

```

1  $\Theta \leftarrow \text{HYCAM}(\Delta);$ 
2  $T_0(\alpha) \leftarrow A;$ 
3  $p \leftarrow \text{push}(p, T_0(\alpha));$ 
4 while (not  $(\text{IsEmpty}(p))$ ) do
5    $\text{Arg} \leftarrow \text{Top}(p);$ 
6    $\text{Pop}(p);$ 
7    $\Phi \leftarrow \text{Support}(\text{Arg});$ 
8    $\varphi \leftarrow \text{SupAnc}(\text{Arg});$ 
9   forall  $\Gamma \in \Theta$  do
10    if  $((\Gamma \cap \Phi) \neq \emptyset)$  and  $(\Gamma \setminus \Phi \not\subseteq \varphi)$  then
11       $\text{AddNodes}(\text{Arg}, \langle \Gamma \setminus \Phi, \neg\Phi \rangle);$ 
12       $\text{push}(p, \langle \Gamma \setminus \Phi, \neg\Phi \rangle);$ 
13 return  $T(\alpha)$ 

```

the last SAT competitions¹ and instances standardly generated ones (random_1, random_2). Table 1 reports a sample of experiments over significant examples. Experimentations have been conducted on Intel Xeon 3GHz under Linux CentOS 4.1. (kernel 2.6.9) with a RAM limit of 2GB.

For each experimentation, Table 1 reports the instance name (*Instance*), its numbers of variables and clauses (*#var*, *#cl*), the conclusion aimed at (*Statement*), the number of generated arguments (*#Arguments*), which actually represents the number of MUSes involving the wanted conclusion and extracted by HYCAM [5], the average size of the arguments in term of the number of clauses from the CNF (*Average size*), and finally, computation time in seconds (*Time*).

First, not surprisingly, various numbers of arguments can be returned if different conclusions are wanted (see e.g. barrel2: third and ninth clause). In addition, for some instances, no argument can be extracted, which means that no MUS of the instance involves the wanted conclusion (ezfact32-1). Let us emphasize that the obtained results show that in numerous cases, the algorithm is able to extract arguments in a very short time. A sample of results over larger benchmarks (e.g. C202_FW_SZ_123 consisting of 8687 clauses and encoding automotive product configuration) are also reported.

Very often, the number of MUSes containing a clause from the negation of the conclusion is rather small hence the computing time is short. When the

¹<http://www.satcompetition.org>

Instance	(#var, #cl)	Statement	#Arguments	Average size	Time
barrel2	(50, 159)	\neg (9th clause)	18	91	0.06
barrel2	(50, 159)	\neg (3rd clause)	27	86	0.08
aim100-2.0-no-3	(100, 200)	\neg (1st clause)	1	27	0.06
aim100-2.0-no-4	(100, 200)	\neg (1st clause)	1	31	0.07
aim200-2.0-no-4	(200, 400)	\neg (1st clause)	2	42	0.06
ezfact32-1	(769, 4777)	\neg (1st clause)	0	-	0.81
c499	(606, 1870)	\neg (1st clause)	1	1638	4.96
C170_FR_SZ_92	(1659, 5083)	\neg (1st clause)	267	61	70.5
C202_FW_SZ_123	(1799, 8687)	\neg (1st clause)	4	34	18.3
random_1	(200, 800)	\neg (1st clause)	0	-	0.5
random_1	(200, 800)	\neg (8th clause)	5107	33	0.52
random_2	(350, 1100)	\neg (1st clause)	60253	31	24.7

Table 1. Experiments with BA over various instances.

number of MUSes containing some clause from the negation of the conclusion is large then so is the time needed by BA to generate all the arguments.

Lastly, Table 1 contains data for results for randomly generated benchmarks consisting of binary clauses and yielding a large number of MUSes. In the `random_1` case, 5107 arguments are generated in less than one second. In `random_2` case, HYCAM extracts 60253 MUSes that contains the conclusion. from the negation of the conclusion and it took 24.7 seconds in order to generate the arguments.

Those good results for both structured and random formulae suggest that in spite of a high worst case complexity, this approach could deal with large CNF that encode complex real-life situations. Obviously enough, we should keep in mind that such a computation can only be tractable when the number of MUSes is itself tractable.

7 Conclusion

Generating arguments and counter-arguments is a fundamental task in argumentation. We have introduced a technique to generate arguments for a given statement, and to generate canonical undercuts through computing MUSes of the knowledge base. Our approach is complete in the sense that all arguments relative to the statement at hand are generated and so are all relevant counter-arguments (the latter is evidenced by the fact that the argumentation trees generated are complete). An obvious topic for future work is to lift our approach to the quantified case.

References

[1] Ph. Besnard and A. Hunter. *Elements of Argumentation*. 2008.

[2] S. K. Das, J. Fox, and P. Krause. A unified framework for hypothetical and practical reasoning (1): Theoretical foundations. In *FAPR*, volume 1085 of *LNCS*, pages 58–72, 1996.

[3] V. Efstathiou and A. Hunter. Algorithms for effective argumentation in classical propositional logic: A connection graph approach. In *FoIKS*, volume 4932 of *LNCS*, pages 272–290, 2008.

[4] G. Ferguson, J. F. Allen, and B. W. Miller. Trains-95: Towards a mixed-initiative planning assistant. In *AIPS*, pages 70–77, 1996.

[5] É. Grégoire, B. Mazure, and C. Piette. Using local search to find MSSes and MUSes. *European J. of Op. Res.*, 199(3):640–646, 2009.

[6] M. H. Liffiton and K. A. Sakallah. On finding all minimally unsatisfiable subformulas. In *SAT*, volume 3569 of *LNCS*, pages 173–186, 2005.

[7] S. Parsons, C. Sierra, and N. R. Jennings. Agents that reason and negotiate by arguing. *J. Log. Comput.*, 8(3):261–292, 1998.

[8] H. Prakken. An argumentation framework in default logic. *Ann. Math. Artif. Intell.*, 9(1-2):93–132, 1993.

[9] H. Prakken and G. Sartor. A dialectical model of assessing conflicting arguments in legal reasoning. *Artif. Intell. Law*, 4(3-4):331–368, 1996.

[10] G. Vreeswijk. An algorithm to compute minimally grounded and admissible defence sets in argument systems. In *COMMA*, volume 144 of *Frontiers in Artificial Intelligence and Applications*, pages 109–120, 2006.