Approximate Arguments for Efficiency in Logical Argumentation

Anthony Hunter
Department of Computer Science
University College London
Gower Stree, London WC1E 6BT, UK

Abstract

There are a number of frameworks for modelling argumentation in logic. They incorporate a formal representation of individual arguments and techniques for comparing conflicting arguments. A common assumption for logic-based argumentation is that an argument is a pair $\langle \Phi, \alpha \rangle$ where Φ is minimal subset of the knowledgebase such that Φ is consistent and Φ entails the claim α. Different logics are based on different definitions for entailment and consistency, and give us different options for argumentation. For a variety of logics, in particular for classical logic, the computational viability of generating arguments is an issue. Here, we propose ameliorating this problem by using approximate arguments.

Introduction

Argumentation is a vital aspect of intelligent behaviour by humans. Consider diverse professionals such as politicians, journalists, clinicians, scientists, and administrators, who all need to collate and analyse information looking for pros and cons for consequences of importance when attempting to understand problems and make decisions.

There are a number of proposals for logic-based formalisations of argumentation (for reviews see (Prakken & Vreeswijk 2000; Chesnevar, Maguitman, & Loui 2001)). These proposals allow for the representation of arguments for and against some claim, and for attack relationships between arguments. In a number of key examples of argumentation systems, an argument is a pair where the first item in the pair is a minimal consistent set of formulae that proves the second item which is a formula. Furthermore, in these approaches, the notion of attack is a form of undercut, where one argument undercuts another argument when the claim of the first argument negates the premises of the second argument.

In this paper, we consider how we can undertake argumentation more efficiently. Let us start by considering the construction of individual arguments. If Δ is a knowledgebase, and we are interested in a claim α, we look for an argument $\langle \Phi, \alpha \rangle$ where $\Phi \subseteq \Delta$. Deciding whether a set of propositional classical formulae is classically consistent is an NP-complete decision problem and deciding whether a set of propositional formulae classically entails a given formula is a co-NP-complete decision problem. However, if we consider the problem as an abduction problem, where we seek the existence of a minimal subset of a set of formulae that implies the consequent, then the problem is in the second level of the polynomial hierarchy (Eiter & Gottlob 1995). Even worse deciding whether a set of first-order classical formulae is consistent is an undecidable decision problem. So even finding the basic units of argumentation is computationally challenging.

Proof procedures and algorithms have been developed for finding preferred arguments from a knowledgebase following for example Dung's preferred semantics (see for example (Prakken & Sartor 1997; Kakas & Toni 1999; Cayrol, Doutre, & Mengin 2001; Dimopoulos, Nebel, & Toni 2002; Dung, Kowalski, & Toni 2006)). However, these techniques and analyses do not offer any ways of ameliorating the computational complexity inherent in finding arguments and counterarguments even though it is a significant source of computational inefficiency.

A possible approach to ameliorate the cost of entailment is to use approximate entailment: Proposed in (Levesque 1984), and developed in (Schaerf & Cadoli 1995), classical entailment is approximated by two sequences of entailment relations. Approximate entailment has been developed for anytime coherence reasoning (Koriche 2002). However, the approach still needs to be further developed and evaluated for finding arguments and counterarguments in argumentation. This would need to start with a conceptualization of the notions of argument and counteragument derived using approximate entailment and approximate coherence.

In this paper, we take a different approach by presenting a new solution that uses approximate arguments. First we review an existing version of logic-based argumentation in order to illustrate our ideas, and then we present our framework for approximate arguments.

Logical argumentation

In this section we review an existing proposal for logic-based argumentation (Besnard & Hunter 2001). We consider a classical propositional language with classical deduction denoted by the symbol $\vdash$. We use $\alpha, \beta, \gamma, \ldots$ to denote formulae and $\Delta, \Phi, \Psi, \ldots$ to denote sets of formulae.

For the following definitions, we first assume a knowledgebase Δ (a finite set of formulae) and use this Δ throughout. We further assume that every subset of Δ is given an enumeration $\langle \alpha_1, \ldots, \alpha_n \rangle$ of its elements, which we call its canonical enumeration. This really is not a demanding constraint: In particular, the constraint is satisfied whenever we impose an arbitrary total ordering over Δ. Importantly, the order has no meaning and is not meant to represent any respective importance of formulae in Δ. It is only a convenient way to indicate the order in which we assume the formulae in any subset of Δ are conjoined to make a formula logically equivalent to that subset.

The paradigm for the approach is a large repository of information, represented by Δ, from which arguments can be constructed for and against arbitrary claims. Apart from information being understood as declarative statements, there is no a priori restriction on the contents, and the pieces of information in the repository can be as complex as possible. Therefore, Δ is not expected to be consistent. It need even not be the case that every single formula in Δ is consistent.

The framework adopts a very common intuitive notion of an argument. Essentially, an argument is a set of relevant formulae that can be used to classically prove some claim, together with that claim. Each claim is represented by a formula.

Definition 1. *An* **argument** *is a pair $\langle\Phi,\alpha\rangle$ such that: (1) $\Phi \subseteq \Delta$; (2) $\Phi \nvdash \bot$; (3) $\Phi \vdash \alpha$; and (4) there is no $\Phi' \subset \Phi$ such that $\Phi' \vdash \alpha$. We say that $\langle\Phi,\alpha\rangle$ is an argument for α. We call α the* **claim** *of the argument and Φ the* **support** *of the argument (we also say that Φ is a support for α).*

Example 1. *Let $\Delta = \{\alpha, \alpha \to \beta, \gamma \to \neg\beta, \gamma, \delta, \delta \to \beta, \neg\alpha, \neg\gamma\}$. Some arguments are:*

$$\langle\{\alpha, \alpha \to \beta\}, \beta\rangle$$
$$\langle\{\neg\alpha\}, \neg\alpha\rangle$$
$$\langle\{\alpha \to \beta\}, \neg\alpha \vee \beta\rangle$$
$$\langle\{\neg\gamma\}, \delta \to \neg\gamma\rangle$$

Arguments are not independent. In a sense, some encompass others (possibly up to some form of equivalence). To clarify this requires a few definitions as follows.

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

Example 2. *$\langle\{\alpha\}, \alpha \vee \beta\rangle$ is more conservative than $\langle\{\alpha, \alpha \to \beta\}, \beta\rangle$.*

Some arguments directly oppose the support of others, which amounts to the notion of an undercut.

Definition 3. *An* **undercut** *for an argument $\langle\Phi,\alpha\rangle$ is an argument $\langle\Psi, \neg(\phi_1 \wedge \ldots \wedge \phi_n)\rangle$ where $\{\phi_1, \ldots, \phi_n\} \subseteq \Phi$.*

Example 3. *Let $\Delta = \{\alpha, \alpha \to \beta, \gamma, \gamma \to \neg\alpha\}$. Then, $\langle\{\gamma, \gamma \to \neg\alpha\}, \neg(\alpha \wedge (\alpha \to \beta))\rangle$ is an undercut for $\langle\{\alpha, \alpha \to \beta\}, \beta\rangle$. A less conservative undercut for $\langle\{\alpha, \alpha \to \beta\}, \beta\rangle$ is $\langle\{\gamma, \gamma \to \neg\alpha\}, \neg\alpha\rangle$.*

Definition 4. *$\langle\Psi,\beta\rangle$ is a* **maximally conservative undercut** *of $\langle\Phi,\alpha\rangle$ iff $\langle\Psi,\beta\rangle$ is an undercut of $\langle\Phi,\alpha\rangle$ such that no undercuts of $\langle\Phi,\alpha\rangle$ are strictly more conservative than $\langle\Psi,\beta\rangle$ (that is, for all undercuts $\langle\Psi',\beta'\rangle$ of $\langle\Phi,\alpha\rangle$, if $\Psi' \subseteq \Psi$ and $\beta \vdash \beta'$ then $\Psi \subseteq \Psi'$ and $\beta' \vdash \beta$).*

The value of the following definition of canonical undercut is that we only need to take the canonical undercuts into account. This means we can justifiably ignore the potentially very large number of non-canonical undercuts.

Definition 5. *An argument $\langle\Psi, \neg(\phi_1 \wedge \ldots \wedge \phi_n)\rangle$ is a* **canonical undercut** *for $\langle\Phi,\alpha\rangle$ iff it is a maximally conservative undercut for $\langle\Phi,\alpha\rangle$ and $\langle\phi_1, \ldots, \phi_n\rangle$ is the canonical enumeration of Φ.*

An argument tree describes the various ways an argument can be challenged, as well as how the counter-arguments to the initial argument can themselves be challenged, and so on recursively.

Definition 6. *A* **complete argument tree** *for α is a tree where the nodes are arguments such that*

1. *The root is an argument for α.*
2. *For no node $\langle\Phi,\beta\rangle$ with ancestor nodes $\langle\Phi_1,\beta_1\rangle, \ldots, \langle\Phi_n,\beta_n\rangle$ is Φ a subset of $\Phi_1 \cup \cdots \cup \Phi_n$.*
3. *The children nodes of a node N consist of all canonical undercuts for N that obey 2.*

The second condition in Definition 6 ensures that each argument on a branch has to introduce at least one formula in its support that has not already been used by ancestor arguments. This is meant to avoid making explicit undercuts that simply repeat over and over the same reasoning pattern except for switching the role of some formulae (e.g. in mutual exclusion, stating that α together with $\neg\alpha \vee \neg\beta$ entails $\neg\beta$ is exactly the same reasoning as expressing that β together with $\neg\alpha \vee \neg\beta$ entail $\neg\alpha$, because in both cases, what is meant is that α and β exclude each other). As a notational convenience, in examples of argument trees the $\diamond$ symbol is used to denote the claim of an argument when that argument is a canonical undercut (no ambiguity arises as proven in (Besnard & Hunter 2001)).

Example 4. *Let $\Delta = \{\alpha \vee \beta, \alpha \to \gamma, \neg\gamma, \neg\beta, \delta \leftrightarrow \beta\}$. For this, two argument trees for the consequent $\alpha \vee \neg\delta$ are given.*

$$\begin{tikzcd}
\langle\{\alpha \vee \beta, \neg\beta\}, \alpha \vee \neg\delta\rangle & \langle\{\delta \leftrightarrow \beta, \neg\beta\}, \alpha \vee \neg\delta\rangle \\
\langle\{\alpha \to \gamma, \neg\gamma\}, \diamond\rangle \arrow[u] & \langle\{\alpha \vee \beta, \alpha \to \gamma, \neg\gamma\}, \diamond\rangle \arrow[u]
\end{tikzcd}$$

A complete argument tree is an efficient representation of the counterarguments, counter-counterarguments, ... Furthermore, if Δ is finite, there is a finite number of argument trees with the root being an argument with consequent α that can be formed from Δ, and each of these trees has finite branching and a finite depth (the finite tree property). Note, also the definitions presented in this section can be used directly with first-order classical logic, so Δ and α are from the first-order classical language. Interestingly, the finite tree property also holds for the first-order case (Besnard & Hunter 2005).

Motivation for approximation

We now turn to the potential for approximation when building argument trees. When building arguments, and hence argument trees, it is tempting to think that automated reasoning technology can do more for us than it is guaranteed to. For each argument, we need a minimal set of formulae that proves the claim. An automated theorem prover (an ATP) may use a "goal-directed" approach, bringing in extra premises when required, but they are not guaranteed to be minimal. For example, supposing we have a knowledgebase $\{\alpha, \alpha \wedge \beta\}$, for proving $\alpha \wedge \beta$, the ATP may start with the premise α, then to prove β, a second premise is required, which would be $\alpha \wedge \beta$, and so the net result is $\{\alpha, \alpha \wedge \beta\} \vdash \alpha \wedge \beta$, which does not involve a minimal set of premises. In addition, an ATP is not guaranteed to use a consistent set of premises since by classical logic it is valid to prove anything from an inconsistency.

So if we seek arguments for a particular claim δ, we need to post queries to an ATP to ensure that a set of premises entails δ, that the set of premises is minimal for this, and that it is consistent. So finding arguments for a claim α involves considering subsets Φ of Δ and testing them with the ATP to ascertain whether $\Phi \vdash \alpha$ and $\Phi \not\vdash \bot$ hold. For $\Phi \subseteq \Delta$, and a formula α, let $\Phi?\alpha$ denote a call (a query) to an ATP. If Φ classically entails α, then we get the answer $\Phi \vdash \alpha$, otherwise we get the answer $\Phi \not\vdash \alpha$, In this way, we do not give the whole of Δ to the ATP. Rather we call it with particular subsets of Δ. So for example, if we want to know if $\langle \Phi, \alpha \rangle$ is an argument, then we have a series of calls $\Phi?\alpha$, $\Phi?\bot$, $\Phi \setminus \{\phi_1\}?\alpha$,...,$\Phi \setminus \{\phi_k\}?\alpha$, where $\Phi = \{\phi_1, .., \phi_k\}$. So the first call is to ensure that $\Phi \vdash \alpha$ holds, the second call is to ensure that $\Phi \not\vdash \bot$ holds, the remaining calls are to ensure that there is no subset Φ' of Φ such that $\Phi' \vdash \alpha$ holds. Now if $\Phi \vdash \alpha$ holds, but some of the further calls fail (i.e. Φ is not minimal or it is inconsistent) we still have an "approximate argument". So rather than throwing this away, we can treat it as an intermediate finding, and use it as part of an "approximate argument tree" which we can build with fewer calls to the ATP than building a complete argument tree, and this approximate argument tree can then be refined, as required, with the aim of getting closer to a complete argument tree. We formalize this next.

Approximate argumentation

An approximate argument is a pair $[\Phi, \alpha]$ where $\Phi \subseteq \mathcal{L}$ and $\alpha \in \mathcal{L}$. This is a very general definition. It does not assume that Φ is consistent, or that it entails α, or that it is even a subset of the knowledgebase Δ.

To focus our presentation in this paper, we will restrict consideration to a particular class of approximate arguments, namely entailments. If $\Phi \subseteq \Delta$ and $\Phi \vdash \alpha$, then $[\Phi, \alpha]$ is an **entailment**. Furthermore, we will consider some subclasses of entailments defined as follows: If $[\Phi, \alpha]$ is an entailment, and there is no $\Phi' \subset \Phi$ such that $\Phi' \vdash \alpha$, then $[\Phi, \alpha]$ is a **miniment**; If $[\Phi, \alpha]$ is an entailment, and $\Phi \not\vdash \bot$, then $[\Phi, \alpha]$ is an **altoment**; And if $[\Phi, \alpha]$ is a miniment, and $[\Phi, \alpha]$ is an altoment, then $[\Phi, \alpha]$ is a **preargument**. Each of these kinds of entailment is defined as a relaxation of the definition for an argument: The support of an entailment implies the consequent, but neither an entailment nor an altoment has a support that is necessarily a minimal set of assumptions for implying the consequent and neither an entailment nor a miniment is necessarily a consistent set of assumptions for implying the consequent.

Example 5. *Let $\Delta = \{\alpha, \neg\alpha \vee \beta, \gamma, \neg\beta, \neg\gamma\}$. So entailments for β include $\{A_1, A_2, A_3, A_4, A_5\}$ of which $\{A_1, A_3, A_5\}$ are altoments, $\{A_2, A_5\}$ are miniments, and A_5 is a preargument.*

$$\begin{array}{l}
A_1 = [\{\alpha, \neg\alpha \vee \beta, \gamma, \beta\}, \beta] \\
A_2 = [\{\gamma, \neg\gamma\}, \beta] \\
A_3 = [\{\alpha, \neg\alpha \vee \beta, \gamma\}, \beta] \\
A_4 = [\{\alpha, \neg\alpha \vee \beta, \gamma, \neg\gamma\}, \beta] \\
A_5 = [\{\alpha, \neg\alpha \vee \beta\}, \beta]
\end{array}$$

An altoment is a "potentially overweight proof" in the sense that there are more assumptions in the support than required for the consequent, and a miniment is a "minimal proof" in the sense that if any formula is removed from the support there will be insufficient assumptions in the support for the consequent to be entailed.

Some simple observations that we can make concerning entailments include: (1) If $[\Gamma, \alpha]$ is an altoment, then there is a $\Phi \subseteq \Gamma$ such that $\langle \Phi, \alpha \rangle$ is an argument; (2) If $[\Gamma, \alpha]$ is an entailment, then there is a $\Phi \subseteq \Gamma$ such that $[\Phi, \alpha]$ is a miniment; and (3) If $[\Phi, \alpha]$ is a preargument, then $\langle \Phi, \alpha \rangle$ is an argument.

An **approximate undercut** for an approximate argument $[\Phi, \alpha]$ is an approximate argument $[\Psi, \beta]$ such that $\beta \vdash \neg(\wedge\Phi)$ (where if $\Phi = \{\phi_1, .., \phi_n\}$, and $\langle \phi_1, \ldots, \phi_n \rangle$ is the canonical enumeration of Φ, then $\wedge\Phi$ is $\phi_1 \wedge .. \wedge \phi_n$).

An **approximate tree** is a tree T where each node is an approximate argument from Δ. There are various kinds of approximate tree. Here we define three particular kinds: (1) An **entailment tree** for α is an approximate tree where each node is an entailment and the root is for α; (2) An **altoment tree** for α is an approximate tree where each node is an altoment and the root is for α; and (3) A **preargument tree** for α is an approximate tree where each node is a preargument and the root is for α. For these trees, we do not impose that the children of a node are approximate undercuts, but in the way we construct them, we will aim for this.

Example 6. *In the following, T_1 is an entailment tree, T_2 is an altoment tree, and T_3 is a preargument tree.*

$$\begin{tikzcd}
& T_1 \quad [\{\beta, \neg\beta, \delta, \delta \vee \beta, (\beta \vee \delta) \rightarrow \alpha\}, \alpha] & \\
{[\{\phi, \neg\delta, \delta \vee \neg\beta\}, \neg\beta]} \arrow[ur] & & {[\{\phi, \neg\phi\}, \neg\delta]} \arrow[ul]
\end{tikzcd}$$

$$\begin{tikzcd}
& T_2 \quad [\{\beta, \delta, \delta \vee \beta, (\beta \vee \delta) \rightarrow \alpha\}, \alpha] & \\
{[\{\phi, \neg\delta, \delta \vee \neg\beta\}, \neg\beta]} \arrow[ur] & & {[\{\phi, \neg\beta, \neg\delta \vee \beta\}, \neg\delta]} \arrow[ul]
\end{tikzcd}$$

$$\begin{tikzcd}
T_3 \quad [\{\beta, \beta \rightarrow \alpha\}, \alpha] \\
{[\{\gamma, \gamma \rightarrow \neg\beta\}, \neg(\beta \wedge (\beta \rightarrow \alpha))]} \arrow[u]
\end{tikzcd}$$

Obviously, all preargument trees are altoment trees, and all altoment trees are entailment trees.

Definition 7. *A* **complete preargument tree** *is a preargument tree where (1) for no node $[\Phi, \beta]$ with ancestor nodes $[\Phi_1, \beta_1], \ldots, [\Phi_n, \beta_n]$ is Φ a subset of $\Phi_1 \cup \cdots \cup \Phi_n$ and (2) for each node with a preargument $[\Phi, \beta]$, the children nodes consist of all the prearguments of the form $[\Psi, \neg(\wedge\Phi)]$.*

The first condition above ensures that the support of each preargument on a branch has to include at least one premise that has not been used by its ancestors. The second condition above ensures that the children nodes of a node consist of all prearguments that are approximate undercuts negating the conjunction of the support of the parent. This mirrors the definition for an argument tree (Definition 6) and the following result shows they are isomorphic. As an illustration, T_3 in Example 6 is a complete preargument tree.

Proposition 1. *If T is a complete preargument tree, then there is an argument tree T' and there is a bijection f from the nodes in T to the nodes in T' such that for all the prearguments $[\Phi, \alpha]$ in T, $f([\Phi, \alpha]) = \langle \Phi, \alpha \rangle$.*

From the above result, we see that not only is each preargument effectively equivalent to an argument (i.e. each preargument $[\Phi, \alpha]$ is corresponds to an argument $\langle \Phi, \alpha \rangle$), but that also the constraints on the definition are sufficient for each approximate undercut to behave equivalently to a canonical undercut.

The following definition of refinement is a relationship that holds between some altoment trees. It holds when each altoment in T_2 uses the same or fewer assumptions than its corresponding altoment in T_1 and its claim is weaker or the same as its corresponding altoment in T_1. For this, let $\mathsf{Support}([\Phi, \alpha])$ be Φ, and let $\mathsf{Claim}([\Phi, \alpha])$ be α, where $[\Phi, \alpha]$ is an entailment.

Definition 8. *Let T_1 and T_2 be altoment trees. T_2 is a* **refinement** *of T_1 iff there is a bijection f from the nodes of T_1 to the nodes of T_2 such that for all nodes A in T_1, $\mathsf{Support}(f(A)) \subseteq \mathsf{Support}(A)$ and $\mathsf{Claim}(A) \vdash \mathsf{Claim}(f(A))$. We call f the* **refinement function**.

Proposition 2. *If T_2 is a refinement of T_1 with refinement function f, then for all $[\Phi_1, \alpha_1] \in T_1$, if $f([\Phi_1, \alpha_1]) = [\Phi_2, \alpha_2]$, then $\langle \Phi_2, \alpha_2 \rangle$ is more conservative than $\langle \Phi_1, \alpha_1 \rangle$.*

Refinement is useful because we can build a tree using altoments, and then refine those altoments as part of a process of obtaining a better approximate tree, and if required, a complete preargument tree. We consider this process more generally for entailments in the next section.

Constructing approximate trees

To render the construction, and improvement, of approximate trees implementable, we define the **revision steps** that can be undertaken on a tree T as follows where T' is the result of the revision step, and all entailments come from the knowledgebase Δ.

1. T' is obtained by **resupport** from T by taking an entailment $[\Phi, \alpha]$ in T and removing one formula, say ϕ, such that $[\Phi \setminus \{\phi\}, \alpha]$ is an entailment.
2. T' is obtained by **reconsequent** from T by replacing an entailment $[\Phi, \alpha]$ in T with entailment $[\Phi, \alpha']$ where $[\Psi, \beta]$ is the parent of $[\Phi, \alpha]$ in T and $[\Psi, \beta]$ is the parent of $[\Phi, \alpha']$ in T' and $\alpha \not\equiv \alpha'$ and $\alpha' \equiv \neg(\wedge\Psi)$.
3. T' is obtained by **expansion** from T by taking an entailment $[\Phi, \alpha]$ in T and adding an entailment $[\Psi, \beta]$ such that $[\Psi, \beta]$ is an approximate undercut of $[\Phi, \alpha]$ and it has not been shown that $\Psi \vdash \bot$.
4. T' is obtained by **contraction** from T by removing an entailment $[\Psi, \beta]$ (and all its offspring) such that $[\Psi, \beta]$ is a miniment and $\Psi \vdash \bot$.
5. T' is obtained by **deflation** from T by removing an entailment $[\Psi, \beta]$ (and all its offspring) such that $[\Psi, \beta]$ is a child of $[\Phi, \alpha]$ and $\Psi \cup \Phi \not\vdash \bot$.

We explain the revision steps as follows: Resupport weakens the support of an entailment to remove an unnecessary premise; Reconsequent strengthens the claim of an entailment so that it negates the conjunction of its parents support; Expansion adds a new approximate undercut to the tree (assuming that it has not been shown to have an inconsistent support); Contraction removes a node which has become an inconsistent miniment (after previously being subject to resupport); and Deflation removes a node with a support that is consistent with the support of its parent (after previously one or other being subject to resupport). Illustrations of using revision steps are given in Example 7 and Example 8.

Example 7. *Each of the following three trees is an altoment tree. Furthermore, T_2 is a resupport of T_1 and T_3 is an expansion of T_2.*

$$\begin{tikzcd}
T_1 & {[\{\alpha, \alpha \to \beta, \neg\gamma, \delta\}, \beta]} & T_2 & {[\{\alpha, \alpha \to \beta, \delta\}, \beta]} \\
 & {[\{\gamma, \gamma \to \neg\alpha, \sigma\}, \neg\alpha]} \arrow[u] & & {[\{\gamma, \gamma \to \neg\alpha, \sigma\}, \neg\alpha]} \arrow[u]
\end{tikzcd}$$

$$\begin{tikzcd}
T_3 & {[\{\alpha, \alpha \to \beta, \delta\}, \beta]} \\
 & {[\{\gamma, \gamma \to \neg\alpha, \sigma\}, \neg\alpha]} \arrow[u] \\
 & {[\{\neg\gamma\}, \neg\gamma]} \arrow[u]
\end{tikzcd}$$

The next result shows that we can obtain a complete preargument tree by some finite sequence of revision steps.

Theorem 1. *If T_n is a complete preargument tree for α, then there a sequence $\langle T_1, ..., T_n \rangle$ of approximate trees for α, s.t. T_1 is an altoment tree with just a root node, and for each i, where $i < n$, T_{i+1} is obtained by a revision step from T_i.*

Starting with an altoment tree for α that contains one node, and then using a sequence of revision steps to obtain a complete preargument tree, does not necessarily offer any computational advantages over constructing a complete argument tree directly (by finding an argument for the root, and then finding canonical undercuts to the root, and then by recursion, finding canonical undercuts to the canonical undercuts). To revise an approximate tree involves calls to the ATP, and so the more we revise an approximate tree, the less there is an efficiency advantage over constructing a complete argument tree. The real benefit of approximate argumentation is that an intermediate tree (in the sequence above) is more informative (than a partially constructed argument tree) since it tends to have more nodes, and thereby better indicate the range of potential conflicts arising in Δ. So, in comparison with an argument tree, an approximate tree is less cautious (it compromises on the correctness of the arguments used), is less expensive (it uses fewer calls to an ATP and each call made would also be made to construct each argument in the corresponding argument tree), and is more informative (in reporting on potential conflicts in Δ).

To obtain an entailment tree, we use the following algorithm which has an upper limit on the number of revision steps used.

Definition 9. *The algorithm* $\texttt{Generate}(\Delta, \Phi, \alpha, \lambda)$ *returns an entailment tree that is constructed in a fixed number of cycles (delineated by the number λ), for a knowledgebase Δ, and an altoment $[\Phi, \alpha]$ for the root of the entailment tree that is returned.*

```
Generate($\Delta, \Phi, \alpha, \lambda$)
        Let $T$ be the node $[\Phi, \alpha]$
        Let counter $= 0$
        While counter $\leq \lambda$ and there is
                    a revision step $T'$ of $T$
                Let $T = T'$
                Let counter $=$ counter $+ 1$
        Return $T$
```

The aim of using the above algorithm is to start with an entailment for α as the root, and then incrementally revise this entailment tree to get as close as possible to a complete preargument tree within an acceptable number of iterations.

Example 8. *Let $\Delta = \{\alpha, \alpha \rightarrow \beta, \delta, \gamma, \neg\alpha \vee \neg\beta, \beta\}$ For trees $T_1, .., T_7$ below, T_{i+1} is obtained from T_i by the revision step given in brackets. Let $\diamond$ denote the claim that is the negation of the conjunction of the support of its parent.*

$$[\{\alpha, \alpha \rightarrow \beta, \delta\}, \alpha \wedge \beta]$$

(3)

$$\begin{tikzcd}
{[\{\alpha, \alpha \rightarrow \beta, \delta\}, \alpha \wedge \beta]} \\
{[\{\alpha, \neg\alpha \vee \neg\beta\}, \neg\beta]} \arrow[u]
\end{tikzcd}$$

(3)

$$\begin{tikzcd}
& {[\{\alpha, \alpha \rightarrow \beta, \delta\}, \alpha \wedge \beta]} & \\
{[\{\alpha, \neg\alpha \vee \neg\beta\}, \neg\beta]} \arrow[ur] & & {[\{\neg\alpha \vee \neg\delta, \delta\}, \neg\alpha]} \arrow[ul]
\end{tikzcd}$$

(1)

$$\begin{tikzcd}
& {[\{\alpha, \alpha \rightarrow \beta\}, \alpha \wedge \beta]} & \\
{[\{\alpha, \neg\alpha \vee \neg\beta\}, \neg\beta]} \arrow[ur] & & {[\{\neg\alpha \vee \neg\delta, \delta\}, \neg\alpha]} \arrow[ul]
\end{tikzcd}$$

(2)

$$\begin{tikzcd}
& {[\{\alpha, \alpha \rightarrow \beta\}, \alpha \wedge \beta]} & \\
{[\{\alpha, \neg\alpha \vee \neg\beta\}, \diamond]} \arrow[ur] & & {[\{\neg\alpha \vee \neg\delta, \delta\}, \neg\alpha]} \arrow[ul]
\end{tikzcd}$$

(1)

$$\begin{tikzcd}
& {[\{\alpha, \alpha \rightarrow \beta\}, \alpha \wedge \beta]} & \\
{[\{\neg\alpha \vee \neg\beta\}, \diamond]} \arrow[ur] & & {[\{\neg\alpha \vee \neg\delta, \delta\}, \neg\alpha]} \arrow[ul]
\end{tikzcd}$$

(2)

$$\begin{tikzcd}
& {[\{\alpha, \alpha \rightarrow \beta\}, \alpha \wedge \beta]} & \\
{[\{\neg\alpha \vee \neg\beta\}, \diamond]} \arrow[ur] & & {[\{\neg\alpha \vee \neg\delta, \delta\}, \diamond]} \arrow[ul]
\end{tikzcd}$$

The following shows that if we run the $\texttt{Generate}(\Delta, \Phi, \alpha, \lambda)$ algorithm for sufficient time, it will eventually result in a complete preargument tree for α.

Theorem 2. *For all propositional knowledgebases Δ and altoment $[\Phi, \alpha]$, there is a $\lambda \in \mathbb{N}$ such that* $\texttt{Generate}(\Delta, \Phi, \alpha, \lambda)$ *returns an altoment tree T for α, and T is a complete preargument tree for α.*

Corollary 1. *Let $\langle T_1, ..., T_n \rangle$ be a sequence of entailment trees for α such that T_1 is an altoment tree with just a root node, and for each i, where $1 \leq i < n$, T_{i+1} is obtained by a revision step from T_i. If T_n is such that no further revision steps are possible on it, then T_n is a complete preargument tree for α.*

We can improve the algorithm by incorporating selection criteria to prefer some revision steps over others. For example, we may prefer: (1) an expansion that adds a node higher up the tree rather than lower down the tree; (2) resupport steps on nodes with large supports rather than those with small supports; (3) as many expansion steps as possible in order to get an impression of as many as possible of the conflicts that potentially exist. By adopting selection criteria, we aim to have more meaningful (from a user's perspective) approximate trees returned when the threshold for the number of revision steps undertaken by the algorithm.

Bias in approximate trees

In this section, we consider another way to bias the construction of approximate trees by amending the revision steps available. To explain and motivate this, we first require some further definitions. For an approximate tree T, each node in T is either an **attacking node** or a **defending node**. If a node A_r is the root, then A_r is a defending node. If a node A_i is a defending node, then any child A_j of A_i is an attacking node. If a node A_j is an attacking node, then any child A_k of A_j is a defending node. The intuition of this nomenclature is apparent in the following definition for judging.

Definition 10. *The* **win judge** *is a function, denoted* Win, *from the set of approximate trees to $\{yes, no\}$ such that* $\mathsf{Win}(T) = yes$ *iff $mark(A_r) = U$ where A_r is the root node of T. For a node A_i, $mark(A_i) = U$ when it is undefeated, and $mark(A_i) = D$ it is defeated. Deciding whether a node is defeated or undefeated depends on whether or not all its children are defeated: For all non-leaf nodes A_i, $mark(A_i) = D$ iff there is a child A_j of A_i s.t. $mark(A_j) = U$. For all leaves A_l, $mark(A_l) = U$.*

So $\mathsf{Win}(T)$ is yes when the approximate argument at the root is defended from all attacks. This is a well-studied criterion for argumentation (e.g. (García & Simari 2004)) though there are some interesting alternatives. However we stress at this point that we believe the primary purpose of logical argumentation is to highlight the key arguments and counterarguments in the knowledgebase Δ rather than applying a judge function. Nonetheless, the win judge is a useful way of assessing the conflicts in a knowledgebase.

Example 9. *Consider the following preargument tree T where $A_1 = [\{\delta, \delta \rightarrow \sigma\}, \sigma]$, $A_2 = [\{\neg\gamma, \neg\gamma \rightarrow \neg\delta\}, \diamond]$, $A_3 = [\{\alpha, \beta, \alpha \wedge \beta \rightarrow \gamma\}, \diamond]$, $A_4 = [\{\gamma \vee \delta\}, \diamond]$, $A_5 = [\{\neg\alpha\}, \diamond]$, and $A_6 = [\{\neg\beta\}, \diamond]$.*

$$\begin{tikzcd}
& & & A_1 \\
& & A_2 \arrow[ur] & \\
& A_3 \arrow[ur] & & A_4 \arrow[ul] \\
A_5 \arrow[ur] & & A_6 \arrow[ul] &
\end{tikzcd}$$

Hence, $\mathsf{Win}(T) = yes$ *holds. The intuition in this example is that A_4 is sufficient to defeat A_2 irrespective of A_3. And* *so the existence of A_5 and/or A_6 does not affect the ability of A_4 to defeat A_2 and hence allow A_1 to be undefeated.*

We now consider how we can incorporate bias into the construction of altoment trees. For this, we adapt the revision step of resupport as follows: T' is obtained by **attack-resupport** from T by taking an entailment $[\Phi, \alpha]$ for an attacking node in T and removing one formula, say ϕ, such that $[\Phi \setminus \{\phi\}, \alpha]$ is an entailment. Using this alternative to resupport, we can define the following type of biased tree.

Definition 11. *Let $\langle T_1, ..., T_n \rangle$ be a sequence of altoment trees for α such that T_1 is a node with an altoment $[\Phi, \alpha]$, and for each i, where $1 \leq i < n$, T_{i+1} is obtained by an attack-resupport, reconsequent, contraction, or deflation step from T_i, and when no such step is possible for T_i, then T_{i+1} is obtained by an expansion step of T_i. If T_n is such that no further step is possible, then T_n is an* **attack-bias tree** *for α started from $[\Phi, \alpha]$.*

For purposes of comparison, suppose we have a λ s.t. $\texttt{Generate}(\Delta, \Phi, \alpha, \lambda)$ returns a complete preargument tree T for α. In this case, we say that T is a complete preargument tree for α started from $[\Phi, \alpha]$.

Proposition 3. *If T is an attack-bias tree, then each attacking node is a preargument, whereas each defending node is an altoment but not necessarily a preargument.*

The above result means that in an attack-bias tree it is easier to undercut a defending node than an attacking node. This leads us to the following result.

Proposition 4. *Let T_a be a attack-bias tree started from $[\Phi, \alpha]$, and let T_p be a complete preargument tree started from $[\Phi, \alpha]$. If $\mathsf{Win}(T_a) = yes$, then $\mathsf{Win}(T_p) = yes$.*

Constructing an attack-bias tree may involve fewer revision steps than constructing a complete preargument tree, and from the above, it provides a bound on the outcome of the corresponding complete preargument tree. We can define further biased constructions such as for a defence-bias tree by restricting resupport steps to the defending nodes.

Discussion

Much progress has been made on developing formalisms for argumentation. Some algorithms for argumentation have been developed (for example (Kakas & Toni 1999; Cayrol, Doutre, & Mengin 2001; Baroni & Giacomin 2002; García & Simari 2004)). However, relatively little progress has been made in developing techniques for overcoming the computational challenges of constructing arguments. Even though there is a proposal for storing information about previous calls to an ATP in a form of lemma generation called contouring (Hunter 2006), and there are proposals for making defeasible logic programming more efficient by storing non-instantiated arguments in a database, again for reducing the number of called to an ATP (Capobianco, Chesnevar, & Simari 2005), and by the use of prunning strategies (Chesnevar, Simari, & Godo 2005), there does appear to be a pressing need to look more widely for approaches for improving the efficiency of argumentation.

In this paper, we have introduced a framework for approximate arguments that can be used as useful intermediate results when undertaking argumentation using an ATP. Finding approximate arguments requires fewer calls to an ATP, but it involves compromising the correctness of arguments. Our next step is to undertake empirical studies with the algorithm to investigate this efficiency-correctness trade-off.

Approximate arguments may also be a useful vehicle for studying approximate arguments as arising in human cognition, and may be important for characterising some kinds of discussions and negotiations in multi-agent dialogue systems where agents may put forward approximate arguments that are not altoments, or even not entailments (perhaps for competitive reasons or through ignorance).

Whilst our presentation is based on a particular approach to logic-based argumentation, we believe the proposal could be adapted for a range of other logic-based approaches to argumentation (for example (García & Simari 2004; Amgoud & Cayrol 2002)) by adapting our definitions for approximate arguments and for revision steps.

Acknowledgements

The author is very grateful to Philippe Besnard for feedback on an earlier draft of this paper.

References

Amgoud, L., and Cayrol, C. 2002. A model of reasoning based on the production of acceptable arguments. *Annals of Mathematics and Artificial Intelligence* 34:197–216.

Baroni, P., and Giacomin, M. 2002. Argumentation through a distributed self-stabilizing approach. *J. Experimental & Theoretical AI* 14(4):273–301.

Besnard, P., and Hunter, A. 2001. A logic-based theory of deductive arguments. *Artificial Intelligence* 128:203–235.

Besnard, P., and Hunter, A. 2005. Practical first-order argumentation. In *Proc. of the 20th National Conference on Artificial Intelligence (AAAI'2005)*, 590–595. MIT Press.

Capobianco, M.; Chesnevar, C.; and Simari, G. 2005. Argumentation and the dynamics of warranted beliefs in changing environments. *International Journal on Autonomous Agents and Mutliagent Systems* 11:127–151.

Cayrol, C.; Doutre, S.; and Mengin, J. 2001. Dialectical proof theories for the credulous preferred semantics of argumentation frameworks. In *Proc. ECSQARU'01*, volume 2143 of *LNCS*, 668–679. Springer.

Chesnevar, C.; Maguitman, A.; and Loui, R. 2001. Logical models of argument. *ACM Comp. Surveys* 32:337–383.

Chesnevar, C.; Simari, G.; and Godo, L. 2005. Computing dialectical trees efficiently in possibilistic defeasible logic programming. In *Proceedings of the 8th Interntational Logic Programming and Non-monotonic Reasoning Conference*, Lecture Notes in Computer Science. Springer.

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

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

Eiter, T., and Gottlob, G. 1995. The complexity of logic-based abduction. *Journal of the ACM* 42:3–42.

García, A., and Simari, G. 2004. Defeasible logic programming: An argumentative approach. *Theory and Practice of Logic Programming* 4(1):95–138.

Hunter, A. 2006. Contouring of knowledge for intelligent searching for arguments. In *Proceedings of the European Conference on Artificial Intelligence (ECAI'06)*. (in press).

Kakas, A., and Toni, F. 1999. Computing argumentation in logic programming. *J. Logic and Computation* 9:515–562.

Koriche, F. 2002. Approximate coherence-based reasoning. *J. of Applied Non-classical Logics* 12(2):239–258.

Levesque, H. 1984. A logic of implicit and explicit belief. In *Proceedings of the National Conference on Artificial Intelligence (AAAI'84)*, 198–202.

Prakken, H., and Sartor, G. 1997. Argument-based extended logic programming with defeasible priorities. *Journal of Applied Non-Classical Logics* 7:25–75.

Prakken, H., and Vreeswijk, G. 2000. Logical systems for defeasible argumentation. In Gabbay, D., ed., *Handbook of Philosophical Logic*. Kluwer.

Schaerf, M., and Cadoli, M. 1995. Tractable reasoning via approximation. *Artificial Intelligence* 74:249–310.