Contouring of Knowledge for Intelligent Searching for Arguments

Anthony Hunter[1]

Abstract. A common assumption for logic-based argumentation is that an argument is a pair $\langle \Phi, \alpha \rangle$ where Φ is a 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 these give us different options for argumentation. For a variety of logics, in particular for classical logic, there is a need to develop intelligent techniques for generating arguments. Since building a constellation of arguments and counterarguments involves repeatedly querying a knowledgebase, we propose a framework based on what we call "contours" for storing information about a knowledgebase that provides boundaries on what is provable in the knowledgebase. Using contours allows for more intelligent searching of a knowledgebase for arguments and counterarguments.

1 INTRODUCTION

Argumentation is a vital aspect of intelligent behaviour by humans. To capture this, there are a number of proposals for logic-based formalisations of argumentation (for reviews see [14, 7]). These proposals allow for the representation of arguments for and against some conclusion, 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 (the support) is a minimal consistent set of formulae that proves the second item (the claim) which is a formula (see for example [3, 10, 4, 1, 11, 5]).

Problematically, finding arguments and counterarguments involves much searching of a knowledgebase to ensure they are all found (exhaustiveness), and to ensure each of them has a minimal consistent set of premises entailing the claim (correctness). Proof procedures and algorithms have been developed for finding preferred arguments from a knowledgebase following for example Dung's preferred semantics (see for example [13, 12, 6, 8, 9]). However, these procedures and algorithms do not offer any ways of ameliorating the cost of repeatedly querying a knowledgebase as part of the process of ensuring exhaustiveness and correctness. In this paper, we address this computational inefficiency by presenting a new technique for intelligently searching a knowledgebase for arguments and counterarguments.

2 LOGICAL ARGUMENTATION

In this section we review an existing proposal for logic-based argumentation [4]. We consider a classical propositional language with deduction denoted by $\vdash$. We use $\alpha, \beta, \gamma, \ldots$ to denote formulae and $\Delta, \Phi, \Psi, \ldots$ to denote sets of formulae.

[1] UCL Department of Computer Science, London, UK. Email: a.hunter@cs.ucl.ac.uk

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 of arbitrary complexity. 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) $\Phi \not\vdash \bot$; (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 $\Delta = \{\alpha, \alpha \rightarrow \beta, \gamma \rightarrow \neg\beta, \gamma, \delta, \delta \rightarrow \beta, \neg\alpha, \neg\gamma\}$. Some arguments are $\langle \{\alpha, \alpha \rightarrow \beta\}, \beta \rangle$, $\langle \{\neg\alpha\}, \neg\alpha \rangle$, $\langle \{\alpha \rightarrow \beta\}, \neg\alpha \vee \beta \rangle$, and $\langle \{\neg\gamma\}, \delta \rightarrow \neg\gamma \rangle$.*

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 $\langle \{\alpha, \alpha \rightarrow \beta\}, \beta \rangle$.*

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

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 [4]).

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

$$\begin{array}{cc} \langle \{\alpha \vee \beta, \neg\beta\}, \alpha \vee \neg\delta \rangle & \langle \{\delta \leftrightarrow \beta, \neg\beta\}, \alpha \vee \neg\delta \rangle \\ \uparrow & \uparrow \\ \langle \{\alpha \rightarrow \gamma, \neg\gamma\}, \diamond \rangle & \langle \{\alpha \vee \beta, \alpha \rightarrow \gamma, \neg\gamma\}, \diamond \rangle \end{array}$$

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 [5].

3 MOTIVATION FOR CONTOURING

We now turn to automating the construction of arguments and counterarguments. It is tempting to think that automated theorem proving 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 particular 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 $\Phi \nvdash \bot$ 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 $\Phi \nvdash \alpha$, 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 $\Phi \vdash \alpha$, the second call is to ensure that $\Phi \nvdash \bot$, the remaining calls are to ensure that there is no subset Φ' of Φ such that $\Phi' \vdash \alpha$.

We can now summarise all the queries that are posted to the ATP for finding all the arguments for α from the knowledgebase Δ. We summarise it by the set $\mathsf{NaiveQuerying}(\Delta, \alpha)$ defined next.

Definition 7. *For a knowledgebase Δ and a formula α,*

$$\mathsf{NaiveQuerying}(\Delta, \alpha) = \{(\Phi?\alpha) \mid \Phi \subseteq \Delta\} \cup \{(\Phi?\bot) \mid \Phi \subseteq \Delta\}$$

Proposition 1. *For Δ and α, if $|\Delta| = n$, then $|\mathsf{NaiveQuerying}(\Delta, \alpha)| = 2^{n+1}$.*

Clearly, by using all the queries in $\mathsf{NaiveQuerying}(\Delta, \alpha)$, we are taking no account of any of the results that we have gained at any intermediate stage. In other words, we are not being intelligent. Yet if we want to harness automated reasoning, we need principled means for intelligently querying an ATP in order to search a knowledgebase for arguments.

Now when we think of the more difficult problem of not just finding all arguments for α, but then finding all undercuts to these arguments, and by recursion, undercuts to these undercuts. During the course of building an argument tree, there will be repeated attempts to ask the same query, and there will be repeated attempts to ask a query with the same support set. But, importantly, each time a particular subset is tested for a particular claim, we gain more information about Δ. So instead of taking the naive approach of always throwing the result of querying away, we see that this information can be collated about subsets to help guide the search for further arguments and counterarguments. For example, if we know that a particular subset Φ is such that $\Phi \nvdash \alpha$, and we are looking for an argument with claim $\alpha \wedge \beta$, then we can infer that $\Phi \nvdash \alpha \wedge \beta$, and there is no need to test Φ for this claim (i.e. it is not useful to make the call $\Phi?\alpha \wedge \beta$).

So as we undertake more tests of Δ for various subsets of Δ and for various claims, we build up a picture of Δ. We can consider these being stored as contours on the (cartographical) map of $\wp(\Delta)$. We formalise this in Section 4.

To make our presentation more concise, from now on, we refer to each subset of Δ by a binary number. Suppose Δ has cardinality n, then we adopt the arbitrary enumeration $\langle \alpha_1, .., \alpha_n \rangle$ of Δ, presented in Section 2. Using this, we can then represent any subset Φ of Δ by

a n digit binary number of the form $d_1, ..., d_n$. For the ith formula (i.e. α_i) in $\langle \alpha_1, .., \alpha_n \rangle$, if α_i is in Φ, then the ith digit (i.e. d_i) in $d_1, ..., d_n$ is 1, and if α_i is not in Φ, then the ith digit (i.e. d_i) in $d_1, ..., d_n$ is 0. For example, for the enumeration $\langle \alpha, \beta \wedge \neg\gamma, \gamma \vee \epsilon \rangle$, 000 is $\{\}$, 100 is $\{\alpha\}$, 010 is $\{\beta \wedge \neg\gamma\}$, 001 is $\{\gamma \vee \epsilon\}$, 110 is $\{\alpha, \beta \wedge \neg\gamma\}$, 111 is $\{\alpha, \beta \wedge \neg\gamma, \gamma \vee \epsilon\}$, etc.

Since we will be considering the power set of a knowledgebase, the following definition of the MaxWidth funtion will be useful for us: If n is $|\Delta|$ and m is $\lfloor n/2 \rfloor$ (i.e. m is the greatest integer less than or equal to $n/2$), then $\mathsf{MaxWidth}(n)$ is $n!/(n-m)!m!$. In other words, of all cardinalities for subsets of Δ, the most numerous are those of cardinality $\lfloor n/2 \rfloor$, and so the MaxWidth function gives the number of subsets of Δ of cardinality $\lfloor n/2 \rfloor$.

4 FRAMEWORK FOR CONTOURING

We start with the ideal situation where we have substantial information about the knowledgebase Δ.

Definition 8. *The* **contour** *for α is $\mathsf{C}(\alpha) = \{\Phi \subseteq \Delta \mid \Phi \vdash \alpha$ and there is no $\Psi \subset \Phi$ such that $\Psi \vdash \alpha\}$, the* **uppercontour** *for α is $\mathsf{U}(\alpha) = \{\Phi \subseteq \Delta \mid \Phi \vdash \alpha\}$, and the* **lowercontour** *for α, is $\mathsf{L}(\alpha) = \{\Phi \subseteq \Delta \mid \Phi \nvdash \alpha\}$.*

Clearly, $\mathsf{C}(\alpha) \subseteq \mathsf{U}(\alpha)$, and $\mathsf{L}(\alpha) \cup \mathsf{U}(\alpha)$ is $\wp(\Delta)$, and $\mathsf{L}(\alpha) \cap \mathsf{U}(\alpha)$ is $\emptyset$. Furthermore, it is possible that any set in $\mathsf{L}(\alpha)$, $\mathsf{C}(\alpha)$, or $\mathsf{U}(\alpha)$ is inconsistent. Obviously, all the sets in $\mathsf{U}(\bot)$ are inconsistent, and none in $\mathsf{L}(\bot)$ is inconsistent.

Proposition 2. *For any $\Phi \subseteq \Delta$, $\Phi \in \mathsf{C}(\alpha)$ iff (1) for all $\Psi \in \mathsf{U}(\alpha)$, $\Psi \not\subset \Phi$, and (2) for all $\Psi \in \mathsf{L}(\alpha)$, $\Phi \not\subseteq \Psi$.*

Example 5. *For $\langle \alpha \wedge \neg\alpha, \beta, \neg\beta \rangle$, $\mathsf{U}(\bot) = \{100, 110, 101, 011, 111\}$, $\mathsf{C}(\bot) = \{100, 011\}$, $\mathsf{L}(\bot) = \{000, 010, 001\}$, $\mathsf{U}(\alpha \vee \beta) = \{100, 010, 101, 110, 011, 111\}$, $\mathsf{C}(\alpha \vee \beta) = \{100, 010\}$, and $\mathsf{L}(\alpha \vee \beta) = \{000, 001\}$.*

We can use contours directly to generate arguments. For this, we obtain $\mathsf{L}(\bot)$ from $\mathsf{C}(\bot)$ as follows: $\mathsf{L}(\bot) = \{\Phi \subseteq \Delta \mid \Phi \subseteq \Psi$ and $\Psi \in \mathsf{C}(\bot)\}$.

Proposition 3. *For any Φ and α, $\Phi \in \mathsf{C}(\alpha) \cap \mathsf{L}(\bot)$ iff $\langle \Phi, \alpha \rangle$ is an argument.*

This means that if we have $\mathsf{C}(\alpha)$ and $\mathsf{C}(\bot)$, we have all the knowledge we require to generate all arguments for α. By this we mean, we do not need to use the ATP.

However, we may be in a position where we have some γ for which we want to generate arguments, but for which we do not have $\mathsf{C}(\gamma)$. In general, we cannot expect to have a contour for every possible claim for which we may wish to contruct an argument. To address this, we will require an ATP, but we can focus our search for the arguments, so that we can reduce the number of calls that we make to the ATP. For this we can identify Boolean constituents of γ for which we do have the contours. To support this, we require the following definition.

Definition 9. *For any α, β, the* **contour conjunction** *and* **contour disjunction** *operators, denoted $\oplus$ and $\otimes$ respectively, are defined as follows.*

$$\mathsf{C}(\alpha) \oplus \mathsf{C}(\beta) = \mathsf{C}(\alpha) \cup \mathsf{C}(\beta) \cup \{\Phi \cup \Psi \mid \Phi \in \mathsf{C}(\alpha) \& \Psi \in \mathsf{C}(\beta)\}$$
$$\mathsf{C}(\alpha) \otimes \mathsf{C}(\beta) = \mathsf{C}(\alpha) \cup \mathsf{C}(\beta) \cup \{\Phi \cap \Psi \mid \Phi \in \mathsf{C}(\alpha) \& \Psi \in \mathsf{C}(\beta)\}$$

Example 6. *Consider $\langle \alpha, \beta, \alpha \vee \beta \rightarrow \gamma, \alpha \vee \beta \rightarrow \delta \rangle$. So $\mathsf{C}(\gamma) = \{1010, 0110\}$ and $\mathsf{C}(\delta) = \{1001, 0101\}$. Hence,*

$$\mathsf{C}(\gamma) \oplus \mathsf{C}(\delta) = \{1011, 0111, 1111, 1010, 0110, 1001, 0101\}$$
$$\mathsf{C}(\gamma) \otimes \mathsf{C}(\delta) = \{1000, 0000, 0100, 1010, 0110, 1001, 0101\}$$

Note, $\mathsf{C}(\gamma \wedge \delta) = \{1011, 0111\}$ and $\mathsf{C}(\gamma \vee \delta) = \{1010, 0110, 1001, 0101\}$.

Clearly, the $\oplus$ and $\otimes$ operators are associative and commutative. We also obtain the following containment results.

Proposition 4. *For any α, β, $\mathsf{C}(\alpha \wedge \beta) \subseteq (\mathsf{C}(\alpha) \oplus \mathsf{C}(\beta))$*

Proposition 5. *For any α, β, $\mathsf{C}(\alpha \vee \beta) \subseteq (\mathsf{C}(\alpha) \otimes \mathsf{C}(\beta))$*

So if we are looking for an argument for $\alpha \wedge \beta$, we look in the set $\mathsf{C}(\alpha) \oplus \mathsf{C}(\beta)$. Similarly if we are looking for an argument for $\alpha \vee \beta$, we look in the set $\mathsf{C}(\alpha) \oplus \mathsf{C}(\beta)$. Furthermore, the number of sets to consider in $\mathsf{C}(\alpha) \oplus \mathsf{C}(\beta)$, and similarly in $\mathsf{C}(\alpha) \otimes \mathsf{C}(\beta)$, is a quadratic function of the number of sets in each of $\mathsf{C}(\alpha)$ and $\mathsf{C}(\beta)$.

Proposition 6. *For any α and β, if $|\mathsf{C}(\alpha)| = p$ and $|\mathsf{C}(\beta)| = q$, then $|\mathsf{C}(\alpha) \oplus \mathsf{C}(\beta)| \leq pq + p + q$ and $|\mathsf{C}(\alpha) \otimes \mathsf{C}(\beta)| \leq pq + p + q$.*

Now we consider an operator to facilitate the use of a contour $\mathsf{C}(\alpha)$ to find arguments with the claim $\neg\alpha$.

Definition 10. *For any $X \subseteq \wp(\Delta)$, the* **shift** *operator, denoted $\div$, is defined as follows.*

$$\div X = \{\Phi \subseteq \Delta \mid \text{ for all } \Psi \in X, \Phi \not\subseteq \Psi \text{ and } \Psi \not\subseteq \Phi\}$$

Example 7. *Consider $\langle \alpha \vee \beta, \neg\alpha, \neg\beta, \neg\gamma \wedge \delta, \neg\delta \wedge \beta \rangle$. So $\mathsf{C}(\beta)$ is $\{11000, 00001\}$. Hence, $\div\mathsf{C}(\beta)$ is $\{10100, 10010, 10110, 01100, 01010, 01110, 00100, 00010, 00110\}$. By comparison, $\mathsf{C}(\neg\beta)$ is $\{00100\}$.*

Whilst $\div$ is, in a weak sense, a kind of complementation operator, properties such as $\div(\div\mathsf{C}(\alpha)) = \mathsf{C}(\alpha)$ do not hold. But we do have the following useful property which shows how given $\mathsf{C}(\alpha)$, we can use $\div\mathsf{C}(\alpha)$ to focus our search for an argument for $\neg\alpha$.

Proposition 7. *For any α, $(\mathsf{C}(\neg\alpha) \cap \mathsf{L}(\bot)) \subseteq (\div\mathsf{C}(\alpha) \cap \mathsf{L}(\bot))$*

We conclude this section by considering undercuts. For any undercut, the claim is the negation of the support of its parent, and the support of the parent is some subset of Δ. Suppose the support of the parent is $\{\delta_1, .., \delta_k\}$, so the claim of any undercut is $\neg(\delta_1 \wedge .. \wedge \delta_k)$. This claim is equivalent to $\neg\delta_1 \vee .. \vee \neg\delta_k$. So, if we have contours for each of $\neg\delta_1$,..,$\neg\delta_k$, we can focus our search for any these undercut in the space delineated by $\mathsf{C}(\neg\delta_1) \otimes ... \otimes \mathsf{C}(\neg\delta_k)$. So we may consider keeping a contour for the negation of each element in Δ.

5 USING CONTOURS

We now consider how contours can improve our use of an ATP by using fewer queries than $\mathsf{NaiveQuerying}(\Delta, \alpha)$. For a knowledgebase Δ, a formula α, and a set of contours Θ, if $\mathsf{C}(\alpha) \in \Theta$, then we know it is not necessary to make any calls to the ATP.

If we are looking for arguments with a claim that is a conjunction of formulae for which we have contours, the queries to the ATP are delineated by the following function.

$$\mathsf{ConjunctionQuerying}(\{\mathsf{C}(\alpha), \mathsf{C}(\beta), \mathsf{C}(\bot)\}, \alpha \wedge \beta)$$
$$= \{(\Phi?\alpha) \mid \Phi \in \mathsf{C}(\alpha) \oplus \mathsf{C}(\beta) \text{ and } \Phi \in \mathsf{L}(\bot)\}$$

Similarly, if we are looking for arguments with a claim that is a disjunction of formulae for which we have contours, the queries to the ATP are delineated by the following function.

$$\begin{aligned}&\mathsf{DisjunctionQuerying}(\{\mathsf{C}(\alpha),\mathsf{C}(\beta),\mathsf{C}(\bot)\},\alpha\vee\beta)\\&\quad=\{(\Phi?\alpha)\mid\Phi\in\mathsf{C}(\alpha)\otimes\mathsf{C}(\beta)\text{ and }\Phi\in\mathsf{L}(\bot)\}\end{aligned}$$

Proposition 8. *For Δ, $\alpha\wedge\beta$, and $\alpha\vee\beta$, if $|\mathsf{C}(\alpha)|=p$ and $|\mathsf{C}(\beta)|=q$ and $r=pq+p+q$ then*

$$\begin{aligned}&|\mathsf{ConjunctionQuerying}(\{\mathsf{C}(\alpha),\mathsf{C}(\beta),\mathsf{C}(\bot)\},\alpha\wedge\beta)|\leq r\\&|\mathsf{DisjunctionQuerying}(\{\mathsf{C}(\alpha),\mathsf{C}(\beta),\mathsf{C}(\bot)\},\alpha\vee\beta)|\leq r\end{aligned}$$

Now we turn to finding undercuts which can be a substantial part of the cost of constructing an argument tree, and indeed can involve many times more work than just finding the root of an argument tree. If we have a contour for the negation of each formula in Δ, then we can use the following delineation on queries.

$$\begin{aligned}&\mathsf{UndercutQuerying}(\{\mathsf{C}(\neg\delta_1),..,\mathsf{C}(\neg\delta_k),\mathsf{C}(\bot)\},\neg\delta_1\vee..\vee\neg\delta_k)\\&=\{(\Phi?\alpha)\mid\Phi\in\mathsf{C}(\neg\delta_1)\otimes..\otimes\mathsf{C}(\neg\delta_k)\text{ and }\Phi\in\mathsf{L}(\bot)\}\end{aligned}$$

Example 8. *For $\langle\alpha,\beta,\delta\rightarrow(\neg\alpha\vee\neg\beta),\delta,\gamma,\gamma\rightarrow\neg\delta\rangle$, $\mathsf{C}(\neg\alpha)=\{011100,000111\}$, $\mathsf{C}(\neg\beta)=\{101100,000111\}$, $\mathsf{C}(\neg(\delta\rightarrow(\neg\alpha\vee\neg\beta)))=\{110100,000111\}$, $\mathsf{C}(\neg\delta)=\{000011\}$, $\mathsf{C}(\neg\gamma)=\{111100,000101\}$, $\mathsf{C}(\neg(\gamma\rightarrow\neg\delta))=\{111100,000110\}$, and $\mathsf{C}(\bot)=\{111100,000111\}$. Consider undercuts for $\langle110000,\alpha\wedge\beta\rangle$, i.e. those with the claim $\neg\alpha\vee\neg\beta$. So* $\mathsf{UndercutQuerying}(\{\mathsf{C}(\neg\alpha),\mathsf{C}(\neg\beta),\mathsf{C}(\bot)\},\neg\alpha\vee\neg\beta)=\{(001100?\neg\alpha\vee\neg\beta),(011100?\neg\alpha\vee\neg\beta),(101100?\neg\alpha\vee\neg\beta)\}$. *By comparison, $\mathsf{C}(\neg\alpha\vee\neg\beta)=\{001100\}$.*

Proposition 9. *For Δ and α, if $|\Delta|=n$, then* $0\leq|\mathsf{UndercutQuerying}(\{\mathsf{C}(\neg\delta_1),..,\mathsf{C}(\neg\delta_k),\mathsf{C}(\bot)\},\neg\delta_1\vee..\vee\neg\delta_k)|\leq 2^{n-2}+1$.

The best case for the UndercutQuerying function is when $k=1$. This is when the argument being undercut has a support with just one premise. So the undercut has to just undercut this premise, and therefore, the undercut can be obtained directly from the relevant contour $\mathsf{C}(\neg\delta_1)$ without recourse to the ATP. The worst case for the UndercutQuerying function is when $|\Delta|=n$ and $k=(\mathsf{MaxWidth}(n)-1)$, and there is a $\Phi_1\in\mathsf{C}(\neg\delta_1)$ and .. and a $\Phi_k\in\mathsf{C}(\neg\delta_k)$ such that for each $\Phi_i\in\{\Phi_1,..,\Phi_k\}$, $|\Phi_i|=\lfloor n/2\rfloor$, and for all $\Phi_i,\Phi_j\in\{\Phi_1,..,\Phi_k\}$, $\Phi_i\neq\Phi_j$. The worst case is clearly an extreme situation and it would seem that on average the number of queries raised by the UndercutQuerying function would be significantly lower, as can be seen for lower values of k using iterated applications of Proposition 8.

6 PARTIAL CONTOURS

Let Π be a set of answers to queries to the ATP. In other words, for a series of queries $(\Phi_1?\alpha_1)$, ..., $(\Phi_n?\alpha_n)$, we have obtained a set of answers $\Pi=\{\pi_1,..,\pi_n\}$, where for each $\pi_i\in\Pi$, the answer is either of the form $\Phi_i\vdash\alpha_i$ or of the form $\Phi_i\nvdash\alpha_i$. Using Π, we want to generate a partial contour $\mathsf{C}(\Pi,\alpha)$ for α. We will define it so that $\mathsf{C}(\Pi,\alpha)$ is a subset of $\mathsf{C}(\alpha)$ calculated on the basis of Π.

Definition 11. *For Π and α, the* **partial uppercontour**, *is* $\mathsf{U}(\Pi,\alpha)=\{\Phi\subseteq\Delta\mid\Psi\subseteq\Phi$ *and* $(\Psi\vdash\alpha)\in\Pi\}$ *and the* **partial lowercontour**, *is* $\mathsf{L}(\Pi,\alpha)=\{\Phi\subseteq\Delta\mid\Phi\subseteq\Psi$ *and* $(\Psi\nvdash\alpha)\in\Pi\}$.

For $\Pi\subseteq\Pi'$, $\mathsf{U}(\Pi,\alpha)\subseteq\mathsf{U}(\Pi',\alpha)$ and $\mathsf{L}(\Pi,\alpha)\subseteq\mathsf{L}(\Pi',\alpha)$.

Definition 12. *For Π, the* **partial contour** *for $\mathsf{C}(\Pi,\alpha)$ is* $\mathsf{C}(\Pi,\alpha)=\{\Phi\in\mathsf{U}(\Pi,\alpha)\mid$ *for all* $\phi\in\Phi,(\Phi\setminus\{\phi\})\in\mathsf{L}(\Pi,\alpha)\}$.

So from Π, we construct the partial uppercontour and the partial lower contour for some α, and then from these we can construct the partial contour for α.

Example 9. *For $\langle\alpha,\neg\alpha,\beta\rangle$, let* $\Pi=\{(111\vdash\bot),(101\nvdash\bot),(011\nvdash\bot),(100\nvdash\bot),(110\vdash\bot),(010\nvdash\bot)\}$. *Hence,* $\mathsf{C}(\Pi,\bot)=\{110\}$.

We use partial contours in the same way as we use full contours. The only proviso is that with partial contours, we have incomplete information about the knowledgebase. So for example, since $\mathsf{C}(\Pi,\alpha)\subseteq\mathsf{C}(\alpha)$, we are not guaranteed to obtain all arguments for α from just using $\mathsf{C}(\Pi,\alpha)$. However, for any $\Phi\in\mathsf{C}(\Pi,\alpha)$, we know that Φ or any superset of Φ implies α, and any proper subset does not imply Φ. So any element in $\mathsf{C}(\Pi,\alpha)$ can have a profound effect on searching for arguments. We use the shift operator to formalise this.

Proposition 10. *If $\langle\Phi,\alpha\rangle$ is an argument, then* $\Phi\in((\mathsf{C}(\Pi,\alpha)\cup\div\mathsf{C}(\Pi,\alpha))$.

Proposition 11. *If $\Phi\in\mathsf{C}(\Pi,\alpha)\cap\mathsf{L}(\bot)$, then $\langle\Phi,\alpha\rangle$ is an argument.*

We can delineate the queries sent to the ATP when searching for an argument for α, with a partial contour $\mathsf{C}(\Pi,\alpha)$, using the following definition, where $\mathsf{M}(\Pi,\bot)$ is $\mathsf{L}(\Pi,\bot)\cup\mathsf{U}(\Pi,\bot)$).

$$\begin{aligned}&\mathsf{PartialQuerying}(\{\mathsf{C}(\Pi,\alpha),\mathsf{C}(\Pi,\bot)\},\alpha)=\\&\quad\{(\Phi?\alpha)\mid\Phi\in\div\mathsf{C}(\Pi,\alpha)\text{ and }\Phi\notin\mathsf{U}(\Pi,\bot)\}\\&\qquad\cup\{(\Phi?\bot)\mid\Phi\in(\div\mathsf{C}(\Pi,\alpha)\cup\mathsf{C}(\Pi,\alpha))\text{ and }\Phi\notin\mathsf{M}(\Pi,\bot)\}\end{aligned}$$

The utility of a partial contour is dependent on the membership of Π. However, we do have $\mathsf{PartialQuerying}(\{\mathsf{C}(\Pi,\alpha),\mathsf{C}(\Pi,\bot)\},\alpha)$ being a subset of $\mathsf{SimpleQuerying}(\Delta,\mathsf{C}(\bot),\alpha)$ and as Π increases, the difference is increasingly marked. So in any case, whatever is in a partial contour can reduce the number of calls to the ATP.

Now we turn to the question of what are the bounds on the number of queries (i.e. size of Π) to build a contour. In the worst case, for any α, to ensure $\mathsf{C}(\Pi,\alpha)=\mathsf{C}(\alpha)$, it is necessary for Π to have 2^n queries where $\Pi=\{(\Phi?\alpha)\mid\Phi\subseteq\Delta\}$. However, we can take a more intelligent approach to decrease Π. For example, if we generate Π dynamically, once we have enough information from Π to add a particular $\Phi\subseteq\Delta$ to a particular contour, we have no need to seek supersets or subsets of Φ for that contour. Furthermore, we can envisage that the contours are built over time, as a by-product of using a knowledgebase. So each time the ATP is queried, the answer to the query is added to Π. As items are added to Π, the contours are constructed incrementally. We can therefore think of contours as being formed by a kind of lemma generation.

Finally, if we use partial contours, we do not even need to assume that the knowledgebase is fixed, since every partial contour of a knowledgebase Δ is a partial contour of a knowledgebase $\Delta\cup\Delta'$ for any Δ'.

7 STORING CONTOURS

So far we have made a case for the value of using contours. Even in the worst case, they offer an improvement over naive searching for arguments and counterarguments. By maintaining appropriate contours, the numbers of calls to the ATP is always decreased. The downside to maintaining contours, or even partial contours, is the amount of information that needs to be kept about the knowledgebase over time.

Proposition 12. *For any α, if $|\Delta| = n$, then $1 \leq \mathsf{C}(\alpha) \leq \mathsf{MaxWidth}(n)$.*

So for example, if the cardinality of our knowledgebase is 10, we may in the worst case, have a cardinality for $\mathsf{C}(\bot)$ of 252. This is not surprising given that $\mathsf{C}(\bot)$ is the set of minimally inconsistent subsets of Δ. Of course, this is a worst case scenario, and it would indicate a remarkably inconsistent knowledgebase for which it would be difficult to imagine arising in the real-world. Nonetheless, it raises the question of whether we could compromise on the information we retain from Π, and yet still reduce the number of queries to the ATP. We address this next.

The following result suggests that a simple solution is that when a contour or partial contour is being constructed, it appears to be too large to be manageable, it may be better to erase it, and construct the elements of the contour as and when required, and furthermore, if it is sufficiently large, it is relatively straightforward to find them because they are the subsets of Δ of cardinality $\lfloor n/2 \rfloor$.

Proposition 13. *Let $|\Delta| = n$. For any α, as the cardinality of $\mathsf{C}(\alpha)$ increases towards $\mathsf{MaxWidth}(n)$, the average size of the elements of $\mathsf{C}(\alpha)$ approaches $\lfloor n/2 \rfloor$.*

As another alternative to storing all the precise information we have about the contours, given some Π, we propose using outline contours. To do this, we require the subsidiary notion of a k-**partition**: For a set $\Lambda = \{\Phi_1, ..., \Phi_n\}$, where $k \leq n$, a k-partition is a partitioning of Λ into k disjoint subsets of Λ such that the partition with most members has at most one member more than the partition with fewest members. So a k-partition is as close as possible to a partitioning with partitions of equal size. Because a k-partition is a partitioning, each element of Λ is in exactly one partition. We also require the subsidiary notion of a **serial union**: For a set $\Lambda = \{\Phi_1, ..., \Phi_n\}$, where $k \leq n$, suppose $(\Lambda_1, ..., \Lambda_k)$ is a k-partitioning of Λ. Then the serial union of $(\Lambda_1, ..., \Lambda_k)$ is $(\cup(\Lambda_1), ..., \cup(\Lambda_k))$.

Definition 13. *For a set of answers Π, a formula α, and a threshold $k \in \mathbb{N}$, an* **outline contour** *$\mathsf{C}(\Pi, \alpha, k)$ is defined as follows: If $|\mathsf{C}(\Pi, \alpha)| < k$, then $\mathsf{C}(\Pi, \alpha, k)$ is $\mathsf{C}(\Pi, \alpha)$, otherwise let $\mathsf{C}(\Pi, \alpha, k)$ be the serial union of a k-partitioning of $\mathsf{C}(\Pi, \alpha)$.*

So if $\mathsf{C}(\Pi, \alpha, k)$ is $\mathsf{C}(\Pi, \alpha)$, then we are keeping the explicit information we have about the contour. But, if $|\mathsf{C}(\Pi, \alpha)| > k$, then we merge sets in $\mathsf{C}(\Pi, \alpha)$ so that we never have more than k sets. In any case, we can recover the contours from each outline contour.

Proposition 14. *If $\mathsf{C}(\Pi, \alpha) = \mathsf{C}(\alpha)$, and $k = 1$, then $\mathsf{C}(\Pi, \alpha, k) = \{\bigcup \mathsf{C}(\alpha)\}$.*

Example 10. *For $\langle \alpha, \neg\alpha, \beta \wedge \neg\beta, \neg\alpha \wedge \gamma, \alpha \vee \beta, \neg\beta \rangle$,*

$$\mathsf{C}(\bot) = \{110000, 001000, 100100, 010011\}$$

Let Π be such that $\mathsf{C}(\Pi, \bot) = \mathsf{C}(\bot)$ and let $k = 2$. So there are four possibilities for $\mathsf{C}(\Pi, \bot, k)$. One of them is $\{111000, 110111\}$.

Proposition 15. *Let Π be such that $\mathsf{C}(\Pi, \alpha) = \mathsf{C}(\alpha)$ For any k, if $\langle \Phi, \alpha \rangle$ is an argument, then there is a $\Psi \in \mathsf{C}(\Pi, \alpha, k)$, such that $\Phi \subseteq \Psi$.*

So outline contours involve more queries to the ATP than contours, but less space is required to store them.

8 DISCUSSION

Much progress has been made in developing formalisms for argumentation. Some algorithms for argumentation have been developed (for example [12, 6, 2, 11]). However, relatively little progress has been made in developing techniques for overcoming the computational challenges of constructing arguments, and in particular for intelligent searching for arguments.

In this paper, we have (1) clarified the need to manage the querying of an ATP when constructing arguments and counterarguments; (2) introduced contours as a way of representing information about a knowledgebase obtained by querying with an ATP; (3) shown how we can construct arguments and counterarguments using contours and shown how this is more efficient than naively using the knowledgebase with the ATP to search for arguments; (4) shown how we can contruct contours incrementally; and (5) considered a couple of simple strategies for offsetting the cost of maintaining larger contours. From the theoretical framework for contouring presented in this paper, it should be straightforward to develop algorithms for harnessing an ATP and undertaking empirical evaluation.

Whilst our presentation is based on a particular approach to logic-based argumentation, we believe the proposal could easily be adapted for a range of other logic-based approaches to argumentation (for example [11, 1, 9]).

REFERENCES

[1] L Amgoud and C Cayrol, 'A model of reasoning based on the production of acceptable arguments', *Annals of Mathematics and Artificial Intelligence*, **34**, 197–216, (2002).

[2] P Baroni and M Giacomin, 'Argumentation through a distributed self-stabilizing approach', *Journal of Experimental and Theoretical Artificial Intelligence*, **14**(4), 273–301, (2002).

[3] S Benferhat, D Dubois, and H Prade, 'Argumentative inference in uncertain and inconsistent knowledge bases', in *Proceedings of Uncertainty in Artificial Intelligence*, pp. 1449–1445. Morgan Kaufmann, (1993).

[4] Ph Besnard and A Hunter, 'A logic-based theory of deductive arguments', *Artificial Intelligence*, **128**, 203–235, (2001).

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

[6] C Cayrol, S Doutre, and J Mengin, 'Dialectical proof theories for the credulous preferred semantics of argumentation frameworks', in *Quantitative and Qualitative Approaches to Reasoning with Uncertainty*, volume 2143 of *LNCS*, pp. 668–679. Springer, (2001).

[7] C Chesnevar, A Maguitman, and R Loui, 'Logical models of argument', *ACM Computing Surveys*, **32**, 337–383, (2001).

[8] Y Dimopoulos, B Nebel, and F Toni, 'On the computational complexity of assumption-based argumentation for default reasoning', *Artificial Intelligence*, **141**, 57–78, (2002).

[9] P Dung, R Kowalski, and F Toni, 'Dialectic proof procedures for assumption-based, admissible argumentation', *Artificial Intelligence*, (2005). (in press).

[10] M Elvang-Gøransson, P Krause, and J Fox, 'Dialectic reasoning with classically inconsistent information', in *Proceedings of the Ninth Conference on Uncertainty in Artificial Intelligence*, pp. 114–121. Morgan Kaufmann, (1993).

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

[12] A Kakas and F Toni, 'Computing argumentation in logic programming', *Journal of Logic and Computation*, **9**, 515–562, (1999).

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

[14] H Prakken and G Vreeswijk, 'Logical systems for defeasible argumentation', in *Handbook of Philosophical Logic*, ed., D Gabbay, Kluwer, (2000).