Artificial Intelligence 173 (2009) 1406–1423

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Encoding deductive argumentation in quantified Boolean formulae

Philippe Besnard [a], Anthony Hunter [b,*], Stefan Woltran [c]

[a] IRIT-CNRS, Universitè Paul Sabatier, 118 rte de Narbonne, 31062 Toulouse, France
[b] Department of Computer Science, University College London, Gower Street, London, WC1E 6BT, UK
[c] Institute for Information Systems 184/2, Technische Universität Wien, Favoritenstrasse 9-11, 1040 Vienna, Austria

ARTICLE INFO

Article history:
Received 27 January 2009
Received in revised form 12 May 2009
Accepted 25 June 2009
Available online 27 June 2009

Keywords:
Argument systems
Argumentation
Classical logic
Inconsistency
Quantified Boolean formulae
Conflicting knowledge

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 knowledge-base such that Φ is consistent and Φ entails the claim α. Different logics provide different definitions for consistency and entailment and hence give us different options for argumentation. Classical propositional logic is an appealing option for argumentation but the computational viability of generating an argument is an issue. To better explore this issue, we use quantified Boolean formulae to characterise an approach to argumentation based on classical logic.

1. 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 is a range of proposals for logic-based formalisations of argumentation (for reviews see [8,13,31]). These proposals allow for the representation of arguments for and against some claim, and for counterargument relationships between arguments.

In a number of key proposals for argumentation, an argument is a pair where the first item in the pair is a consistent set (or a minimal consistent set) of formulae that proves the second item which is a formula (see for example [1,5,7,15, 24,26,30]). Hence, different underlying logics provide different definitions for consistency and entailment and hence give us different options for defining the notion of an argument.

Since classical logic has many advantages for representing and reasoning with knowledge including syntax, proof theory and semantics for the intuitive language incorporating negation, conjunction, disjunction and implication, it is an interesting and promising choice for the underlying logic for argumentation. However, it is computationally challenging to generate arguments from a knowledge-base using classical logic. 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 [23]. Furthermore, given a knowledge-base Δ and a formula α, it has been shown that ascertaining whether there is a subset Φ of Δ such that $\langle \Phi, \alpha \rangle$ is an argument (i.e. Φ is consistent, Φ entails α, and there is no subset of Φ that entails α) is a Σ_2^p-complete decision problem [29].

* Corresponding author.
E-mail address: A.Hunter@cs.ucl.ac.uk (A. Hunter).

0004-3702/$ – see front matter
doi:10.1016/j.artint.2009.06.006

Beyond these observations, there remains a range of further important computational complexity questions. So to better understand the use of classical logic in argumentation, and in particular to understand its computational properties, we use quantified Boolean formulae (QBFs) to characterise an approach to argumentation that is based on classical logic. This characterisation can then be used to obtain computational complexity results in terms of upper bounds.

A further reason to characterise logic-based argumentation in the form of QBFs is that we can then harness implementations of QBF solvers to develop prototype implementations for logic-based argumentation. There are numerous QBF solvers available (see, e.g, [28] and the references therein), and the encodations we present in this paper can be straightforwardly handled in them.

2. Preliminaries

2.1. Logical argumentation

In this section we review an existing proposal for logic-based argumentation [7]. We consider a classical propositional language. We use $\alpha, \beta, \gamma, \ldots$ to denote formulae and $\Delta, \Phi, \Psi, \ldots$ to denote sets of formulae. Deduction in classical propositional logic is denoted by the symbol $\vdash$ and deductive closure by Th so that $\mathsf{Th}(\Phi) = \{\alpha \mid \Phi \vdash \alpha\}$.

For the following definitions, we first assume a knowledge-base 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 not even 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** (or consequent) 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:

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

By monotonicity of classical logic the following equivalent characterisation easily follows.

Proposition 1. *A pair $\langle \Phi, \alpha \rangle$ is an argument iff it satisfies* (1)–(3) *from Definition* 1 *together with* (4′) *for each $\phi \in \Phi$, $(\Phi \setminus \{\phi\}) \nvdash \alpha$.*

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 $\langle \{\alpha, \alpha \to \beta\}, \beta \rangle$.

Definition 3. An argument $\langle \Phi, \alpha \rangle$ is **strictly more conservative** than an argument $\langle \Psi, \beta \rangle$ iff $\Phi \subseteq \Psi$, $\beta \vdash \alpha$, and either $\Psi \not\subseteq \Phi$ or $\alpha \nvdash \beta$.

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

Definition 4. An **undercut** for an argument $\langle \Phi, \alpha \rangle$ is an argument $\langle \Psi, \neg(\phi_1 \wedge \cdots \wedge \phi_n) \rangle$ 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 5. $\langle\Psi, \beta\rangle$ is a **maximally conservative undercut** for $\langle\Phi, \alpha\rangle$ iff $\langle\Psi, \beta\rangle$ is an undercut for $\langle\Phi, \alpha\rangle$ such that no undercuts of $\langle\Phi, \alpha\rangle$ are strictly more conservative than $\langle\Psi, \beta\rangle$.

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 6. An argument $\langle\Psi, \neg(\phi_1 \wedge \cdots \wedge \phi_n)\rangle$ 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.

The next result is central.

Proposition 2. *(See Theorem 5.4 [7].) A pair $\langle\Psi, \neg(\phi_1 \wedge \cdots \wedge \phi_n)\rangle$ is a canonical undercut for $\langle\Phi, \alpha\rangle$ iff it is an undercut for $\langle\Phi, \alpha\rangle$ and $\langle\phi_1, \ldots, \phi_n\rangle$ is the canonical enumeration of Φ.*

In other words, the canonical undercuts for $\langle\Phi, \alpha\rangle$ are given by all arguments of the form $\langle\Psi, \neg(\phi_1 \wedge \cdots \wedge \phi_n)\rangle$ where $\langle\phi_1, \ldots, \phi_n\rangle$ is the canonical enumeration of Φ. Later we need to refer to all possible supports of canonical undercuts for an argument. We thus introduce the following concept.

Definition 7. For $\langle\Phi, \alpha\rangle$, we define *UndercutSupports*$(\langle\Phi, \alpha\rangle)$ as the set of its supports:

$$\big\{\Psi \mid \langle\Psi, \beta\rangle \text{ is a canonical undercut for } \langle\Phi, \alpha\rangle\big\}.$$

We shall make use of the notation *UndercutSupports*$(\langle\Phi, \alpha\rangle)$ later when defining suitable representations of argument trees. Using Proposition 2, we can alternatively characterise the set *UndercutSupports*$(\langle\Phi, \alpha\rangle)$ as follows.

Proposition 3. *For $\langle\Phi, \alpha\rangle$, with $\langle\phi_1, \ldots, \phi_n\rangle$ the canonical enumeration of Φ,*

$$\mathit{UndercutSupports}\big(\langle\Phi, \alpha\rangle\big) = \big\{\Psi \mid \big\langle\Psi, \neg(\phi_1 \wedge \cdots \wedge \phi_n)\big\rangle \textit{ is an argument}\big\}.$$

Next we recall the notion of an argument tree following [7], and then introduce a more succinct notion to represent argument trees which is also more suitable for our later purposes.

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 8. An **annotated tree** is a tree where each node is a pair $\langle\Phi, \beta\rangle$. An **argument tree** for α is an annotated tree, such that

1. each node is an argument with the root being 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 some canonical undercuts for N that obey 2.

A **complete argument tree** is as just defined with "some" replaced by "all" in item 3 above.

The definition of an argument tree 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 (as illustrated in Example 5 below).

As a notational convenience, in examples of argument trees the $\diamond$ symbol is used to denote the consequent of an argument when that argument is a canonical undercut (no ambiguity arises as proven in [7]).

Example 4. Given $\Delta = \{\alpha, \alpha \to \beta, \gamma, \gamma \to \neg\alpha, \neg\gamma \vee \neg\alpha\}$, we have the following argument tree.

$$\begin{array}{ccc} & \big\langle\{\alpha, \alpha \to \beta\}, \beta\big\rangle & \\ \nearrow & & \nwarrow \\ \big\langle\{\gamma, \gamma \to \neg\alpha\}, \neg\big(\alpha \wedge (\alpha \to \beta)\big)\big\rangle & & \big\langle\{\gamma, \neg\gamma \vee \neg\alpha\}, \neg\big(\alpha \wedge (\alpha \to \beta)\big)\big\rangle \end{array}$$

Note the two undercuts are equivalent. They do count as two arguments because they are based on two different items of the database (even though these items turn out to be logically equivalent).

Example 5. Let $\Delta = \{\alpha, \alpha \to \beta, \gamma \to \neg\alpha, \gamma\}$.

$$\begin{array}{c} \langle\{\alpha, \alpha \to \beta\}, \beta\rangle \\ \uparrow \\ \langle\{\gamma, \gamma \to \neg\alpha\}, \diamond\rangle \\ \uparrow \\ \langle\{\alpha, \gamma \to \neg\alpha\}, \diamond\rangle \end{array}$$

This is not an argument tree because the undercut to the undercut is actually making exactly the same point (that α and γ are incompatible) as the undercut itself does, just by using modus tollens instead of modus ponens.

Example 6. Given $\Delta = \{\alpha, \beta, \alpha \to \gamma, \beta \to \delta, \neg\alpha \vee \neg\beta\}$, consider the following tree.

$$\begin{array}{ccc} & \langle\{\alpha, \beta, \alpha \to \gamma, \beta \to \delta\}, \gamma \wedge \delta\rangle & \\ \nearrow & & \nwarrow \\ \langle\{\alpha, \neg\alpha \vee \neg\beta\}, \neg\beta\rangle & & \langle\{\beta, \neg\alpha \vee \neg\beta\}, \neg\alpha\rangle \end{array}$$

This is not an argument tree because the two children nodes are not maximally conservative undercuts. The first undercut is essentially the same argument as the second undercut in a rearranged form (relying on α and β being incompatible, assume one and then conclude that the other doesn't hold). If we replace these by the maximally conservative undercut $\langle\{\neg\alpha \vee \neg\beta\}, \diamond\rangle$, we obtain an argument tree.

Notably, there is a finite number of argument trees with the root being an argument with the claim α that can be formed from Δ, and each of these trees has finite branching and a finite depth (the finite tree property).

For our purposes in this paper, we require a more formal representation of argument trees. It makes use of the fact that all consequences in the nodes (except the root) of an argument tree are determined by their direct ancestor (as already mentioned above when introducing $\diamond$). To this end, a node is now a set of formulae rather than an argument, and a parent function determines the structure of the tree.

Definition 9. A **parent function** p (over $k \geqslant 1$) is a partial function from $\{1..k\}$ to $\{1..k\}$, such that $p(j)$ is undefined for $j = 1$ but $p(j)$ is defined and $p(j) < j$, for any $1 < j \leqslant k$.

p is a parent function **for a sequence** $\mathcal{A} = \langle\Phi_1, \ldots, \Phi_k\rangle$ of subsets of Δ if p is a parent function over k and is such that $p(i) = p(j)$ implies $\Phi_i \neq \Phi_j$, for any $1 < j < i \leqslant k$.

A **tuple form** is a triple $\langle\alpha, \mathcal{A}, p\rangle$, where α is a formula, $\mathcal{A}$ is a sequence of subsets of Δ, and p is a parent function for $\mathcal{A}$.

Given a tuple form $\langle\alpha, \langle\Phi_1, \ldots, \Phi_k\rangle, p\rangle$, we define, for each $1 \leqslant i \leqslant k$, an **associated pair**, $A(i)$, as follows $A(1) = \langle\Phi_1, \alpha\rangle$ and, for $i > 1$, $A(i) = \langle\Phi_i, \neg(\phi_1 \wedge \cdots \wedge \phi_n)\rangle$, where $\langle\phi_1, \ldots, \phi_n\rangle$ is the canonical enumeration of $\Phi_{p(i)}$.

Tuple forms are an alternative way to denote annotated trees. Intuitively, $\mathcal{A}$ collects all supports of the tree's nodes, α is the claim of the root node, and the parent function p links each node to its parent node, and thus determines the structure of the tree. This is feasible, since p is defined for each node except the root and links to a previous element in $\mathcal{A}$; the condition that $p(i) = p(j)$ implies $\Phi_i \neq \Phi_j$, for $i \neq j$ just avoids duplicate children nodes.

The concept of tuple forms is best illustrated by examples.

Example 7. First, consider the tree from Example 4. That annotated tree can be represented in tuple form $\langle\beta, \langle\Phi_1, \Phi_2, \Phi_3\rangle, p\rangle$ where $\Phi_1 = \{\alpha, \alpha \to \beta\}$, $\Phi_2 = \{\gamma, \gamma \to \neg\alpha\}$, $\Phi_3 = \{\gamma, \neg\gamma \vee \neg\alpha\}$, and p is defined as $p(2) = p(3) = 1$. An alternative way to represent the same annotated tree would be to exchange the sets for Φ_2 and Φ_3.

Conversely, given the tuple form $\langle\beta, \langle\Phi_1, \Phi_2, \Phi_3\rangle, p\rangle$, we can derive from it an annotated tree as follows: The nodes are given by $A(1)$, $A(2)$, $A(3)$, and we get by definition of p that $A(2)$ and $A(3)$ are the children of the root node $A(1)$.

Example 8. As a second example, consider Example 5. The only way to achieve a tuple form for that tree is $\langle\beta, \langle\Phi_1, \Phi_2, \Phi_3\rangle, p\rangle$ where $\Phi_1 = \{\alpha, \alpha \to \beta\}$, $\Phi_2 = \{\gamma, \gamma \to \neg\alpha\}$, $\Phi_3 = \{\alpha, \gamma \to \neg\alpha\}$, and p is defined as $p(2) = 1$, $p(3) = 2$.

We now formally describe these relations.

Definition 10. We define a mapping *TreeForm* from tuple forms to graphs as follows: For each $t = \langle\alpha, \langle\Phi_1, \ldots, \Phi_k\rangle, p\rangle$, the nodes of *TreeForm*(t) are given by the set $\{A(i) \mid 1 \leqslant i \leqslant k\}$; and a pair $(A(i), A(j))$ is an edge of *TreeForm*(t) iff $p(j) = i$, for $1 < j \leqslant k$.

Lemma 1. *For any tuple form t, TreeForm(t) is an annotated tree.*

Proof. *TreeForm*(t) is a tree because it is a graph which is connected (ignoring direction of edges) and has exactly one edge less than it has vertices:

By Definition 10, $(A(i), A(j))$ is an edge iff $p(j) = i$ (for $1 < j \leqslant k$). I.e., $(A(p(j)), A(j))$ for $j = 1..k$ exhausts all edges. Since p is a parent function, *TreeForm*(t) has $k - 1$ edges. There remains to show that *TreeForm*(t) is connected (when directions of edges are ignored). This easily follows from the fact that any node in *TreeForm*(t) is connected to $A(1)$ (the latter is true because if $1 < j \leqslant k$, then there exists n such that $p^n(j) = 1$ as p is a parent function). In short, *TreeForm*(t) is a tree. It is an annotated tree because Definition 10 trivially shows that all nodes in *TreeForm*(t) are pairs $A(i)$ for $i = 1..k$. □

In view of the above lemma, we call, for a given tuple form t, *TreeForm*(t) the **tree associated to** t. As well, we say that t **represents tree** *TreeForm*(t).

We now characterise argument trees and complete argument trees via tuple forms. This result is valuable later when characterising argument trees via QBFs. We need one more technical notation.

Definition 11. Given a tuple form $\langle \alpha, \langle \Phi_1, \ldots, \Phi_k \rangle, p \rangle$, we define, for each $1 \leqslant i \leqslant k$, $p^*(i)$ as the set of indices of Φ_i's ancestors, i.e.,

$$p^*(i) = \left\{ p^n(i) \mid \text{ there exists } m \geqslant n \geqslant 1 \text{ such that } p^m(i) = 1 \right\}.$$

Lemma 2. *A tuple form $\langle \alpha, \langle \Phi_1, \ldots, \Phi_k \rangle, p \rangle$ represents*

- *an argument tree iff*
  (1) $\langle \Phi_1, \alpha \rangle$ *is an argument,*
  (2) *for each* $1 < i \leqslant k$, $\Phi_i \not\subseteq \bigcup_{j \in p^*(i)} \Phi_j$, *and*
  (3) *for each* $1 < i \leqslant k$, $\Phi_i \in$ *UndercutSupports*$(A(p(i)))$ *hold*;
- *a complete argument tree iff,* (1)–(3) *hold together with*
  (4) *for each* $1 \leqslant i \leqslant k$ *and for each* $\Psi \in$ *UndercutSupports*$(A(i))$, *there exists an index* $j \in \{1..k\}$, *such that* $\Phi_j = \Psi$ *and* $p(j) = i$.

Proof. The first statement in Lemma 2 means that *TreeForm*(t) is an argument tree iff (1)–(3) hold together. Let us first assume that *TreeForm*(t) is an argument tree. Then, (1) and (2) are easily verified. By item 3 in Definition 8, the children of a node N are canonical undercuts for N. So, if N is $\langle \Phi, \beta \rangle$, any child of N is a canonical undercut $A(i) = \langle \Phi_i, \neg(\phi_1 \wedge \cdots \wedge \phi_n) \rangle$ of $N = A(p(i))$. Then, Definition 7 directly yields $\Phi_i \in$ *UndercutSupports*$(A(p(i)))$. That is, (3) holds as well.

As to the other direction, let us assume that (1)–(3) hold. By (3) and item 4 in Definition 9, Definition 7 means that $A(i)$ is an argument for $1 < i \leqslant k$. Due to (1), it follows that item 1 in Definition 8 is verified. It is easy to verify that (2) implies item 2 in Definition 8. Lastly, (3) and item 4 in Definition 9 entail (cf Definition 7) that for $1 < i \leqslant k$, each $A(i)$ is a canonical undercut of $A(p(i))$. I.e., item 3 in Definition 8 holds.

Let us assume (1)–(4). Let us further assume that *TreeForm*(t) is not a complete argument tree. In view of Definition 8, this can only happen due to a node $N = A(p(i))$ lacking at least one canonical undercut as a child (*TreeForm*(t) is an argument tree, as proved above). By Definition 7, there then exists Ψ in *UndercutSupports*$(A(j))$, for some j, satisfying $\Psi = \Phi_i$ for no i such that $p(i) = j$. This contradicts (4). So, the if direction is proved. Proof of the only if direction is easy and is omitted. □

Example 9. Consider again the tuple form for the tree in Example 5, as given in Example 8. We have $p^*(3) = \{1, 2\}$ and thus $\bigcup_{j \in p^*(3)} \Phi_j = \Phi_1 \cup \Phi_2 = \{\alpha, \gamma, \gamma \rightarrow \neg\alpha\}$. Since $\Phi_3 = \{\alpha, \gamma \rightarrow \neg\alpha\}$ is a subset of that set, condition (2) in Lemma 2 is violated. Thus, we have that the tuple form does not represent an argument tree.

Lemma 3. *Each argument tree is represented by a tuple form.*

Proof. Consider an argument tree T with nodes $N_1, \ldots, N_k$, where nodes are of the form $N_i = (\Phi_i, \alpha_i)$, for each $1 \leqslant i \leqslant k$, and N_1 is the root of T. Consider $t_T = \langle \alpha_1, \langle \Phi_1, \ldots, \Phi_k \rangle, p \rangle$ where p is a partial function $\{1..k\}$ to $\{1..k\}$ satisfying, for each $1 \leqslant j < k$, $p(j) = i$ iff N_j is a children node of N_i in T. Since T is an argument tree, p is in fact a parent function over k. Thus t_T is a tuple form and one can show that *TreeForm*$(t_T) = T$, which holds by the observation that the pairs associated to t_T satisfy $A(i) = N_i$, for each $1 \leqslant i \leqslant k$. □

2.2. Quantified Boolean formulae

Quantified Boolean formulae (QBFs) generalize ordinary propositional formulae by the admission of quantifications over propositional variables. In particular, the language of QBFs contains, for any atom p, unary operators of the form $\forall p$ and $\exists p$, called *universal* and *existential quantifiers*, respectively. However, the quantifiers do not range over some arbitrary domain,

but over truth assignments. Thus, a QBF of form $\forall p \exists q\, F$ is satisfiable iff, for all truth assignments of p, there is a truth assignment of q such that F is satisfiable; see also Example 10 below.

An occurrence of a propositional variable p in a QBF F is *free* iff it does not appear in the scope of a quantifier $\mathsf{Q}p$ ($\mathsf{Q} \in \{\forall, \exists\}$), otherwise the occurrence of p is *bound*. If F contains no free variable occurrences, then F is *closed*, otherwise F is *open*. Furthermore, we write $F[p/\phi]$ to denote the result of uniformly substituting each free occurrence of the variable p in F by a ϕ. For a set $P = \{p_1, \ldots, p_n\}$ of propositional variables, we let $\forall P\, F$ stand for the formula $\forall p_1 \forall p_2 \cdots \forall p_n$, and $\exists P\, F$ for the formula $\exists p_1 \exists p_2 \cdots \exists p_n$.

By an *interpretation*, I, we mean a set of atoms. Informally, an atom p is true under I iff $p \in I$. In general, the truth value, $\nu_I(F)$, of a QBF F under an interpretation I is recursively defined as follows:

1. if $F = \top$, then $\nu_I(F) = 1$;
2. if $F = p$ is an atom, then $\nu_I(F) = 1$ if $p \in I$, and $\nu_I(F) = 0$ otherwise;
3. if $F = \neg G$, then $\nu_I(F) = 1 - \nu_I(G)$;
4. if $F = (F_1 \wedge F_2)$, then $\nu_I(F) = \min(\{\nu_I(F_1), \nu_I(F_2)\})$;
5. if $F = \forall p\, G$, then $\nu_I(F) = \nu_I(G[p/\top] \wedge G[p/\bot])$;
6. if $F = \exists p\, G$, then $\nu_I(F) = \nu_I(G[p/\top] \vee G[p/\bot])$.

The truth conditions for $\bot$, $\vee$, $\rightarrow$, and $\leftrightarrow$ follow from the above in the usual way. We say that F is *true under* I iff $\nu_I(F) = 1$, otherwise F is *false under* I. If $\nu_I(F) = 1$, then I is a *model* of F. If F has some model, then F is said to be *satisfiable*. If F is true under any interpretation, then F is *valid*. Observe that a closed QBF is either valid or unsatisfiable, because closed QBFs are either true under each interpretation or false under each interpretation. Hence, for closed QBFs, there is no need to refer to particular interpretations. Therefore, closed QBFs are simply either true or false. Two QBFs are *logically equivalent* iff they possess the same models.

Example 10. Consider the QBF $F_1 = \forall q(p \leftrightarrow q)$. In this QBF, the propositional variable p if free, while q is bound. To evaluate the QBF, we thus consider two interpretations: $I_1 = \emptyset$ setting p to false; and $I_2 = \{p\}$ setting p to true.

In general, given an interpretation I, we can evaluate a QBF with respect to I in two ways: (i) first evaluate the free variables according to I and then apply the semantics for the now closed QBF; (ii) first apply the semantics for quantifiers and then evaluate the now quantifier-free formula using I.

So, in our example (i) is as follows: For I_1, we get $\forall q(\bot \leftrightarrow q)$, i.e., $\forall q(\neg q)$; and for I_2, we get $\forall q(\top \leftrightarrow q)$, i.e., $\forall q(q)$. Both closed QBFs are false, thus neither I_1 nor I_2 is a model of F_1. Following attempt (ii), we first treat the universal quantification for q according to the semantics and get $(p \leftrightarrow \top) \wedge (p \leftrightarrow \bot)$ which is equivalent to $p \wedge \neg p$. Clearly, neither I_1 nor I_2 is a model of this propositional formula. Hence, neither I_1 nor I_2 is a model of F_1. Observe that we thus can also state that the closed QBF

$$\exists p \forall q(p \leftrightarrow q)$$

is false.

Now consider the QBF $F_2 = \exists q(p \leftrightarrow q)$. As before, interpretations I_1 and I_2 are of interest. According to (ii), F_2 reduces to $(p \leftrightarrow \top) \vee (p \leftrightarrow \bot)$ which is equivalent to $p \vee \neg p$. Now both, I_1 and I_2 are models of that formula and thus of F_2. This leads us to the further observation that the closed QBF

$$\forall p \exists q(p \leftrightarrow q)$$

is true.

QBFs allow us to talk about semantical concepts in propositional logic. For instance, a propositional formula F over propositional variables V is satisfiable iff the closed QBF $\exists V(F)$ is true. Likewise, F is valid iff the closed QBF $\forall V(F)$ is true. Consequently, given a knowledge-base Δ and a formula, both over V, $\Delta \vdash \alpha$ holds iff the QBF $\forall V(\bigwedge_{\delta \in \Delta} \delta \rightarrow \alpha)$ is true.

Example 11. Consider $\Delta = \{p, p \rightarrow q\}$ and let $\alpha = q$. We have $V = \{p, q\}$ and thus consider the closed QBF

$$\forall p \forall q\big(\big(p \wedge (p \rightarrow q)\big) \rightarrow q\big).$$

Observe that the inner part of that QBF, i.e., the propositional formula $(p \wedge (p \rightarrow q)) \rightarrow q$ is valid, and thus true under all assignments. Hence, the above QBF is true.

In the same way as the satisfiability problem of classical propositional logic is the "prototypical" problem of NP, i.e., being an NP-complete problem, the satisfiability problem of QBFs in *prenex form* are the "prototypical" problems of the k-th level of the polynomial hierarchy.

Proposition 4. *(See [35].) Given a propositional formula ϕ with its atoms partitioned into $i \geqslant 1$ sets $P_1, \ldots, P_i$, deciding whether $\mathsf{Q}_1 P_1 \mathsf{Q}_2 P_2 \ldots \mathsf{Q}_i P_i \phi$ is true is* (i) *Σ_i^p-complete, if $\mathsf{Q}_1 = \exists$*; (ii) *Π_i^p-complete, if $\mathsf{Q}_1 = \forall$.*

In fact, the hardness results in above proposition hold only for those QBFs where the quantifiers in the *prefix* $\mathsf{Q}_1 P_1 \mathsf{Q}_2 P_2 \ldots \mathsf{Q}_i P_i$ are *alternating*, i.e., $\mathsf{Q}_j \neq \mathsf{Q}_{j+1}$ holds, for each $1 \leqslant j < i$. We call such QBFs also (Q_1, i)-QBFs.

The complexity landscape can be extended to arbitrary closed QBFs if the maximal number of quantifier alternations along a path in the QBF's formula tree is taken into account. In turn, an arbitrary QBF can be transformed into an equivalent QBF in prenex form. This transformation is not deterministic and it is crucial for the performance of QBF solvers requiring the input formula in this normal form (for details, see [20,21]).

Finally, we highlight the used reduction approach. Given a decision problem D, we aim at finding a translation scheme $\mathcal{T}_D$ into closed QBFs, such that

1. $\mathcal{T}_D(\cdot)$ is faithful, i.e., $\mathcal{T}_D(K)$ is true iff K is a yes-instance of D;
2. for each instance K, $\mathcal{T}_D(K)$ is computable in polynomial time with respect to the size of K; and
3. determining the truth of the QBFs resulting from $\mathcal{T}_D(\cdot)$ is not computationally harder (by means of Proposition 4) than the computational complexity of D.

In addition, if we are interested in a search problem S we aim at establishing a certain one-to-one correspondence between the models of the QBF encodings and the solutions to S. Indeed the transformation $\mathcal{T}_D(\cdot)$ then has to yield open QBFs instead of closed QBFs. Given the models of the QBF $\mathcal{T}_D(K)$, the computation of the solutions of K has to be feasible in polynomial time.

2.3. Basic concept of encodings

We now sketch our basic ideas for capturing logic-based argumentation in QBFs. In the following, we assume a knowledge-base Δ to be given over a set of atoms V_Δ. Moreover, α, β always refer to formulae, which are, without loss of generalization, assumed to be given over atoms from V_Δ. In general, for a set Φ of formulae, the set V_Φ contains all atoms occurring in Φ.

Given a finite knowledge-base Δ, we assign to each element of Δ several new atoms via a generator function. The aim of this function to provide new atoms, such that interpretations over those atoms are used to represent subsets of Δ. The formal definition is as follows:

Definition 12. A **generator function** g maps each $\delta \in \Delta$ to a new propositional atom $g(\delta) \notin V_\Delta$, such that $g(\delta_1) = g(\delta_2)$ implies $\delta_1 = \delta_2$, for all $\delta_1, \delta_2 \in \Delta$. With some abuse of notation we write, for any subset $\Phi \subseteq \Delta$, $g(\Phi)$ to denote the set $\{g(\delta) \mid \delta \in \Phi\}$. Moreover, for two different generator functions g_1, g_2, we ensure $g_1(\Delta) \cap g_2(\Delta) = \emptyset$, i.e., each generator function provides its own fresh atoms.

Interpretations (usually given over arbitrary atoms) are linked to subsets of Δ via generator functions as follows.

Definition 13. Let I be an interpretation, g be a generator function, and $\Phi \subseteq \Delta$. We say that I **represents** Φ via g iff $I \cap g(\Delta) = g(\Phi)$. Moreover, for a sequence $\mathcal{A} = \langle \Phi_1, \ldots, \Phi_k \rangle$ of subsets of Δ and a corresponding sequence $\mathcal{G} = \langle g_1, \ldots, g_k \rangle$ of different generator functions, we say that an interpretation I **represents** $\mathcal{A}$ via $\mathcal{G}$ iff $I \cap g_i(\Delta) = g_i(\Phi_i)$, holds for all $1 \leqslant i \leqslant k$.

A word of caution is in order here: When I represents Φ via some g, I may, but need not, be a model of Φ. The forthcoming Definition 14 and Lemma 4 provide the missing link.

Definition 14. For $\Gamma \subseteq \Delta$, a formula α, and a generator function g, define

$$\Gamma \Rightarrow_g \alpha = \forall V_{\Gamma \cup \{\alpha\}} \left(\left(\bigwedge_{\delta \in \Gamma} \big(g(\delta) \rightarrow \delta\big) \right) \rightarrow \alpha \right).$$

Lemma 4. *Let Δ be a knowledge base, and I be an interpretation. For $\Gamma \subseteq \Delta$, and $\Phi \subseteq \Gamma$, such that I represents Φ via generator function g, then, for all α, we have that, $\Gamma \Rightarrow_g \alpha$ is true under I iff $\Phi \vdash \alpha$.*

Proof. We have that $\Phi \vdash \alpha$ iff each model over atoms $V_{\Phi \cup \{\alpha\}}$ of Φ is also a model of α. By the semantics of QBFs, it is easily verified that the latter holds iff the closed QBF

$$\forall V_{\Phi \cup \{\alpha\}} \left(\left(\bigwedge_{\delta \in \Phi} \delta \right) \rightarrow \alpha \right) \tag{1}$$

is true. (Recall that a closed QBF is either true under any interpretation I or false under any interpretation I.) We next increase the set of quantified variables in (1) from $V_{\Phi\cup\{\alpha\}}$ to $V_{\Gamma\cup\{\alpha\}}$, which yields

$$\forall V_{\Gamma\cup\{\alpha\}}\Big(\Big(\bigwedge_{\delta\in\Phi}\delta\Big)\to\alpha\Big). \tag{2}$$

This QBF is also closed since $\Phi\subseteq\Gamma$ and it holds that (2) is true iff (1) is true, since the added quantified variables do not have any influence here.

Next, we replace each δ by the equivalent formula $\top\to\delta$, and add trivially true conjuncts of the form $\bot\to\gamma$, yielding[1]

$$\forall V_{\Gamma\cup\{\alpha\}}\Big[\Big(\Big(\bigwedge_{\delta\in\Phi}(\top\to\delta)\Big)\wedge\Big(\bigwedge_{\gamma\in\Gamma\setminus\Phi}(\bot\to\gamma)\Big)\Big)\to\alpha\Big]. \tag{3}$$

So far, this shows that $\Phi\vdash\alpha$ iff the closed QBF (3) is true. Now, let g be a generator function, and consider any interpretation I which represents Φ via g. Hence, for each $\delta\in\Phi$, $g(\delta)\in I$, and for each $\gamma\in\Gamma\setminus\Phi$, $g(\gamma)\notin I$ holds. Recall that $g(\Delta)\cap V_\Delta=\emptyset$, and thus by our assumptions $g(\Delta)\cap V_{\Gamma\cup\{\alpha\}}=\emptyset$. We therefore can rewrite (3) to

$$\forall V_{\Gamma\cup\{\alpha\}}\Big[\Big(\Big(\bigwedge_{\delta\in\Phi}\big(g(\delta)\to\delta\big)\Big)\wedge\Big(\bigwedge_{\gamma\in\Gamma\setminus\Phi}\big(g(\gamma)\to\gamma\big)\Big)\Big)\to\alpha\Big]. \tag{4}$$

Observe that the atoms $g(\cdot)$ are free in (4) and thus are subject to interpretations. In fact, by the definition of a representation (cf. Definition 13), it is easy to see that (4) is true under any I which represents Φ via g iff (3) is true. To conclude the proof, observe (4) is equivalent to $\Gamma\Rightarrow_g\alpha$. □

Example 12. Let $\Delta=\{p,p\to q\}$, $\alpha=q$, and let us consider $g(\Delta)=\{g_p,g_{p\to q}\}$. Hence, the generator function provides for each $\delta\in\Delta$ a new variable of the form $g(\delta)=g_\delta$. Then, $\Delta\Rightarrow_g\alpha$ is given by

$$\forall p\forall q\big[\big((g_p\to p)\wedge\big(g_{p\to q}\to(p\to q)\big)\big)\to q\big]. \tag{5}$$

Note that, for each $\Phi\subseteq\Delta$, we thus have interpretations representing Φ via g. Since $g(\Delta)$ are the only free variables in $\Delta\Rightarrow_g\alpha$ it is thus sufficient to investigate the following four interpretations for being models of $\Delta\Rightarrow_g\alpha$:

$$I_1=\emptyset$$
$$I_2=\{g_p\}$$
$$I_3=\{g_{p\to q}\}$$
$$I_4=\{g_p,\ g_{p\to q}\}.$$

Let us now evaluate (5) under these four interpretation. We shall do so by first evaluating the free variables in (5) and then inspect the remaining QBF, i.e., following method (ii) as sketched in Example 10. We start with I_1. Then (5) reduces to closed QBF

$$\forall p\forall q\big[\big((\bot\to p)\wedge\big(\bot\to(p\to q)\big)\big)\to q\big]$$

which is equivalent to

$$\forall p\forall q[q].$$

This QBF is obviously false, and hence, I_1 is not a model of (5).

For I_2 one of the conjuncts in the antecedent survives. We get

$$\forall p\forall q\big[\big((\top\to p)\wedge\big(\bot\to(p\to q)\big)\big)\to q\big]$$

which is equivalent to

$$\forall p\forall q[p\to q].$$

Still, this QBF is false, and hence, also I_2 is not a model of (5).

For I_3, we get

$$\forall p\forall q\big[\big((\bot\to p)\wedge\big(\top\to(p\to q)\big)\big)\to q\big]$$

which is equivalent to

$$\forall p\forall q\big[(p\to q)\to q\big].$$

[1] This can be done since the replacement theorem holds for QBFs.

Again, this QBF is false, and hence, also I_3 is not a model of (5).

Finally, evaluating (5) under I_4 yields

$$\forall p \forall q \big[\big((\top \to p) \wedge \big(\top \to (p \to q)\big)\big) \to q\big]$$

which is equivalent to

$$\forall p \forall q \big[\big(p \wedge (p \to q)\big) \to q\big].$$

This QBF is true since the inner part $(p \wedge (p \to q)) \to q$ is indeed a valid formula of propositional logic. Therefore, I_4 is a model of (5).

So having I_4 as the only model, we conclude that the set it represents via g, namely $\{p,\ p \to q\}$, is the only subset Φ of Δ, for which $\Phi \vdash \alpha$ holds.

QBFs abbreviated by $\Gamma \Rightarrow_g \alpha$ will be used as subformulae in various more complex QBF formulae. In a sense, they are useful building blocks that can be used repeatedly. We will refer to a schema like $\Gamma \Rightarrow_g \alpha$ as a module.

3. Characterisations

In what follows, we will employ the basic encoding $\Gamma \Rightarrow_g \alpha$ to characterise various problems for logic-based argumentation. We start by characterising arguments and undercuts via models of certain QBFs. Then, we suitably combine the latter in such a way that the resulting formulae will allow us to reason about argument trees. We will first consider argument trees of a fixed structure (i.e., where the parent function is given when constructing the encodings) and then also provide encodings, where the parent function is characterised by the QBF itself.

3.1. Arguments and undercuts

Definition 15. For a knowledge base Δ, a formula α, and a generator function g, define

$$arg(g, \Delta, \alpha) = \neg(\Delta \Rightarrow_g \bot) \wedge (\Delta \Rightarrow_g \alpha) \wedge \bigwedge_{\delta \in \Delta} \big(g(\delta) \to \neg\big(\big(\Delta \setminus \{\delta\}\big) \Rightarrow_g \alpha\big)\big).$$

Note that the three main parts of the encoding check properties (2), (3), and respectively (4′) from Proposition 1.

Theorem 1. *For a knowledge base Δ, a formula α, an interpretation I, and $\Phi \subseteq \Delta$, such that I represents Φ via generator function g, we have that $arg(g, \Delta, \alpha)$ is true under I iff (Φ, α) is an argument.*

Proof. Using Lemma 4, we immediately conclude that the first two conjuncts of $arg(g, \Delta, \alpha)$ are true in I iff conditions (2) and (3) from Proposition 1 hold. So, there only remains to take care of the third conjunct in $arg(g, \Delta, \alpha)$, i.e.:

$$\bigwedge_{\delta \in \Delta} \big(g(\delta) \to \neg\big(\big(\Delta \setminus \{\delta\}\big) \Rightarrow_g \alpha\big)\big).$$

However, all of the following five statements are equivalent:

(i) $I \models \bigwedge_{\delta \in \Delta} \big(g(\delta) \to \neg\big(\big(\Delta \setminus \{\delta\}\big) \Rightarrow_g \alpha\big)\big)$

(ii) $I \models \bigwedge_{\delta \in \Phi} \neg\big(\big(\Delta \setminus \{\delta\}\big) \Rightarrow_g \alpha\big)$

(iii) $I \models \neg\big(\big(\Delta \setminus \{\delta\}\big) \Rightarrow_g \alpha\big)$ for all $\delta \in \Phi$

(iv) $I \models \neg \forall V_{(\Delta \setminus \{\delta\}) \cup \{\alpha\}} \Big(\Big(\bigwedge_{\sigma \in (\Delta \setminus \{\delta\})} \big(g(\sigma) \to \sigma\big)\Big) \to \alpha\Big)$ for all $\delta \in \Phi$

(v) $I \models \neg \forall V_{(\Delta \setminus \{\delta\}) \cup \{\alpha\}} \Big(\bigwedge\big(\Phi \setminus \{\delta\}\big) \to \alpha\Big)$ for all $\delta \in \Phi$

where the first and last steps are correct because I represents Φ via g.

Since $(\Phi \setminus \{\delta\}) \subseteq (\Delta \setminus \{\delta\})$, all propositional symbols in $\bigwedge(\Phi \setminus \{\delta\}) \to \alpha$ are quantified upon through $\forall V_{(\Delta \setminus \{\delta\}) \cup \{\alpha\}}$. Hence, (v) holds iff $\bigwedge(\Phi \setminus \{\delta\}) \to \alpha$ is invalid in propositional logic, or, equivalently, iff $\Phi \setminus \{\delta\} \nvdash \alpha$. In other words, condition (4′) from Proposition 1 is satisfied iff (v) holds, i.e., iff (i) holds. □

We now consider the following example in order to compare the functioning of the third condition of Definition 15 with a simpler alternative that, whilst plausible, does not behave as required. In fact, consider one replaces

$$\bigwedge_{\delta\in\Delta}\big(g(\delta)\to\neg\big((\Delta\setminus\{\delta\})\Rightarrow_g\alpha\big)\big)\quad\text{in } arg(g,\Delta,\alpha) \text{ by }\quad \neg\big((\Delta\setminus\{\delta\})\Rightarrow_g\alpha\big).$$

We observe the following problem.

Example 13. Let $\Delta=\{p,q\}$, let α be p, and let $g(\Delta)=\{g_p,g_q\}$. So the original version of the third condition in Definition 15 gives the following

$$\big(g_p\to\neg\forall p\forall q\big((g_q\to q)\to p\big)\big)\wedge\big(g_q\to\neg\forall p\forall q\big((g_p\to p)\to p\big)\big).$$

We evaluate this with the following interpretations getting the answers we expect for the QBF in each case.

$I_1=\emptyset$	therefore the QBF is true
$I_2=\{g_p\}$	therefore the QBF is true
$I_3=\{g_q\}$	therefore the QBF is true
$I_4=\{g_p,g_q\}$	therefore the QBF is false.

Now consider the alternative (incorrect) version of the definition which gives the following

$$\big(\neg\forall p\forall q\big((g_q\to q)\to p\big)\big)\wedge\big(\neg\forall p\forall q\big((g_p\to p)\to p\big)\big).$$

We evaluate this with the following interpretations which shows that we fail to get answer we expect for the QBF with I_2.

$I_1=\emptyset$	therefore the QBF is true
$I_2=\{g_p\}$	therefore the QBF is false
$I_3=\{g_q\}$	therefore the QBF is true
$I_4=\{g_p,g_q\}$	therefore the QBF is false.

With the encoding from Definition 15 at hand, we can decide a number of typical decision problems, e.g., question whether given $\langle\Phi,\alpha\rangle$, is $\langle\Phi,\alpha\rangle$ an argument? A more general variant of this question is as follows: Given Δ and disjoint subsets Δ^+ and Δ^- of Δ, does there exist an argument $\langle\Phi,\alpha\rangle$, such that $\Delta^+\subseteq\Phi$ and $\Phi\cap\Delta^-=\emptyset$?

Definition 16. Let g be a generator function and $\Delta^+,\Delta^-\subseteq\Delta$. Then, we define as an abbreviation

$$fix(g,\Delta^+,\Delta^-)=\bigwedge_{\delta\in\Delta^+}g(\delta)\wedge\bigwedge_{\delta\in\Delta^-}\neg g(\delta).$$

Corollary 1. *Given Δ, two disjoint sets Δ^+ and Δ^-, a generator function g, and a formula α, there exists an argument $\langle\Phi,\alpha\rangle$ such that $\Delta^+\subseteq\Phi$ and $\Phi\cap\Delta^-=\emptyset$ iff*

$$\exists g(\Delta)\big(fix(g,\Delta^+,\Delta^-)\wedge arg(g,\Delta,\alpha)\big)\tag{6}$$

is true.

Obviously, by setting $\Delta^+=\Phi$ and $\Delta^-=\Delta\setminus\Phi$ in (6), we can answer the question given $\langle\Phi,\alpha\rangle$, is $\langle\Phi,\alpha\rangle$ an argument? In this setting, we shall also write $fix(g,\Phi)$ instead of $fix(g,\Delta^+,\Delta^-)$. Another question is whether a certain element $\delta\in\Delta$ is part of a support for α. For this, we can set $\Delta^+=\{\delta\}$ and $\Delta^-=\emptyset$ in (6). Finally, if we drop the $fix(g,\Delta^+,\Delta^-)$ conjunct, i.e., we set $\Delta^+=\Delta^-=\emptyset$ in (6), then our encoding is true iff there is a subset Φ of Δ such that $\langle\Phi,\alpha\rangle$ is an argument.

Next, we show how to use two different generator functions g_1 and g_2 to characterise subsets of Δ simultaneously; in fact, this module allows us to derive the supports of undercuts.

Definition 17. For a knowledge base Δ and generator functions g_1, g_2, define

$$suc(g_1,g_2,\Delta)=arg\Big(g_1,\Delta,\neg\bigwedge_{\delta\in\Delta}\big(g_2(\delta)\to\delta\big)\Big).$$

Theorem 2. *For a knowledge base Δ, an interpretation I, and $\Phi_1,\Phi_2\subseteq\Delta$, such that I represents $\langle\Phi_1,\Phi_2\rangle$ via generator functions $\langle g_1,g_2\rangle$, we have that $suc(g_1,g_2,\Delta)$ is true under I iff $(\Phi_1,\neg(\phi_1\wedge\cdots\wedge\phi_n)$ is an argument, where $\langle\phi_1,\ldots,\phi_n\rangle$ is the canonical enumeration of Φ_2.*

Proof. Since I represents Φ_2 via g_2, it follows that, in I, $g_2(\delta) \to \delta$ is equivalent with δ whenever $\delta \in \Phi_2$ and is equivalent with $\top$ whenever $\delta \in \Delta \setminus \Phi_2$. So, the next two statements are equivalent:

(i) $I \models arg\Big(g_1, \Delta, \neg \bigwedge_{\delta \in \Delta} \big(g_2(\delta) \to \delta\big)\Big)$

(ii) $I \models arg\Big(g_1, \Delta, \neg \bigwedge \Phi_2\Big)$.

Since I represents Φ_1 via g_1, Theorem 1 yields that (ii) holds iff $\langle \Phi_1, \neg \bigwedge \Phi_2 \rangle$ is an argument. Therefore, (i), which means that $suc(g_1, g_2, \Delta)$ is true in I, holds iff $\langle \Phi_1, \neg \bigwedge \Phi_2 \rangle$ is an argument. □

Corollary 2. *For a knowledge base Δ, a formula α, an interpretation I, and $\Phi_1, \Phi_2 \subseteq \Delta$, such that I represents $\langle \Phi_1, \Phi_2 \rangle$ via $\langle g_1, g_2 \rangle$, we have that*

$$arg(g_2, \Delta, \alpha) \wedge suc(g_1, g_2, \Delta)$$

is true under I iff $\langle \Phi_2, \alpha \rangle$ is an argument and $\Phi_1 \in UndercutSupports(\langle \Phi_2, \alpha \rangle)$.

3.2. Argument trees with fixed structure

We now show how to characterise trees via their tuple form using QBFs. We start with encodings where the tree structure is fixed via a given parent function, but the nodes of the tuple form can be arbitrarily characterised by assignments to the atoms from generator functions. In other words, given a parent function p over k and a formula α, we characterise all sequences $\mathcal{A} = \langle \Phi_1, \ldots, \Phi_k \rangle$, such that $\langle \alpha, \mathcal{A}, p \rangle$ is a tuple form. We then refine these encodings to obtain all such sequences $\mathcal{A}$, such that the tuple form $\langle \alpha, \mathcal{A}, p \rangle$ represents a (complete) argument tree. Note that in the forthcoming encodings, we also assume that p^* comes together with p as an input. The aim of the forthcoming module is to ensure that p correctly applies to the sequence $\mathcal{A}$, in such a way that p does not lead to duplicate children notes (as required in Definition 9).

Definition 18. For a knowledge base Δ, a sequence of generator functions $\mathcal{G} = \langle g_1, \ldots, g_k \rangle$, and a parent function p over k, we define

$$distinct(\mathcal{G}, \Delta, p) = \bigwedge_{i,j:\ p(i)=p(j); i \neq j} \neg\Big(\bigwedge_{\delta \in \Delta} g_i(\delta) \leftrightarrow g_j(\delta)\Big).$$

Lemma 5. *For a knowledge base Δ, a parent function p over k, and an interpretation I representing $\mathcal{A} = \langle \Phi_1, \ldots, \Phi_k \rangle$ via $\mathcal{G} = \langle g_1, \ldots, g_k \rangle$, we have that $distinct(\mathcal{G}, \Delta, p)$ is true under I iff p is a parent function for $\mathcal{A}$.*

Proof. (Only if direction) If $distinct(\mathcal{G}, \Delta, p)$ is true in I then for all two distinct i and j (each in the range $1..k$) where $p(i) = p(j)$, there must exist $\delta' \in \Delta$ such that $g_i(\delta') \leftrightarrow g_j(\delta')$ is false in I. So, $I \models g_i(\delta') \wedge \neg g_j(\delta')$ or $I \models \neg g_i(\delta') \wedge g_j(\delta')$. The cases are symmetric, so it is enough to consider the former: $I \models g_i(\delta') \wedge \neg g_j(\delta')$. Since $g_i(\delta')$ and $g_j(\delta')$ are atoms, it then follows that $g_i(\delta') \in I$ and $g_j(\delta') \notin I$. Therefore, $g_j(\delta') \notin I \cap g_j(\Delta)$ whereas $g_i(\delta') \in I \cap g_i(\Delta)$ (as $g_i(\delta') \in g_i(\Delta)$ due to $\delta' \in \Delta$). However, I represents $\mathcal{A} = \langle \Phi_1, \ldots, \Phi_k \rangle$ via $\mathcal{G} = \langle g_1, \ldots, g_k \rangle$ meaning that $I \cap g_i(\Delta)$ is $g_i(\Phi_i)$ and $I \cap g_j(\Delta)$ is $g_j(\Phi_i)$ (Definition 13). Whence $g_i(\delta') \in g_i(\Phi_i)$ and $g_j(\delta') \notin g_j(\Phi_j)$. I.e., $g_i(\delta') \in \{g_i(\delta) \mid \delta \in \Phi_i\}$ and $g_j(\delta') \notin \{g_j(\delta) \mid \delta \in \Phi_j\}$ (cf. Definition 12). As an immediate consequence, $\delta' \notin \Phi_j$. On the other hand, $g_i(\delta') \in \{g_i(\delta) \mid \delta \in \Phi_i\}$ implies $\delta' \in \Phi_i$ because g_i is injective according to Definition 12. Now, $\delta' \notin \Phi_j$ together with $\delta' \in \Phi_i$ yields $\Phi_i \neq \Phi_j$.

(If direction) We must show that $distinct(\mathcal{G}, \Delta, p)$ is true in I. In fact, we show that, for any two distinct i and j (each in the range $1..k$) such that $p(i) = p(j)$, then $g_i(\delta') \leftrightarrow g_j(\delta')$ is false in I for some $\delta' \in \Delta$. To start with, $\Phi_i \neq \Phi_j$ because p is a parent function for $\mathcal{A} = \langle \Phi_1, \ldots, \Phi_k \rangle$. Trivially, there then exists $\delta' \in \Delta$ such that either $\delta' \in \Phi_i$ and $\delta' \notin \Phi_j$ or $\delta' \notin \Phi_i$ and $\delta' \in \Phi_j$. The cases are symmetric, so it is enough to consider the former. From $\delta' \notin \Phi_j$, we easily get $g_j(\delta') \notin \{g_j(\delta) \mid \delta \in \Phi_j\}$, which, by Definition 12, means $g_j(\delta') \notin g_j(\Phi_j)$. That is, $g_j(\delta') \notin I \cap g_j(\Delta)$ because I represents $\mathcal{A} = \langle \Phi_1, \ldots, \Phi_k \rangle$ via $\mathcal{G} = \langle g_1, \ldots, g_k \rangle$ (cf. Definition 13). In view of $g_j(\delta') \in g_j(\Delta)$ (due to $\delta' \in \Delta$), it then follows that $g_j(\delta') \notin I$. On the other hand, $\delta' \in \Phi_i$. So, $g_i(\delta') \in \{g_i(\delta) \mid \delta \in \Phi_i\} = g_i(\Phi_i)$. Then, $g_i(\delta') \in I \cap g_i(\Delta)$ because I represents $\mathcal{A} = \langle \Phi_1, \ldots, \Phi_k \rangle$ via $\mathcal{G} = \langle g_1, \ldots, g_k \rangle$ (cf. Definition 13). So, $g_i(\delta') \in I$. Combined with $g_j(\delta') \notin I$ as proven above, this yields $I \not\models g_i(\delta') \leftrightarrow g_j(\delta')$. □

Given a parent function p, we now know how to characterise sequences $\mathcal{A} = \langle \Phi_1, \ldots, \Phi_k \rangle$ via QBFs, such that, p is a parent function for $\mathcal{A}$. Thus, we can already obtain all tuple forms $\langle \alpha, \mathcal{A}, p \rangle$, for a given p. Next, we add further conditions to get only those $\mathcal{A}$, such that $\langle \alpha, \mathcal{A}, p \rangle$ represents also an argument tree.

Definition 19. For a knowledge base Δ, a formula α, a sequence of generator functions $\mathcal{G} = \langle g_1, \ldots, g_k \rangle$, and a parent function p over k, we define

$$argtree(\mathcal{G}, \alpha, \Delta, p) = arg(g_1, \Delta, \alpha) \wedge \bigwedge_{i=2}^{k} \big(suc(g_i, g_{p(i)}, \Delta)\big) \wedge \bigwedge_{i=2}^{k} \bigvee_{\delta \in \Delta} \Big(g_i(\delta) \wedge \bigwedge_{j \in p^*(i)} \neg g_j(\delta) \Big).$$

Theorem 3. *For a knowledge base Δ, an interpretation I, and a tuple form $\langle \alpha, \mathcal{A}, p \rangle$, such that I represents $\mathcal{A}$ via generator functions $\mathcal{G}$ we have that the QBF*

$$distinct(\mathcal{G}, \Delta, p) \wedge argtree(\mathcal{G}, \alpha, \Delta, p)$$

is true under I iff $\langle \alpha, \mathcal{A}, p \rangle$ represents an argument tree.

Proof. By Lemma 5 and the fact that $\langle \alpha, \mathcal{A}, p \rangle$ is a tuple form (see item 3 in Definition 9), $distinct(\mathcal{G}, \Delta, p)$ is true under I. So, we need only focus on $argtree(\mathcal{G}, \alpha, \Delta, p)$.

Lemma 2 states that $\langle \alpha, \mathcal{A}, p \rangle$ represent an argument tree iff (i) $\langle \Phi_1, \alpha \rangle$ is an argument, and (ii) $\Phi_i \in UndercutSupports(A(p(i)))$ for $1 < i \leqslant k$, and (iii) $\Phi_i \not\subseteq \Phi_{p(i)} \cup \Phi_{p^2(i)} \cup \cdots \cup \Phi_{p^n(i)=1}$ for $1 < i \leqslant k$. We show that each of (i)–(iii) holds iff the corresponding conjunct in $argtree(\mathcal{G}, \alpha, \Delta, p)$ is true under I. By Theorem 1, (i) holds iff $arg(g_1, \Delta, \alpha)$, namely the first conjunct in $argtree(\mathcal{G}, \alpha, \Delta, p)$, is true under I. Let us turn to (ii). In fact, $\Phi_i \in UndercutSupports(A(p(i)))$ means that $\langle \Phi_i, \neg \bigwedge \Phi_{p(i)} \rangle$ is an undercut of $\langle \Phi_{p(i)}, \ldots \rangle$. By Theorem 2, $\langle \Phi_i, \neg \bigwedge \Phi_{p(i)} \rangle$ is an argument iff $suc(g_i, g_{p(i)}, \Delta)$, namely the second conjunct in $argtree(\mathcal{G}, \alpha, \Delta, p)$, is true under I. Let us turn to (iii). Trivially, $\Phi_i \not\subseteq \Phi_{p(i)} \cup \Phi_{p^2(i)} \cup \cdots \cup \Phi_{p^n(i)=1}$ means that there exists $\delta' \in \Phi_i$ such that $\delta' \notin \Phi_{p(i)} \cup \Phi_{p^2(i)} \cup \cdots \cup \Phi_{p^n(i)=1}$. Since I represents $\mathcal{A}$ via $\mathcal{G}$, for all $j = 1..k$, Definition 13 tells us that $g_j(\delta')$ is true under I iff $\delta' \in \Phi_j$. Then, $\Phi_i \not\subseteq \Phi_{p(i)} \cup \Phi_{p^2(i)} \cup \cdots \cup \Phi_{p^n(i)=1}$ iff for some δ', $g_i(\delta')$ is true under I while $g_{p(i)}(\delta'), g_{p^2(i)}(\delta'), \ldots, g_{p^n(i)=1}(\delta')$ are all false under I. So, $\Phi_i \not\subseteq \Phi_{p(i)} \cup \Phi_{p^2(i)} \cup \cdots \cup \Phi_{p^n(i)=1}$ iff

$$\bigvee_{\delta \in \Delta} \Big(g_i(\delta) \wedge \bigwedge_{j \in p^*(i)} \neg g_j(\delta) \Big)$$

is true under I. □

Our next definition captures the condition that for a sequence of generator functions $\mathcal{G}$, and for each argument that can be represented via a generator function g_i in $\mathcal{G}$, if there is an undercut for it that can be represented by a generator function g, then g is also in $\mathcal{G}$.

Definition 20. For a knowledge base Δ, a parent function p over k, a sequence of generator functions $\mathcal{G} = \langle g_1, \ldots, g_k \rangle$, and a further generator function g, we define

$$complete(\mathcal{G}, \Delta, p) = \bigwedge_{i=1}^{k} \forall g(\Delta) \Big(suc(g, g_i, \Delta) \rightarrow \bigvee_{j : p(j)=i} \bigwedge_{\delta \in \Delta} \big(g(\delta) \leftrightarrow g_j(\delta) \big) \Big).$$

Theorem 4. *For a knowledge base Δ, an interpretation I, and a tuple form $\langle \alpha, \mathcal{A}, p \rangle$, such that I represents $\mathcal{A}$ via generator functions $\mathcal{G}$ we have that the QBF*

$$distinct(\mathcal{G}, \Delta, p) \wedge argtree(\mathcal{G}, \alpha, \Delta, p) \wedge complete(\mathcal{G}, \Delta, p)$$

is true under I iff $\langle \alpha, \mathcal{A}, p \rangle$ represents a complete argument tree.

Proof. By Theorem 2, $suc(g, g_i, \Delta)$ is true under I iff $\langle \Psi, \neg \bigwedge \Phi_i \rangle$ is an argument (Ψ is taken to denote the set that g represents under I), or, equivalently, $\langle \Psi, \ldots \rangle$ is an undercut of $\langle \Phi_i, \ldots \rangle$. Since I represents $\mathcal{A}$ via $\mathcal{G}$, Definition 13 means that $g_j(\delta)$ is true under I iff $\delta \in \Phi_j$. Therefore, $g(\delta) \leftrightarrow g_j(\delta)$ is true under I iff $\Psi = \Phi_j$. So, $complete(\mathcal{G}, \Delta, p)$ is true under I iff for each $1 \leqslant i \leqslant k$ and for each $\Psi \in UndercutSupports(A(i))$, there exists an index $j \in \{1..k\}$ such that $\Phi_j = \Psi$ and $p(j) = i$. Then, apply Lemma 2 and Theorem 3. □

As already shown for single arguments, we can use now the $fix(g, \Phi)$ module to encode further decision problems. In our first example (given in Corollary 3), we can ensure that the argument tree has a particular argument as the root of the tree.

Corollary 3. *For a knowledge base Δ, $\Psi \subseteq \Delta$, an interpretation I, and a tuple form $\langle \alpha, \mathcal{A}, p\rangle$, such that I represents $\mathcal{A}$ via generator functions $\mathcal{G}$ we have that the QBF*

$$\mathit{fix}(g_1, \Psi) \wedge \mathit{distinct}(\mathcal{G}, \Delta, p) \wedge \mathit{argtree}(\mathcal{G}, \alpha, \Delta, p)$$

is true under I iff $\langle \alpha, \mathcal{A}, p\rangle$ represents an argument tree with root (Ψ, α).

As a further example, we can check whether a given tuple form $\langle \alpha, \mathcal{A}, p\rangle$ with $\mathcal{A} = \langle \Phi_1, \ldots, \Phi_k\rangle$ via QBFs, represents an argument tree. To this end let, $\mathcal{G}(\Delta) = \bigcup_{i=1}^{k} g_i(\Delta)$.

Corollary 4. *A tuple form $\langle \alpha, \langle \Phi_1, \ldots, \Phi_k\rangle, p\rangle$ represent an argument tree iff the closed QBF*

$$\exists \mathcal{G}(\Delta)\left(\bigwedge_{i=1}^{k} \mathit{fix}(g_i, \Phi_i) \wedge \mathit{distinct}(\mathcal{G}, \Delta, p) \wedge \mathit{argtree}(\mathcal{G}, \alpha, \Delta, p)\right)$$

is true.

Likewise, we can apply these two corollaries to complete argument trees by adding the conjunct $\mathit{complete}(\mathcal{G}, \Delta, p)$ accordingly.

3.3. Argument trees with arbitrary structure

Compared to the previous characterisation, we now shall not only compute the sequence $\mathcal{A}$ for a tuple form $\langle \alpha, \mathcal{A}, p\rangle$ with given p, but also possible parent functions p via the encodings. Hence, we first have to represent functions p as well as its closure p^*. Given a parent function p over k, we use further new atoms $P_k = \{p_{i,j} \mid 1 \leqslant j < i \leqslant k\}$ and $P_k^* = \{p_{i,j}^* \mid 1 \leqslant j < i \leqslant k\}$. Intuitively, if an atom $p_{i,j}$ is true under I, then I is used to characterise a parent function p with $p(i) = j$. To show how this can be done, we first need a weaker notion than a parent function. We sometimes also regard a parent function over k as a binary relation over $\{1, \ldots, k\}$ satisfying the restrictions in the following definition.

Definition 21. For $k = 1$, let $p = \emptyset$, and for $k > 1$, let $p \subseteq \{2..k\} \times \{1..k\}$ be a relation where $(i, j) \in p$ implies $j < i$.
We say that an interpretation I **represents** p **via** P_k iff, for all $1 \leqslant j < i \leqslant k$, $p_{i,j} \in I$ iff $(i, j) \in p$.

The forthcoming propositional encoding has those interpretations as its models which represent relations (according to Definition 21) satisfying the requirement for being a parent functions (according to Definition 9).

Definition 22. For any $k \geqslant 1$, define

$$\mathit{preparent}(P_k) = \bigwedge_{i=2}^{k} \bigvee_{j=1}^{i-1} p_{i,j} \wedge \bigwedge_{i=3}^{k} \bigwedge_{j=2}^{i-1} \left(p_{i,j} \rightarrow \bigwedge_{l=1}^{j-1} \neg p_{i,l}\right).$$

Example 14. Consider $k = 4$. There are six possible trees (i.e., realizations of a parent relation p according to Definition 9) that can be formed from 4 nodes. These can be represented by the following six options:

Option 1	$p(4) = 3$,	$p(3) = 2$,	$p(2) = 1$
Option 2	$p(4) = 3$,	$p(3) = 1$,	$p(2) = 1$
Option 3	$p(4) = 1$,	$p(3) = 2$,	$p(2) = 1$
Option 4	$p(4) = 2$,	$p(3) = 1$,	$p(2) = 1$
Option 5	$p(4) = 2$,	$p(3) = 2$,	$p(2) = 1$
Option 6	$p(4) = 1$,	$p(3) = 1$,	$p(2) = 1$.

By Definition 22, formula $\mathit{preparent}(P_4)$ is as follows

$$p_{2,1} \wedge (p_{3,2} \vee p_{3,1}) \wedge (p_{4,3} \vee p_{4,2} \vee p_{4,1}) \wedge (p_{3,2} \rightarrow \neg p_{3,1}) \wedge (p_{4,2} \rightarrow \neg p_{4,1}) \wedge (p_{4,3} \rightarrow \neg p_{4,1} \wedge \neg p_{4,2}).$$

Note that $(p_{4,3} \rightarrow \neg p_{4,1} \wedge \neg p_{4,2})$ implies $(p_{4,3} \rightarrow \neg p_{4,1})$ and $(p_{4,3} \rightarrow \neg p_{4,2})$. Hence, by contraposition, we get $(p_{4,1} \rightarrow \neg p_{4,3})$ and $(p_{4,2} \rightarrow \neg p_{4,3})$, and thereby get the constraints we require on the relation p to form a parent function.

Lemma 6. *Let $p \subseteq \{2..k\} \times \{1..k\}$ be a relation where $(i, j) \in p$ implies $j < i$, and I be an interpretation, such that I represents p via atoms P_k. Then, the formula $\mathit{preparent}(P_k)$ is true under I iff p is a parent function over k.*

Proof. Since p is such that $(i, j) \in p$ implies $j < i$, the lemma holds iff $preparent(P_k)$ expresses that p is a function. In view of Definition 21, that every i in $\{2..k\}$ has an image by p is expressed by

$$I \models \bigwedge_{i=2}^{k} \bigvee_{j=1}^{i-1} p_{i,j}.$$

That i in $\{2..k\}$ only has one image by p can be expressed as follows: if $(i, j) \in p$ then for all $l \neq j$, $(i, l) \notin p$. Since $(i, l) \in p$ implies $l < i$, this test is only necessary for i in $\{3..k\}$ (observe that $(2, 1)$ always is in p for $k > 1$ and there is no other possibility). Moreover, since the test is checked for all j such that $(i, j) \in p$ holds, it is enough to focus on $(i, l) \notin p$ for all $l < j$. Finally, it is sufficient that j ranges from 2 to $i - 1$. In view of Definition 21, this amounts to

$$I \models \bigwedge_{i=3}^{k} \bigwedge_{j=2}^{i-1} \left(p_{i,j} \rightarrow \bigwedge_{l=1}^{j-1} \neg p_{i,l} \right). \qquad \square$$

Next, we show how to suitably characterise the closure p^* (cf. Definition 11) of a parent function p.

Definition 23. For any $k \geqslant 1$, define

$$closure(P_k, P_k^*) = \bigwedge_{i=2}^{k} \bigwedge_{j=1}^{i-1} \left(p^*_{i,j} \leftrightarrow \left(p_{i,j} \vee \bigvee_{l=j+1}^{i-1} (p_{i,l} \wedge p^*_{l,j}) \right) \right).$$

Lemma 7. *Let p be a parent function over k and $q \subseteq \{2..k\} \times \{1..k\}$ a relation where $(i, j) \in q$ implies $j < i$. Moreover, let I be an interpretation representing p via P_k and q via P_k^*. Then, the formula $closure(P_k, P_k^*)$ is true under I iff $p^*(i) = \{j \mid q(i, j)\}$ for $i = 2..k$.*

Proof. By Definition 21, $closure(P_k, P_k^*)$ is, under I, equivalent, for $i = 2..k$ and $j = 1..i - 1$, to

$$q(i, j) \Leftrightarrow \begin{cases} p(i, j), & \text{or} \\ p(i, l) \text{ and } q(l, j) & \text{for some } l \in \{j + 1, \ldots, i - 1\} \end{cases}$$

which, by virtue of p being a parent function and q being such that $(i, j) \in q$ implies $j < i$, amounts to

$$q(i, j) \Leftrightarrow \begin{cases} p(i, j), & \text{or} \\ p(i, l) \text{ and } q(l, j) & \text{for some } l \in \{1, \ldots, k\} \end{cases}$$

which is known to characterise the transitive closure of p (taken as a relation) provided that p has a finite domain and is acyclic but both points are obvious here. $\square$

Example 15. Consider $k = 4$ as in Example 14. One possible parent function was $p(4) = 3$, $p(3) = 2$, $p(2) = 1$. We use atoms $P_4 = \{p_{2,1}, p_{3,1}, p_{4,1}, p_{3,2}, p_{4,2}, p_{4,3}\}$ and likewise P_4^*. Any interpretation I which assigns true to $p_{2,1}$, $p_{3,2}$, $p_{4,3}$, and false to $p_{3,1}$, $p_{4,1}$, and $p_{4,2}$ represents the above parent function p via P_4. Let us now evaluate $closure(P_4, P_4^*)$ under such I. In fact, we then expect that only those I are models of $closure(P_4, P_4^*)$ which assign true to all atoms in P_4^*. By definition, I then represents p^* (according to Definition 11) via P_4^*. Observe that we have

$$closure(P_4, P_4^*) = (p^*_{2,1} \leftrightarrow p_{2,1}) \wedge \tag{7}$$

$$\left(p^*_{3,1} \leftrightarrow \left(p_{3,1} \vee (p_{3,2} \wedge p^*_{2,1})\right)\right) \wedge \tag{8}$$

$$(p^*_{3,2} \leftrightarrow p_{3,2}) \wedge \tag{9}$$

$$\left(p^*_{4,1} \leftrightarrow \left(p_{4,1} \vee (p_{4,2} \wedge p^*_{2,1}) \vee (p_{4,3} \wedge p^*_{3,1})\right)\right) \wedge \tag{10}$$

$$\left(p^*_{4,2} \leftrightarrow \left(p_{4,2} \vee (p_{4,3} \wedge p^*_{3,2})\right)\right) \wedge \tag{11}$$

$$(p^*_{4,3} \leftrightarrow p_{4,3}). \tag{12}$$

Recall that we consider I assigning true to $p_{2,1}$, $p_{3,2}$, $p_{4,3}$; conjuncts (7), (9), and (12) thus require that $p^*_{2,1}$, $p^*_{3,2}$, $p^*_{4,3}$ are also assigned to true by a model I of $closure(P_4, P_4^*)$. Now we have $p^*_{2,1}$ and $p_{3,2}$ in I. Thus by line (8) also $p^*_{3,1}$ is true in I. Similarly for line (10), we already know that $p^*_{3,1}$ and $p_{4,3}$ are true in I, and we can conclude that also $p^*_{4,1}$ is true in I. Finally, (11) forces also $p^*_{4,2}$ to be true in I as well, since we already have seen that for a model I also $p_{4,3}$ and $p^*_{3,2}$ are true in I.

We are now ready to relate interpretations to parent functions in combination with sequences $\mathcal{A} = \langle \Phi_1, \ldots, \Phi_k \rangle$. For this, we have to guarantee that models represent parent functions (this is done with the already introduced conjunct *preparent*(P_k)) and that the represented parent function correctly relates to a represented sequence $\mathcal{A}$ avoiding duplicate children (in a way that is similar to what we did in Definition 18 for the module *distinct*$(\mathcal{G}, \Delta, p)$). The latter task is realized via the second conjunct in the forthcoming definition.

Definition 24. For a knowledge base Δ, and a sequence of generator functions $\mathcal{G} = \langle g_1, \ldots, g_k \rangle$, define

$$parent(\mathcal{G}, \Delta, P_k) = preparent(P_k) \wedge \bigwedge_{i=3}^{k} \bigwedge_{j=2}^{i-1} \bigwedge_{l=1}^{j-1} \Big((p_{i,l} \wedge p_{j,l}) \rightarrow \neg \bigwedge_{\delta \in \Delta} \big(g_i(\delta) \leftrightarrow g_j(\delta) \big) \Big).$$

Lemma 8. *For a knowledge base Δ, and an interpretation I representing $\mathcal{A} = \langle \Phi_1, \ldots, \Phi_k \rangle$ via $\mathcal{G}$, and a relation p via P_k, we have that parent$(\mathcal{G}, \Delta, P_k)$ is true under I iff p is a parent function for $\mathcal{A}$.*

Proof. We already know from Lemma 6 that *preparent*(P_k) is true under I iff p (represented by I) is a parent function (over k). We thus need to show that the remaining part of *parent*$(\mathcal{G}, \Delta, P_k)$ is true under I iff p is a parent function for $\mathcal{A} = \langle \Phi_1, \ldots, \Phi_k \rangle$, that is, $p(i) = p(j)$ implies $\Phi_i \neq \Phi_j$, for any $1 < j < i \leqslant k$ (in fact, it is sufficient to use $2 < j < i \leqslant k$, since the root has obviously no parent node). Since I also represents $\mathcal{A}$ (via $\mathcal{G} = \langle g_1, \ldots, g_k \rangle$), we have $\Phi_i \neq \Phi_j$ iff there exists some $\delta \in \Delta$ such that $g_i(\delta) \leftrightarrow g_j(\delta)$ is false in I. This holds iff $\neg \bigwedge_{\delta \in \Delta} (g_i(\delta) \leftrightarrow g_j(\delta))$ is true under I. Since we perform this test for each pair of children $A(i)$, $A(j)$ of each node $A(l)$ in the annotated tree (represented by I), the claim follows by the same arguments as used to show Lemma 5. □

Definition 25. For a knowledge base Δ, a formula α, and a sequence of generator functions $\mathcal{G} = \langle g_1, \ldots, g_k \rangle$, we define

$$\begin{aligned} argtree(\mathcal{G}, \alpha, \Delta, P_k, P_k^*) = {} & arg(g_1, \Delta, \alpha) \wedge \\ & \bigwedge_{i=2}^{k} \bigwedge_{j=1}^{i-1} \big(p_{i,j} \rightarrow suc(g_i, g_j, \Delta) \big) \wedge \\ & \bigwedge_{i=2}^{k} \bigvee_{\delta \in \Delta} \Big(g_i(\delta) \wedge \bigwedge_{j=1}^{i-1} \big(p^*_{i,j} \rightarrow \neg g_j(\delta) \big) \Big). \end{aligned}$$

Theorem 5. *For a knowledge base Δ, a formula α, and an interpretation I, representing a sequence $\mathcal{A} = \langle \Phi_1, \ldots, \Phi_k \rangle$ via $\mathcal{G}$, and a relation p via P_k we have that the QBF*

$$AT(\mathcal{G}, \Delta, P_k) = parent(\mathcal{G}, \Delta, P_k) \wedge \exists P_k^* \big(closure(P_k, P_k^*) \wedge argtree(\mathcal{G}, \alpha, \Delta, P_k, P_k^*) \big)$$

is true under I iff $\langle \alpha, \mathcal{A}, p \rangle$ represents an argument tree.

Proof. We only give a sketch here, since we already know the following properties:

- By Lemma 8, I represents a parent function for $\mathcal{A}$ iff *parent*$(\mathcal{G}, \Delta, P_k)$ is true under I;
- By Lemma 7, given an interpretation I representing a parent function p (over k) via P_k, I represents the transitive closure of p via P_k^* iff *closure*(P_k, P_k^*) is true under I;
- Formula *argtree*$(\mathcal{G}, \alpha, \Delta, P_k, P_k^*)$ follows the same structure from Definition 19 but instead of using the functions p, p^* explicitly, we represent them (see Definition 25) by the respective sets P_k and P_k^* of variables. Using this observation and by suitably combining the techniques for P_k and P_k^* (as done in the proofs above) with the structure of the proof of Theorem 3, one can show the following relation: Given Δ, α, and an interpretation I representing $\mathcal{A}$ via $\mathcal{G}$, a parent function p for $\mathcal{A}$ via P_k, and the transitive closure of p, i.e. p^*, via P_k^*, we have that $\langle \alpha, \mathcal{A}, p \rangle$ represents an argument tree iff *argtree*$(\mathcal{G}, \alpha, \Delta, P_k, P_k^*)$ is true under I.

From the latter observation and the semantics of the existential quantifier, the claim holds. Note that we used $\exists P_k^*$ just to "hide" the variables representing the transitive closure of the parent function from the user, since it is not an explicit part of the requested problem, but rather an internal detail which is fully determined by p. □

Definition 26. For a knowledge base Δ, a sequence of generator functions $\mathcal{G} = \langle g_1, \ldots, g_k \rangle$, and a further generator function g, we define

$$complete(\mathcal{G}, \Delta, P_k) = \bigwedge_{i=1}^{k} \forall g(\Delta) \Big(suc(g, g_i, \Delta) \rightarrow \bigvee_{j=1}^{i-1} \Big(p_{i,j} \wedge \bigwedge_{\delta \in \Delta} \big(g(\delta) \leftrightarrow g_j(\delta) \big) \Big) \Big).$$

Theorem 6. *For a knowledge base Δ, a formula α, and an interpretation I, representing a sequence $\mathcal{A} = \langle \Phi_1, \ldots, \Phi_k \rangle$ via $\mathcal{G}$, and a relation p over P_k we have that the QBF*

$$CAT(\mathcal{G}, \Delta, P_k) = parent(\mathcal{G}, \Delta, P_k) \wedge \exists P_k^* \big(closure(P_k, P_k^*) \wedge argtree(\mathcal{G}, \alpha, \Delta, P_k, P_k^*) \wedge complete(\mathcal{G}, \Delta, P_k) \big)$$

is true under I iff $\langle \alpha, \mathcal{A}, p \rangle$ represents a complete argument tree.

The proof of Theorem 6 is similar to the proof for Theorem 4 except that at the end, Theorem 5 instead of Theorem 3 is required.

Again, we now can decide different decision problems by using above concepts plus fixing some of the concepts. An interesting question is as follows: Given α and $\mathcal{A}$, does there exist an argument tree (a complete argument tree) using sets $\mathcal{A}$? In other words, can we find a parent function p, such that $\langle \alpha, \mathcal{A}, p \rangle$ represents an argument tree (a complete argument tree)?

Corollary 5. *Given $\mathcal{A} = \Phi_1, \ldots, \Phi_k$ and a formula α, there exists a parent function p, such that $\langle \alpha, \mathcal{A}, p \rangle$ represents*

1. *an argument tree iff the closed QBF*

$$\exists P_k \exists \mathcal{G}(\Delta) \left(\bigwedge_{i=1}^{k} fix(g_i, \Phi_i) \wedge AT(\mathcal{G}, \Delta, P_k) \right)$$

   is true;
2. *a complete argument tree iff the closed QBF*

$$\exists P_k \exists \mathcal{G}(\Delta) \left(\bigwedge_{i=1}^{k} fix(g_i, \Phi_i) \wedge CAT(\mathcal{G}, \Delta, P_k) \right)$$

   is true.

3.4. Remarks

We give a few remarks concerning the actual size of the encodings provided in the previous sections compared to the size of the encoded problem.

First, we address the number of additional atoms (mostly stemming from the generator functions g) required in the encodings. For the encodings given in Section 3.1, note that we require a new atom for each formula in Δ. Hence, if the cardinality of Δ is n, we need n new atoms for most of the encodings in that section, with the exception of the encodings where $suc(\cdot)$ is involved where we used two generator functions g_1, g_2 resulting in $2n$ new atoms. For the encodings in Section 3.2, where we used tuple forms of length k to represent trees with k nodes, also the size of the tree comes into play. One can check that the number of additional atoms required for the encodings in that section is bound by $n(k+1)$. Finally, encodings of argument trees with arbitrary structure (Section 3.3) require a much higher number of additional atoms due to the task of representing the tree structure via further sets P_k, P_k^* of atoms. Note that these sets are of cardinality $k(k-1)/2$. Together with the atoms stemming from the generator functions, we get here a need for $n(k+1) + k^2$ new atoms.

Concerning the size of the encodings, we just mention that the size is linear in the size of Δ for the encodings in Section 3.1. For the more involved encodings in Sections 3.2 and 3.3, the size of the encodings is at most quadratic in the size of Δ and the size of tree.

However, both the theoretical point of view as well as practical experience tell us that the nesting depth of alternating quantifiers is the most crucial parameter for evaluating QBFs, although the parameters of formula size and number of atoms cannot be ignored. We recall that for all our encodings this nesting depth is fixed and independent of the size of Δ or the size of the considered argument tree. In fact, one can check that there are at most two quantifier alternations at each branch of a formula tree associated with our encodings.

As a final remark, we mention that the number of new atoms has to be increased if one wants to employ QBF solvers which rely on inputs in certain normal forms. Here, the transformation to such a normal form further introduces new atoms. Most of the current QBF solvers (see [28]) require such a normal form, but there are also solvers (e.g. [21]) which can be applied to the encodings directly (modulo some transformation steps which do not introduce new atoms).

4. Discussion

There is increasing interest in formalisations for argumentation, and in particular computational models of argument (see for example [4,6,8,13,17,31]). In this paper, we have addressed this issue in the context of argumentation with classical logic as the underlying logic by providing encodation in terms of quantified Boolean formulae. This approach is beneficial with respect to several aspects.

First, it offers the possibility of implementing decision procedures for argumentation based on classical logic using existing QBF solvers.

Second, it allows to obtain novel complexity results for interesting decision problems associated with logic-based argumentation. Indeed, while for abstract argumentation, there has been a comprehensive analysis of computational complexity of some of the key decision problems (in particular [16]), there are only a few published results concerning computational complexity of logic-based argumentation. In [29], for instance, it is shown that given a knowledge-base Δ and a formula α, ascertaining whether there is a $\Phi \subseteq \Delta$ such that $\langle \Phi, \alpha \rangle$ is an argument (i.e. Φ is a minimal consistent set of premises entailing α) is a Σ_2^p-complete decision problem. Our results can be employed to obtain similar results for more involved decision problems. In fact, since all encodings are constructible in polynomial time with respect to the size of the problem description, inspecting the quantifier structure of the encodings and applying Proposition 4 immediately yields upper complexity bounds for the encoded problems. For instance, this shows that several decision problems formulated for argumentation trees remain located in Σ_2^p. Since the evaluation of a single argument is already hard for this class, Σ_2^p-completeness for those decision problems over argumentation trees is expected. However, if complete argumentation trees are taken into account, our encodings indicate that this leads to an increasing complexity, having such problems located in Σ_3^p. Establishing exact complexity results for numerous decision problems in logic-based argumentation is indeed part of our ongoing work.

In another approach to deductive argumentation, Wooldridge et al. [34] show that by taking a "maximal" set of arguments (i.e. a set of arguments that does not include "equivalent" arguments), they can treat the set of arguments as an abstract argument system with the attack relation holding between a pair of arguments A and A' when A is a defeater of A'. This means that the abstract argument system can be evaluated using the notions of acceptability defined by Dung [14]. This offers a different way of assembling and evaluating arguments to that considered in this paper. Furthermore, they provide complexity results concerning the identification of the "maximal" sets of arguments.

Finally, our results are useful for comparing different approaches to argumentation. In fact, there is increasing interest in algorithms and implementations for argumentation systems including for abstract argumentation systems [2,11,12,32], for assumption-based argumentation systems [15,25,27], for logic-based argumentation systems based on defeasible logic [9,10,26,33], and for logic-based argumentation systems based on classical logic [18]. Undertaking empirical evaluations that compare these algorithms is difficult because of the diverse approaches taken in implementing them. So undertaking evaluations via encodations as QBFs offers the opportunity for a level playing field for comparisons that draw out the strengths and weaknesses of each of the algorithms and their underlying reasoning mechanisms. In particular, comparisons with QBF encodings for other argumentation formulations [22] or, encodings in terms of classical logic for nonmonotonic formalisms in general [3,19] are now enabled.

Acknowledgements

This work was supported by the Vienna Science and Technology Fund (WWTF) under grant ICT08-028.

References

[1] L. Amgoud, C. Cayrol, A model of reasoning based on the production of acceptable arguments, Annals of Mathematics and Artificial Intelligence 34 (2002) 197–216.
[2] P. Baroni, M. Giacomin, Argumentation through a distributed self-stabilizing approach, Journal of Experimental and Theoretical Artificial Intelligence 14 (4) (2002) 273–301.
[3] R. Ben-Eliyahu, R. Dechter, Default reasoning using classical logic, Artificial Intelligence 84 (1–2) (1996) 113–150.
[4] T. Bench-Capon, P. Dunne, Argumentation in artificial intelligence, Artificial Intelligence 171 (10–15) (2007) 619–641.
[5] S. Benferhat, D. Dubois, H. Prade, Argumentative inference in uncertain and inconsistent knowledge bases, in: Proceedings of the 9th Annual Conference on Uncertainty in Artificial Intelligence (UAI 1993), Morgan Kaufmann, 1993, pp. 411–419.
[6] Ph. Besnard, S. Doutre, A. Hunter (Eds.), Computational Models of Argument: Proceedings of COMMA 2008, IOS Press, 2008.
[7] Ph. Besnard, A. Hunter, A logic-based theory of deductive arguments, Artificial Intelligence 128 (2001) 203–235.
[8] Ph. Besnard, A. Hunter, Elements of Argumentation, MIT Press, 2008.
[9] D. Bryant, P. Krause, A review of current defeasible reasoning implementations, Knowledge Engineering Review 23 (3) (2008) 227–260.
[10] D. Bryant, P. Krause, G. Vreeswijk, Argue tuprolog: A lightweight argumentation engine for agent applications, in: P. Dunne, T. Bench-Capon (Eds.), Computational Models of Argumentation (COMMA 2006), IOS Press, 2006, pp. 27–31.
[11] C. Cayrol, S. Doutre, J. Mengin, Dialectical proof theories for the credulous preferred semantics of argumentation frameworks, in: Quantitative and Qualitative Approaches to Reasoning with Uncertainty (ECSQARU 2001), in: LNCS, vol. 2143, Springer, 2001, pp. 668–679.
[12] C. Cayrol, S. Doutre, J. Mengin, On decision problems related to the preferred semantics for argumentation frameworks, Journal of Logic and Computation 13 (3) (2003) 377–403.
[13] C. Chesñevar, A. Maguitman, R. Loui, Logical models of argument, ACM Computing Surveys 32 (2000) 337–383.
[14] P. Dung, On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games, Artificial Intelligence 77 (2) (1995) 321–358.
[15] P. Dung, R. Kowalski, F. Toni, Dialectical proof procedures for assumption-based admissible argumentation, Artificial Intelligence 170 (2006) 114–159.
[16] P. Dunne, T. Bench-Capon, Coherence in finite argumentation systems, Artificial Intelligence 141 (2002) 187–203.
[17] P. Dunne, T. Bench-Capon (Eds.), Computational Models of Argument: Proceedings of COMMA 2006, IOS Press, 2006.
[18] V. Efstathiou, A. Hunter, Algorithms for effective argumentation in classical propositional logic, in: Proceedings of the International Symposium on Foundations of Information and Knowledge Systems (FOIKS 2008), in: LNCS, vol. 4932, Springer, 2008, pp. 272–290.
[19] U. Egly, T. Eiter, H. Tompits, S. Woltran, Solving advanced reasoning tasks using quantified boolean formulas, in: Proceedings of the 17th National Conference on Artificial Intelligence (AAAI'00), AAAI Press/MIT Press, 2000, pp. 417–422.

[20] U. Egly, M. Seidl, H. Tompits, S. Woltran, M. Zolda, Comparing different prenexing strategies for quantified boolean formulas, in: Proceedings of the 6th International Conference on the Theory and Applications of Satisfiability Testing (SAT-03), Selected Revised Papers, in: LNCS, vol. 2919, 2004, pp. 214–228.
[21] U. Egly, M. Seidl, S. Woltran, A solver for QBFs in nonprenex form, Constraints 14 (1) (2009) 38–79.
[22] U. Egly, S. Woltran, Reasoning in argumentation frameworks using quantified Boolean formulas, in: Proceedings COMMA'06, IOS Press, 2006, pp. 133–144.
[23] T. Eiter, G. Gottlob, The complexity of logic-based abduction, Journal of the ACM 42 (1995) 3–42.
[24] M. Elvang-Gøransson, P. Krause, J. Fox, Dialectic reasoning with classically inconsistent information, in: Proceedings of the 9th Conference on Uncertainty in Artificial Intelligence (UAI 1993), Morgan Kaufmann, 1993, pp. 114–121.
[25] D. Gaertner, F. Toni, Computing arguments and attacks in assumption-based argumentation, IEEE Intelligent Systems 22 (6) (2007) 24–33 (special issue on argumentation technology).
[26] A. García, G. Simari, Defeasible logic programming: An argumentative approach, Theory and Practice of Logic Programming 4 (1) (2004) 95–138.
[27] A. Kakas, F. Toni, Computing argumentation in logic programming, Journal of Logic and Computation 9 (1999) 515–562.
[28] M. Narizzano, L. Pulina, A. Tacchella, Report of the third QBF solvers evaluation, Journal of Satisfiability, Boolean Modeling and Computation 2 (2006) 145–164.
[29] S. Parsons, M. Wooldridge, L. Amgoud, Properties and complexity of some formal inter-agent dialogues, Journal of Logic and Computation 13 (3) (2003) 347–376.
[30] H. Prakken, G. Sartor, Argument-based extended logic programming with defeasible priorities, Journal of Applied Non-Classical Logics 7 (1997) 25–75.
[31] H. Prakken, G. Vreeswijk, Logical systems for defeasible argumentation, in: D. Gabbay (Ed.), Handbook of Philosophical Logic, Kluwer, 2002.
[32] M. South, G. Vreeswijk, J. Fox, Dungine: A java dung reasoner, in: Computational Models of Argumentation (COMMA 2008), IOS Press, 2008, pp. 360–368.
[33] G. Vreeswijk, An algorithm to compute minimally grounded defence sets in argument systems, in: P. Dunne, T. Bench-Capon (Eds.), Computational Models of Argumentation (COMMA 2006), IOS Press, 2006, pp. 109–120.
[34] M. Wooldridge, P. Dunne, S. Parsons, On the complexity of linking deductive and abstract argument systems, in: Proceedings of the Twenty First National Conference on Artificial Intelligence (AAAI-06), MIT Press, 2006, pp. 299–304.
[35] C. Wrathall, Complete sets and the polynomial-time hierarchy, Theoretical Computer Science 3 (1) (1976) 23–33.