Algorithms for generating arguments and counterarguments in propositional logic

Vasiliki Efstathiou, Anthony Hunter

Department of Computer Science, University College London,

Gower Street, London, WC1E 6BT, UK

Abstract

A common assumption for logic-based argumentation is that an argument is a pair $\langle \Phi, \alpha \rangle$ where Φ is minimal subset of the knowledgebase such that Φ is consistent and Φ entails the claim α. Different logics provide different definitions for consistency and entailment and hence give us different options for formalising arguments and counterarguments. The expressivity of classical propositional logic allows for complicated knowledge to be represented but its computational cost is an issue. In previous work we have proposed addressing this problem using connection graphs and resolution in order to generate arguments for claims that are literals. Here we propose a development of this work to generate arguments for claims that are disjunctive clauses of more than one disjunct, and also to generate counteraguments in the form of canonical undercuts (i.e. arguments that with a claim that is the negation of the conjunction of the support of the argument being undercut).

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 are a number of proposals for logic-based formalisations of argumentation (for reviews see [13, 34, 6]). These proposals allow for the representation of arguments for and against some claim, and for counterargument relationships between arguments. In a number of key examples of argumentation systems, an argument is a pair where the first item in the pair is a minimal consistent set of formulae that proves the second item which is a formula (see

Email addresses: v.efstathiou@cs.ucl.ac.uk (Vasiliki Efstathiou), a.hunter@cs.ucl.ac.uk (Anthony Hunter)

for example [4, 21, 5, 1, 23]). Proof procedures and algorithms have been developed for finding preferred arguments from a knowledgebase using defeasible logic and following for example Dung's preferred semantics (see for example [10, 36, 33, 27, 12, 14, 15]). However, these techniques and analyses do not offer any ways of ameliorating the computational complexity inherent in finding arguments for classical logic.

In this paper we are concerned with arguments and counterarguments based on classical logic. The essential difficulty in this task can be explained as follows. Suppose we use an automated theorem prover (an ATP). If we seek arguments for a particular claim α given a knowledgebase Δ, we need to post queries to the ATP to ensure that a particular set of premises entails α, 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 $\Phi' \vdash \alpha$. This then raises the question of which subsets Φ of Δ to investigate when we are searching for an argument for α. Moreover, if we want to find all arguments for a claim in Δ, in the worst case we need to consider all subsets of Δ.

It is with these issues in mind that we explore an alternative way of finding all the arguments from a knowledgebase Δ for a claim α. Our approach is to adapt the idea of connection graphs to enable us to find arguments. A connection graph [28, 29] is a graph where a clause is represented by a node and an arc (ϕ, ψ) denotes that there is a disjunct in ϕ with its complement being a disjunct in ψ. Essentially, to find arguments with a claim α, the set of complements of the disjuncts of α together with the knowledgebase is used to construct the connection graph. Then, for any clause ϕ in the graph for which there is a disjunct b in the clause and there is no arc (ϕ, ψ) where the complement of b is a disjunct in ψ, the clause ϕ is deleted together with any arcs involving ϕ. The reason for deleting ϕ is that it cannot be used in any proof of α. This process of deletion of clauses is continued until no more clauses can be identified for deletion. If the resulting graph is non-empty, then if there is a set of formulae from Δ that entails α, this will be contained in this part of the graph. Essentially this graph shows this as a proof by contradiction. Furthermore, finding this set of formulae can substantially reduce the number of formulae that need to be considered for finding proofs for α, and therefore for finding arguments for α.

In previous work [16], we have proposed a framework for using connection graphs for finding arguments from a knowledgebase of clauses where each claim is a literal. So, in this paper we generalize the framework in order to generate arguments where each claim is a clause. In addition, we generalize the framework to generate counteraguments for an argument and we give a description of a

software implementation of this framework.

We proceed as follows: in section 2 we review the basic definitions we require for argumentation based on classical logic; in section 3 we focus on properties of deduction with the language of clauses in classical logic; in section 4 we review definitions for connection graphs; in section 5 we review and extend our notion of a support tree that we use to identify a minimal consistent set of formulae that entail a clause; in section 6 we provide some results for finding counterarguments to an argument; in section 7 we present algorithms for generating arguments and counterarguments based on the theoretical developments in the previous sections and in section 8 we conclude and discuss future work.

2. Preliminaries

We review an existing approach to argumentation based on classical logic [5]. Also see [8] for an introductory presentation. In this paper we use classical propositional logic.

Definition 1. *Let Δ be a set of propositional formulae. 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. *For a knowledgebase $\Delta = \{a, b, c, a \rightarrow \neg b \vee c, a \rightarrow \neg b, d, \neg d, \neg c, d \rightarrow a, d \rightarrow e\}$ some arguments include:*

$\langle \{a \rightarrow \neg b\}, \neg(a \wedge b) \rangle$ $\quad$ $\langle \{a, a \rightarrow \neg b\}, \neg b \vee \neg c \rangle$ $\quad$ $\langle \{a, a \rightarrow \neg b\}, \neg b \rangle$
$\langle \{\neg c\}, \neg c \vee b \rangle$ $\quad$ $\langle \{\neg c\}, \neg c \rangle$ $\quad$ $\langle \{a, b, a \rightarrow \neg b \vee c\}, c \rangle$

Given two arguments, it is possible to compare them in terms of how more general is one than the other. The following definition captures this relation between arguments.

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. *Continuing example 1, $\langle \{a \rightarrow \neg b\}, \neg(a \wedge b) \rangle$, is more conservative than $\langle \{a, a \rightarrow \neg b\}, \neg b \rangle$ and $\langle \{a, a \rightarrow \neg b\}, \neg b \vee \neg c \rangle$ is more conservative than $\langle \{a, a \rightarrow \neg b\}, \neg b \rangle$. Also, $\langle \{\neg c\}, \neg c \vee b \rangle$ is more conservative than $\langle \{\neg c\}, \neg c \rangle$.*

Given an argument $A = \langle \Phi, \alpha \rangle$, a number of counterarguments can be presented, where a counterargument for A is an argument A' that negates the premises of the support of A. This can be formalised using the notion of an undercut defined below.

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

Example 3. *Let* $\langle \Phi, \alpha \rangle$ *be the argument* $\langle \{a, b, a \rightarrow \neg b \vee c\}, c \rangle$ *of example 1. Then, arguments* $\langle \{a \rightarrow \neg b\}, \neg(a \wedge b) \rangle$*, and* $\langle \{a, a \rightarrow \neg b\}, \neg b \rangle$ *are undercuts for* $\langle \Phi, \alpha \rangle$*.*

Since undercuts for an argument are themselves arguments, they can be compared on which is more conservative than another. So for instance, for the arguments of example 3 we can say that $\langle \{a \rightarrow \neg b\}, \neg(a \wedge b) \rangle$, is a more conservative undercut than $\langle \{a, a \rightarrow \neg b\}, \neg b \rangle$ and $\langle \{a, a \rightarrow \neg b\}, \neg b \vee \neg c \rangle$ is a more conservative undercut than $\langle \{a, a \rightarrow \neg b\}, \neg b \rangle$. Given a set of undercuts for an argument, the notion of the most conservative undercut is captured in the definition of a maximally conservative undercut defined below.

Definition 4. $\langle \Psi, \beta \rangle$ *is a* **maximally conservative undercut** *for* $\langle \Phi, \alpha \rangle$ *iff 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* $\beta' \vdash \beta$*.*

Example 4. *Continuing example 3,* $\langle \{a \rightarrow \neg b\}, \neg(a \wedge b) \rangle$ *is a maximally conservative undercut for* $\langle \{a, b, a \rightarrow \neg b \vee c\}, c \rangle$*.*

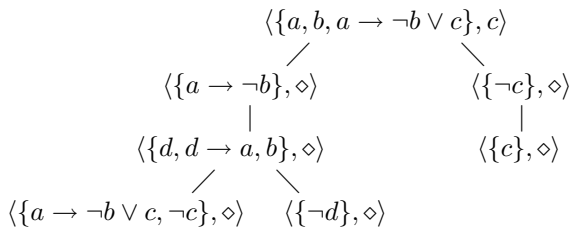
Definition 5. $\langle \Psi, \neg(\phi_1 \wedge \ldots \phi_n) \rangle$ *is a* **canonical undercut** *for* $\langle \Phi, \alpha \rangle$ *iff it is a maximally conservative undercut for* $\langle \Phi, \alpha \rangle$ *and* $\{\phi_1, \ldots, \phi_n\}$ *is the canonical enumeration of* Φ*.*

Example 5. $\langle \{a \rightarrow \neg b\}, \neg(a \wedge b \wedge (a \rightarrow \neg b \vee c)) \rangle$ *is a maximally conservative undercut for* $\langle \{a, b, a \rightarrow \neg b \vee c\}, c \rangle$*.*

For simplicity we will be using the notation $\langle \Psi, \diamond \rangle$ to denote a canonical undercut for $\langle \Phi, \alpha \rangle$. Given a knowledgebase Δ and an argument $A = \langle \Phi, \alpha \rangle$ from Δ there may be a number of canonical undercuts $A'_1, \ldots, A'_n$ for A from Δ. Similarly, for each canonical undercut A'_i of A there may be a number of canonical undercuts $A''_1, \ldots A''_m$ and so on for each A''_j. The different ways an argument can be challenged by the set of all its canonical undercuts and the way these can in turn be challenged and so on can be depicted in a tree structure, the argument tree, where the root is an argument A and each branch is a sequence of arguments where each argument is a canonical argument for its parent in the tree.

Definition 6. *An* **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* $\Phi_1 \cup \ldots \cup \Phi_n$
(3) *The children nodes of a node* N *consist of all canonical undercuts for* N *that obey* (2)*.*

Example 6. *Let* Δ *be the knowledgebase of example 1,* $\Delta = \{a, b, c, a \rightarrow \neg b \vee c, a \rightarrow \neg b, d, \neg d, \neg c, d \rightarrow a, d \rightarrow e\}$*. Then, the following is an argument tree for* $\alpha = c$*.*



The argument tree is a compact way of presenting all the possible sequences of conflicting arguments from a knowledgebase, initialized by the root node. Conditions (2) and (3) of definition 6 ensure that each argument on a branch includes at least one premise not used in any of its ancestor arguments (i.e. arguments above it on the branch to the root argument). So, given a finite knowledgebase, each branch will be finite, and furthermore there is finite branching, meaning each tree is finite.

In this paper we restrict the language we use to a language of disjunctive clauses. The definitions of this section remain the same for this restricted language which is introduced in the next section.

3. Argumentation for a language of clauses

We use a language of clauses $\mathcal{C}$ composed from a set of atoms $\mathcal{A}$ as follows: If a is an atom, then a is a **positive literal**, and $\neg a$ is a **negative literal**. If b is a positive literal, or b is a negative literal, then b is a **literal**. We denote $\mathcal{L}$ the set of all literals. Literals b_1, b_2 are complementary iff they consist of the same atom and one of them is negative and the other one is positive. Then, we say that b_1 is the complement of b_2 and b_2 is the complement of b_1 and we write $b_1 = \overline{b}_2$. If $b_1, .., b_n$ are literals, then $b_1 \vee ... \vee b_n$ is a **clause**. A **clause knowledgebase** is a set of clauses. We consider for simplicity clauses that are equivalent to be identical and so different permutatations of the same set of literals in a disjuntion are considered to be identical clauses and will not appear in the same set.

We use lowercase roman characters to denote atoms, lowercase Greek characters to denote clauses and uppercase Greek characters to denote sets of clauses. We use the symbol $\bot$ to denote the empty clause, that is the clause $b_1 \vee ... \vee b_n$ with $\{b_1, ..., b_n\} = \emptyset$. Then, any set of clauses that contains $\bot$ is unsatisfiable.

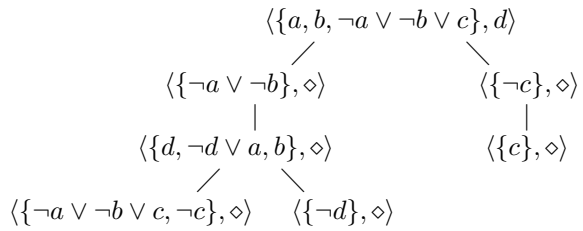
The definition of an argument in $\mathcal{C}$ remains the same, and hence the definitions for a counterargument, a canonical undercut and an argument tree also remain the same, with the restriction that the knowledgebases and the claims considered are from $\mathcal{C}$.

Example 7. *Let* $\Delta' = \{a, b, c, \neg a \vee \neg b \vee c, \neg a \vee \neg b, d, \neg d, \neg c, \neg d \vee a, \neg d \vee e\}$. *Then, observe that for this example there is a one-to-one correspondence between the elements of* Δ' *and those of* $\Delta = \{a, b, c, a \rightarrow \neg b \vee c, a \rightarrow \neg b, d, \neg d, \neg c, d \rightarrow$

$a, d \rightarrow e\}$ *from the examples of section 2. The arguments equivalent to those of example 1 are*

$\langle\{\neg a \vee \neg b\}, \neg a \vee \neg b\rangle$ $\langle\{a, \neg a \vee \neg b\}, \neg b \vee \neg c\rangle$ $\langle\{a, \neg a \vee \neg b\}, \neg b\rangle$
$\langle\{\neg c\}, \neg c \vee b\rangle$ $\langle\{\neg c\}, \neg c\rangle$ $\langle\{a, b, \neg a \vee \neg b \vee c\}, c\rangle$

Example 8. *Using the equivalent* Δ' *of* Δ *as described in example 7 we obtain the following argument tree for* $\alpha = c$ *where each node is logically equivalent to a node from the argument tree of example 6.*



We now present some properties of $\mathcal{C}$. We introduce some basic relations on the elements of $\mathcal{C}$ that will be the basis for the definitions throughout the rest of this paper.

3.1. Relations on $\mathcal{C}$

The definitions of this section are mainly on disjunctive clauses. We start though by introducing functions Disjuncts and Conjuncts for formulae in conjunctive normal form and disjunctive normal form respectively which we will be using in later sections for proofs involving formulae in canonical form that are not necessarily in $\mathcal{C}$. Function Disjuncts is used in this section to define further relations in $\mathcal{C}$ for CNFs of one conjunct.

Definition 7. *Let* ϕ *be a formula in conjunctive normal form:* $\phi = \gamma_1 \wedge \ldots \wedge \gamma_n$*, where each of* $\gamma_1, \ldots, \gamma_n$ *is a disjunction of literals and let* ψ *be a formula in disjunctive normal form,* $\psi = \delta_1 \vee \ldots \vee \delta_n$*, where each of* $\delta_1, \ldots, \delta_n$ *is a conjunction of literals. Then,* $\mathsf{Conjuncts}(\phi)$ *returns the set of disjunctive clauses in* ϕ *i.e.* $\mathsf{Conjuncts}(\phi) = \{\gamma_1, \ldots, \gamma_n\}$ *and* $\mathsf{Disjuncts}(\psi)$ *returns the set of conjunctive clauses in* ψ *i.e.* $\mathsf{Disjuncts}(\psi) = \{\delta_1, \ldots, \delta_n\}$*.*

Hence, functions Conjuncts and Disjuncts return the set of conjuncts or disjuncts respectively for formulae in CNF or DNF. Using the Disjuncts function on disjunctive clauses we define the following binary relations in $\mathcal{C}$.

Definition 8. *Let* $\phi, \psi \in \mathcal{C}$ *and* $b \in \mathcal{C} \cap \mathcal{L}$*. Then, functions* Preattacks *and* Attacks *are defined as follows:*

i) $\mathsf{Preattacks}(\phi, \psi) = \{b \mid b \in \mathsf{Disjuncts}(\phi)$ *and* $\overline{b} \in \mathsf{Disjuncts}(\psi)\}$
ii) If $\mathsf{Preattacks}(\phi, \psi) = \{b\}$ *for some* b *then*
$\mathsf{Attacks}(\phi, \psi) = b$ *otherwise* $\mathsf{Attacks}(\phi, \psi) = null$

Hence, the $\mathsf{Preattacks}$ relation is defined for any pair of clauses ϕ, ψ and returns the set of complementary literals between these clauses while the $\mathsf{Attacks}$ relation is defined for a pair of clauses ϕ, ψ for which $|\mathsf{Preattacks}(\phi, \psi)| = 1$ and returns the literal that is contained in $\mathsf{Preattacks}(\phi, \psi)$.

Example 9. *According to definition 8, the following hold.* $\mathsf{Preattacks}(a \vee \neg b \vee \neg c \vee d, a \vee b \vee \neg d \vee e) = \{\neg b, d\}$, $\mathsf{Preattacks}(a \vee b \vee \neg d \vee e, a \vee \neg b \vee \neg c \vee d) = \{b, \neg d\}$, $\mathsf{Preattacks}(a \vee b \vee \neg d, a \vee b \vee c) = \emptyset$, $\mathsf{Preattacks}(a \vee b \vee \neg d, a \vee b \vee d) = \{\neg d\}$, $\mathsf{Preattacks}(a \vee b \vee \neg d, e \vee c \vee d) = \{\neg d\}$.

Example 10. *According to definition 8, the following hold.* $\mathsf{Attacks}(a \vee \neg b \vee \neg c \vee d, a \vee b \vee \neg d \vee e) = null$, $\mathsf{Preattacks}(a \vee b \vee \neg d \vee e, a \vee \neg b \vee \neg c \vee d) = null$, $\mathsf{Attacks}(a \vee b \vee \neg d, a \vee b \vee c) = null$, $\mathsf{Attacks}(a \vee b \vee \neg d, a \vee b \vee d) = \neg d$, $\mathsf{Attacks}(a \vee b \vee \neg d, e \vee c \vee d) = \neg d$.

For a set of clauses Δ, $\mathsf{Literals}(\Delta)$ returns the set of literals that appear as disjuncts in the elements of Δ.

Definition 9. *Let Δ be a set of clauses. Then* $\mathsf{Literals}(\Delta) = \bigcup_{\delta \in \Delta} \{d \mid d \in \mathsf{Disjuncts}(\delta)\}$

Next, the resolution rule is defined for a pair of clauses that have exactly one pair of complementary literals between them.

Definition 10. *If ϕ and ψ are clauses and* $\mathsf{Attacks}(\phi, \psi) = \{b\}$ *then,*

$$\phi \bullet \psi = \bigvee (\mathsf{Disjuncts}(\phi) \cup \mathsf{Disjuncts}(\psi)) \setminus \{b, \bar{b}\})$$

Hence, $\bullet$ denotes the function of resolution i.e. for a pair of clauses ϕ and ψ, $\phi \bullet \psi$ is the clause that is obtained by resolution from ϕ and ψ. For simplicity, when the function appears in a sequence of more than two clauses, we do not use brackets and we consider that the order in which the resolution function applies to the clauses is the order in which the clauses appear in the sequence. For example $\phi \bullet \chi \bullet \psi \bullet \alpha = ((\phi \bullet \chi) \bullet \psi) \bullet \alpha$

Example 11. *Continuing examples 9 and 10,* $a \vee \neg b \vee \neg c \vee d \bullet a \vee b \vee \neg d \vee e$, $a \vee b \vee \neg d \vee e \bullet a \vee \neg b \vee \neg c \vee d$ *and* $a \vee b \vee \neg d \bullet a \vee b \vee c$ *are not defined, while* $a \vee b \vee \neg d \bullet a \vee b \vee d = a \vee b$, $a \vee b \vee \neg d \bullet e \vee c \vee d = a \vee b \vee e \vee c$.

Now we present some definitions and results from [35] that demonstrate how using the resolvents of pairs of clauses from a knowledgebase can be used to decide on the satisfiability of the knowledgebase.

Definition 11. *Let Φ be a set of clauses. Then,* $\mathsf{Resolve}(\Phi)$ *returns the set of clauses that consists of the members of Φ together with all the resolvents of all pairs of clauses from Φ.* $\mathsf{Resolve}^n(\Phi)$ *is defined for each $n \geq 0$ as follows:* $\mathsf{Resolve}^0(\Phi) = \Phi$ *and* $\mathsf{Resolve}^{n+1}(\Phi) = \mathsf{Resolve}(\mathsf{Resolve}^n(\Phi))$

Definition 12. *For a set of clauses* Φ, $\mathsf{Resolvents}(\Phi) = \mathsf{Resolve}^n(\Phi)$ *iff for some* $n \geq 0$, $\mathsf{Resolve}^{n+1}(\Phi) = \mathsf{Resolve}^n(\Phi)$.

For a finite set of clauses Φ there is a finite number of sets $\mathsf{Resolve}^1(\Phi)$,..., $\mathsf{Resolve}^n(\Phi)$ such that $\mathsf{Resolve}^i(\Phi) \neq \mathsf{Resolve}^{i+1}(\Phi)$. According to the ground resolution theorem that follows, applying resolution recursively on a set of clauses Φ can be used to test Φ for satisfiability.

Theorem 1. *(Robinson 1965) Let* Φ *be a finite set of clauses. Then,* Φ *is unsatisfiable iff* $\mathsf{Resolve}^n(\Phi)$ *contains* $\perp$ *for some* $n \geq 0$.

Example 12. *Let* $\Phi = \{a \vee b, \neg b, \neg a \vee c, \neg c\}$. *Then, according to definition 11,* $\mathsf{Resolve}^0(\Phi) = \Phi$, $\mathsf{Resolve}^1(\Phi) = \{a \vee b, \neg b, a, \neg a \vee c, \neg c, \neg a, b \vee c\}$, $\mathsf{Resolve}^2(\Phi) = \{a \vee b, \neg b, a, \neg a \vee c, \neg c, \neg a, b \vee c, b, c, \perp\}$ *where* $\perp$ *in* $\mathsf{Resolve}^2(\Phi)$ *is the resolvent of* a *and* $\neg a$ *from* $\mathsf{Resolve}^1(\Phi)$. *Hence,* Φ *is unsatisfiable because* $\mathsf{Resolve}^2(\Phi)$ *contains* $\perp$.

Exhaustive generation of the resolvents from a set of clauses Φ can be highly repetitive and the number of clauses produced in every iteration can be large. There are proposals introducing restrictions to the way the resolvent clauses for Φ are generated when testing a set for satisfiability. They introduce strategies that help reduce the number of resolvents produced until the empty clause is reached when testing Φ for satisfiability. Some of these proposals maintain completeness. A linear resolution deduction defined below is shown to be a refutation complete method for the satisfiability check ([30]).

Definition 13. *Given a set of clauses* Ψ, *the set of* **linear resolution deductions** *from* Ψ *is a set* $\mathsf{Deductions}(\Psi) = \{\Gamma_1, \ldots, \Gamma_m\}$ *where each* $\Gamma_l = \{\gamma_1, \ldots, \gamma_n\} \in \mathsf{Deductions}(\Psi)$ *is defined as follows:*

(1) *For each* $\gamma_k \in \Gamma_l$ *such that* $1 < k \leq n$, *either* γ_k *is obtained by resolution from* γ_i *and* γ_j *where* $i, j < k$ *or* $\gamma_k \in \Psi$ *and*
(2) *For each* γ_i *such that* $1 \leq i < n$, *there are* γ_k *and* γ_j *(*$i < k$ *and* $j < k$*) s.t.* γ_k *is obtained by resolution from* γ_i *and* γ_j
(3) *No* $\gamma_k \in \Gamma_l$ *is a tautology*

Each such Γ_l is called linear deduction of δ_n from Ψ. In the above definition, condition (1) ensures that the clauses of a linear deduction are generated by using elements from Ψ and applying resolution recursively. Condition (2) ensures that every clause in the deduction is used to resolve with some other clause and condition (3) ensures that no tautologies will appear in the deduction.

Linear resolution is known to be a refutation complete and sound strategy for automated theorem proving. This is restated in the following proposition.

Proposition 1. *Let* Ψ *be a set of clauses and* $\alpha = a_1 \vee \ldots \vee a_n$ *be a clause. Then,* $\Psi \vdash \alpha$ *iff there is a linear deduction* $\Gamma \in \mathsf{Deductions}(\Psi \cup \{\overline{a_1}, \ldots, \overline{a_n}\})$, $\Gamma = \{\gamma_1, \ldots, \gamma_n\}$ *such that* γ_n *is the empty clause.*

Example 13. *Let* $\Psi = \{a \vee b \vee c, \neg c \vee d, \neg d \vee \neg c\}$ *and* $\alpha = a \vee b \vee e$. *Then* $\Psi \cup \{\overline{a}, \overline{b}, \overline{e}\}) = \{a \vee b \vee c, \neg c \vee d, \neg d \vee \neg c, \neg a, \neg b, \neg e\}$. *Then there is* $\Gamma \in \mathsf{Deductions}(\Psi \cup \{\overline{a}, \overline{b}, \overline{e}\})$ *with* $\Gamma = \{\neg d \vee \neg c, \neg c \vee d, \neg c, a \vee b \vee c, a \vee b, \neg a, b, \neg b, \bot\}$ *and it holds that* $\Psi \vdash \alpha$.

Proposition 2. *Let* α *be a tautology. Then,* $\langle \Phi, \alpha \rangle$ *is an argument iff* $\Phi = \emptyset$.

Hence, by proposition 2 the only set that can be a support for an argument where the claim is a tautology is the empty set. From now on we only consider claims that are not tautologies.

4. Connection Graphs

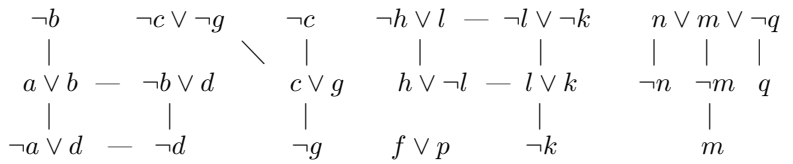
In this section we review a proposal for using connection graph to reduce the search space when looking for arguments [17]. We start by introducing some types of graphs where each node is represented by a clause and the links between each pair of clauses are determined according to the binary relations $\mathsf{Preattacks}$ and $\mathsf{Attacks}$ defined earlier.

4.1. Types of connection graph

For a knowledgebase Δ we define graphs where each node is a clause from Δ and each arc connects a pair of clauses. The **connection graph** for Δ is a graph (N, A) where N is a set of nodes each of which corresponds to a clause from Δ and A is the set of arcs that connect pairs of clauses with complementary literals. The **attack graph** for Δ is a graph where N is a set of nodes each of which corresponds to a clause from Δ and A is the set of arcs that connect pairs of clauses with exactly one complementary literal between them. The **closed graph** for Δ is the subgraph of the attack graph where for each clause ϕ in the subgraph and for each disjunct b in ϕ there is another clause ψ in the subgraph such that $\mathsf{Attacks}(\phi, \psi) = b$ holds. The formal definitions are given below along with examples taken from [17].

Definition 14. *Let* Δ *be a clause knowledgebase. The* **connection graph** *for* Δ*, denoted* $\mathsf{Connect}(\Delta)$*, is a graph* (N, A) *where* $N = \Delta$ *and* $A = \{(\phi, \psi) \mid \mathsf{Preattacks}(\phi, \psi) \neq \emptyset\}$.

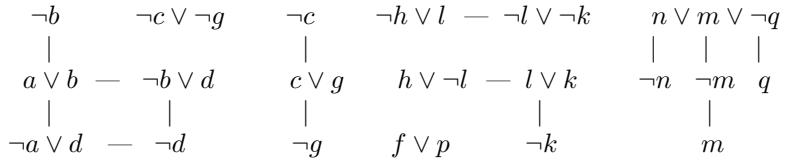
Example 14. *The following is the connection graph for* $\Delta = \{\neg b, \neg c \vee \neg g, \neg c, f \vee p, \neg l \vee \neg k, a \vee b, \neg b \vee d, c \vee g, \neg h \vee l, l \vee k, \neg a \vee d, \neg d, \neg g, h \vee \neg l, \neg k, n \vee m \vee \neg q, \neg m, \neg n, m, q\}$



The attack graph defined below is a subgraph of the connection graph identified using the Attacks function.

Definition 15. *Let Δ be a clause knowledgebase. The* **attack graph** *for Δ, denoted* $\mathsf{AttackGraph}(\Delta)$*, is a graph (N, A) where $N = \Delta$ and $A = \{(\phi, \psi) \mid \mathsf{Attacks}(\phi, \psi) \neq null\}$.*

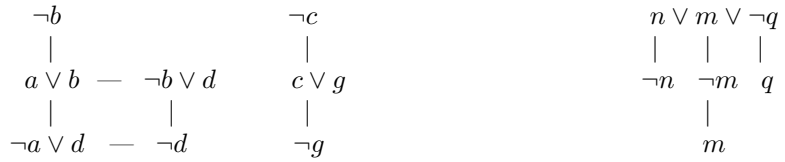
Example 15. *Continuing Example 14, the following is the attack graph for Δ.*



The following definition of closed graph introduces a kind of connected subgraph of the attack graph where for each clause ϕ in the subgraph and for each disjunct b in ϕ there is another clause ψ in the subgraph such that $\mathsf{Attacks}(\phi, \psi) = b$.

Definition 16. *Let Δ be a clause knowledgebase. The* **closed graph** *for Δ, denoted* $\mathsf{Closed}(\Delta)$*, is the largest subgraph (N, A) of* $\mathsf{AttackGraph}(\Delta)$*, such that for each $\phi \in N$, for each $b \in \mathsf{Disjuncts}(\phi)$ there is a $\psi \in N$ with* $\mathsf{Attacks}(\phi, \psi) = b$*.*

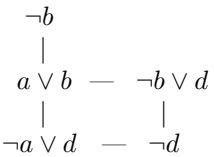
Example 16. *Continuing Example 15, the following is the closed graph for Δ.*



The **focal graph** (defined next) is a subgraph of the closed graph for Δ which is specified by a clause ϕ from Δ and corresponds to the part of the closed graph that contains ϕ. In the following, we assume a component of a graph means that each node in the component is connected to any other node in the component by a path.

Definition 17. *Let Δ be a clause knowledgebase and ϕ be a clause in Δ which we call the* **epicentre***. The* **focal graph** *of ϕ in Δ denoted* $\mathsf{Focal}(\Delta, \phi)$ *is defined as follows: If there is a component X in* $\mathsf{Closed}(\Delta)$ *containing the node ϕ, then* $\mathsf{Focal}(\Delta, \phi) = X$*, otherwise* $\mathsf{Focal}(\Delta, \phi)$ *is the empty graph.*

Example 17. *Continuing Example 16, the following is the focal graph of $\neg b$ in Δ*

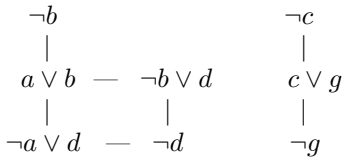


The last example illustrates how the notion of the focal graph of an epicentre ϕ in Δ can be used in order to focus on the part of the knowledgebase that is relevant to ϕ. The focal graph can be used to reduce the search space when looking for arguments for a clause α from a knowledgebase Δ of propositional clauses. This requires using each of the conjuncts of $\neg\alpha$ as the epicentre for a focal graph in Δ. All these focal graphs determined by the conjuncts of $\neg\alpha$ together make the components of a graph that we call the **query graph** of α in Δ. The set of clauses that appear as nodes in the query graph contains all the necessary formulae from Δ that can be supports for arguments for α. The formal definition of the query graph is given below. For a connection graph, function Nodes returns the nodes of a graph.

Definition 18. *Let Δ be a set of clauses and α be a clause. The* **query graph** *of α in Δ denoted $\mathsf{Query}(\Delta, \alpha)$ is the attack graph for the nodes*

$$\bigcup_{a_j \in \mathsf{Disjuncts}(\alpha)} \mathsf{Nodes}(\mathsf{Focal}(\Delta \cup \{\overline{a}_i \mid a_i \in \mathsf{Disjuncts}(\alpha)\}, \overline{a}_j))$$

Example 18. *Let $\Gamma = \{\neg c \vee \neg g, f \vee p, \neg l \vee \neg k, a \vee b, \neg b \vee d, c \vee g, \neg h \vee l, l \vee k, \neg a \vee d, \neg d, \neg g, h \vee \neg l, n \vee m \vee \neg q, \neg m, \neg n, m, q\}$ and let $\alpha = a \vee b \vee k$. Then, $\Gamma = \Delta \setminus \{\neg b, \neg c, \neg k\}$ where Δ is the knowledgebase of examples 14,15 and 16 and $\Gamma' = \Gamma \cup \{\overline{a}_i \mid a_i \in \mathsf{Disjuncts}(\alpha)\} = \Gamma \cup \{\neg b, \neg c, \neg k\} = \Delta$. The query graph of α in Γ is presented below and consists of the components that are the focal graphs of $\neg a$ in Γ' and $\neg b$ in Γ', while the focal graph of $\neg k$ in Γ' is empty.*



In the next section we describe how the search for arguments can be focused on each of the components of the focal graph.

4.2. Theoretical results concerning connection graphs

We now present some theoretical results related to the definitions of section 4.1.

Proposition 3. *Let Ψ be a minimal inconsistent set of clauses and let $(N, A) = \mathsf{Closed}(\Psi)$. Then $N = \Psi$.*

As a consequence of proposition 3, we obtain proposition 4 that follows.

Proposition 4. *Let $\langle \Phi, \alpha \rangle$ be an argument where α is a clause. If $\mathsf{Query}(\Delta, \alpha)$ is such that $\mathsf{Nodes}(\mathsf{Query}(\Delta, \alpha)) \neq \emptyset$, then $\Phi \subset \mathsf{Nodes}(\mathsf{Query}(\Delta, \alpha))$.*

According to proposition 4, all the supports for arguments for α are contained in the query graph of α in Δ. Hence, searching for arguments in the subset of Δ that corresponds to $\mathsf{Query}(\Delta, \alpha)$ instead of Δ provides a reduced search space without affecting completeness.

Corollary 1. *Let Ψ be a minimal inconsistent set of clauses. Then for all $\gamma \in \Psi$, $\mathsf{Focal}(\Psi, \gamma)$ is non-empty.*

5. Proof trees for arguments

In this section we describe how the structure of the query graph of α in Δ can be used in the search for arguments for α.

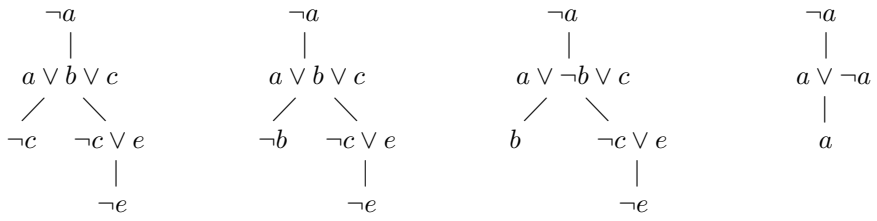
5.1. Definitions for proof trees for arguments

In [16] we defined a presuppport tree for a literal claim α to be a tree structure representing the steps of the search process for an argument for α where $\overline{\alpha}$ represents the root. We now re-define the presupport tree for a claim α that is a clause consisting of one or more disjuncts. Again, the root of the tree is assigned to a literal, which is one of the epicentres of the focal graph of α in Δ.

Definition 19. *Let Δ be a clause knowledgebase and let $\alpha = a_1 \vee \ldots \vee a_n$ be a clause that is not a tautology. A* **presupport tree** *for Δ, α and $a_k \in \mathsf{Disjuncts}(\alpha)$ is a tuple (N, A, f) where (N, A, f) is a tree, and f is a mapping from N to $\Delta \cup \{\overline{a_k}\}$ such that*

(1) *if x is the root of the tree, then $f(x) = \overline{a_k}$ and*
there is exactly one child y of x s.t. $\mathsf{Attacks}(f(y), f(x)) = a_k$,
(2) *for any nodes x, y in the same branch, if $x \neq y$, then $f(x) \neq f(y)$*
(3) *for any nodes x, y in the same branch, if x is the parent of y,*
then $\mathsf{Attacks}(f(x), f(y)) \neq null$

Example 19. *Some presupport trees for $\Delta = \{a \vee b \vee c, a \vee \neg b \vee c, \neg c, b, \neg c \vee e, \neg e, \neg a \vee c, \neg b, a \vee \neg a, a, a \vee c \vee \neg c, c\}$, $\alpha = a \vee b \vee d$ and a are:*



We now define some functions that we use throughout this paper in order to refer to the components of a presupport tree (N, A, f).

Definition 20. *Let Δ be a clause knowledgebase, α be a clause and (N, A, f) be a presupport tree for Δ, α and $a \in \mathsf{Disjuncts}(\alpha)$. Then for a node $x \in N$, $\mathsf{Ancestors}(x)$ is the set of ancestors of x in (N, A, f) (i.e. the set of nodes on the path from the parent of x to the root), and $\mathsf{AncestorLabels}(x)$ is the set of literals that define the arcs between the ancestors of x through the $\mathsf{Attacks}$ function where the sign of each literal is defined from the child to the parent: $\mathsf{AncestorLabels}(x) = \{\mathsf{Attacks}(f(w), f(w')) \mid w' \in \mathsf{Ancestors}(x)$ and $w \in \mathsf{Ancestors}(x) \cup \{x\}\}$. $\mathsf{Subtree}(x)$ is the set of successors of x in (N, A, f), together with x and $\mathsf{Children}(x)$ is the set of children nodes of x: $\mathsf{Children}(x) = \{y \mid (x, y) \in A$ and $y \in \mathsf{Subtree}(x)\}$. A branch is a set of nodes X connected through a sequence of arcs starting from the root node and ending on a leaf node.*

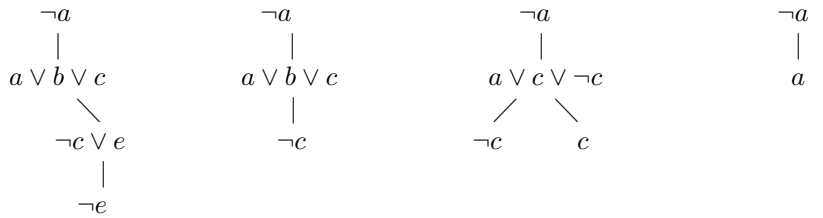
The presupport tree itself does not necessarily indicate a proof for α. If additional restrictions are applied on the way the nodes are arranged on the tree then we get a category of presupport trees that can be used to find a proof for α.

Definition 21. *Let Δ be a clause knowledgebase and let $\alpha = a_1 \vee \ldots \vee a_n$ be a clause. A* **complete presupport tree** *for Δ, α and $a_k \in \mathsf{Disjuncts}(\alpha)$ is a presupport tree (N, A, f) for Δ, α and a_k such that for any non-root $x \in N$, for every $b \in \mathsf{Disjuncts}(f(x))$ exactly one of the following conditions hold:*

i) $b \in \mathsf{Disjuncts}(\alpha) \setminus \{a_k\}$
ii) or there is exactly one arc (y, y') where $y' \in \mathsf{Ancestors}(x)$ such that $\mathsf{Attacks}(f(y), f(y')) = b$
iii) or there is exactly one $y \in \mathsf{Children}(x)$ s.t. $\mathsf{Attacks}(f(y), f(x)) = \overline{b}$

Example 20. *None of the presupport trees of example 19 is a complete presupport tree for Δ, α and a. In the first tree, for x with $f(x) = a \vee b \vee c$ and $c \in \mathsf{Disjuncts}(f(x))$ none of the conditions of definition 21 is satisfied. The second presupport tree is not complete because for node y with $f(y) = a \vee b \vee c$ and $b \in \mathsf{Disjuncts}(f(y))$ both conditions i) and iii) of definition 21 hold. The third presupport tree is not complete because for node z with $f(z) = b$ and $b \in \mathsf{Disjuncts}(f(z))$ both conditions i) and ii) of definition 21 hold. The last presupport tree is not complete because for w, with $f(w) = a$ and $a \in \mathsf{Disjuncts}(f(w))$, none of the conditions of the definition holds.*

Continuing example 19, some complete presupport trees for Δ, α and a are:



With the conditions of the last definition for a complete presupport tree, a unit clause consisting of a unique disjunct can represent a node of the tree only

as the root or a leaf node. Moreover, for a unit clause α, for a presupport tree for Δ, α and a where $\alpha = a$ conditions *ii*) and *iii*) of the above definition are sufficient to provide a complete presupport tree for Δ, α and a.

The idea in building a complete presupport tree (N, A, f) is that in this way the set of clauses produced ($\{f(x) \mid x \in N\}$) is such that for all $x \in N$ and for all $b \in \mathsf{Disjuncts}(f(x)) \setminus \mathsf{Disjuncts}(\alpha)$ there is a $y \in N$ such that $\overline{b} \in \mathsf{Disjuncts}(f(y))$ and the set $\{f(x) \mid x \in N\} \cup \{\neg\alpha\}$ is inconsistent. Apart from ensuring that for all the disjuncts of all $f(z)$ in the set there is a clause $f(y)$ containing their complement, the conditions of definition 21 help controlling the size of the tree. The fact that a branch is expanded below a node z by adding a node y such that $\mathsf{Attacks}(f(y), f(z)) = \overline{b}$ only when b does not appear in $\mathsf{AncestorLabels}(z)$, controls the length of the branches of the tree, and means that since this disjunct has been already dealt with by another node on the branch no additional nodes need to be added on the branch to deal with b. The fact that exactly one child of z is added per each such disjunct of $f(z)$ controls the width of the tree and means that every child of z deals with exactly one disjunct of $f(z)$ that has not been considered earlier on the branch.

From the way it is defined, the complete presupport tree has some properties that relate to its structure that are formalised in the next three propsitions. Proposition 5 suggests that apart from the root node of a complete presupport tree for Δ, α and a, no other complement of the disjuncts of α appears in the clauses that represent the tree.

Proposition 5. *Let (N, A, f) be a complete presupport tree for Δ, α and a. Then, there is no $\overline{b} \in \mathsf{Literals}(\{f(x) \mid x \in N\} \setminus \{\overline{a}\})$ such that $b \in \mathsf{Disjuncts}(\alpha)$.*

According to the next proposition a literal and its complement cannot label arcs on the same branch.

Proposition 6. *Let (N, A, f) be a complete presupport tree for Δ, α and a. Then, there is no $x \in N$ s.t. $b \in \mathsf{AncestorLabels}(x)$ and $\overline{b} \in \mathsf{AncestorLabels}(x)$.*

According to the next proposition, for a node x of a complete presupport tree (N, A, f), a literal from $\mathsf{Disjuncts}(f(x))$ cannot have its complement in $\mathsf{AncestorLabels}(x)$.

Proposition 7. *Let (N, A, f) be a complete presupport tree for Δ, α and a. Then, there is no $x \in N$ s.t. $b \in \mathsf{Disjuncts}(f(x))$ and $\overline{b} \in \mathsf{AncestorLabels}(x)$.*

With some additional conditions on the arcs of a complete presupport tree (N, A, f) for Δ, α and a, the corresponding set of clauses $\{f(x) \mid x \in N\} \setminus \{\overline{a}\} \cup \{\neg\alpha\}$ can be a minimal inconsistent set and so $\{f(x) \mid x \in N\} \setminus \{\overline{a}\}$ can be a support for an argument for α. These conditions are introduced in the definitions of a consistent presupport tree and a minimal presupport tree as follows.

Definition 22. *Let (N, A, f) be a complete presupport tree for Δ, α and a. Then (N, A, f) is a* **consistent presupport tree** *for Δ, α and a iff for any*

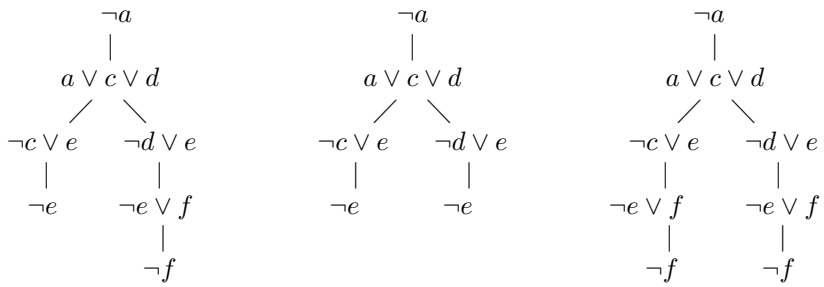
nodes x and y, if x' is the parent of x and y' is the parent of y, $\mathsf{Attacks}(f(x), f(x')) \neq \overline{\mathsf{Attacks}(f(y), f(y'))}$.

Example 21. *From the presupport trees of example 20 only the third is not consistent.*

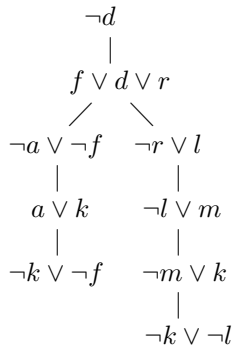
Definition 23. *Let (N, A, f) be a complete presupport tree for Δ, α and a. Then (N, A, f) is a* **minimal presupport tree** *for Δ, α and a iff there is no complete presupport tree (N', A', f') for Δ, α and a such that $\{f'(x') \mid x' \in N'\} \subset \{f(x) \mid x \in N\}$.*

Example 22. *All the complete presupport trees of example 20 are minimal.*

Example 23. *Let $\alpha = a \vee m$ and $\Delta = \{a \vee c \vee d, \neg b \vee b, \neg b, a \vee d \vee f, \neg e, \neg e \vee f, \neg f, \neg e, \neg d \vee e, \neg c \vee e, r \vee j, j \vee \neg s, \neg s \vee k, p\}$. In the following presupport trees for Δ, α and $a = a$, let (N_1, A_1, f_1) be the first presupport tree, (N_2, A_2, f_2) the second and (N_3, A_3, f_3) the third presupport tree. Then, (N_1, A_1, f_1) is not a minimal presupport tree because (N_2, A_2, f_2) and (N_3, A_3, f_3) are complete presupport trees and the sets of of clauses they consist of are contained in the set of clauses from (N_1, A_1, f_1).*



Example 24. *The following presupport tree for $\Delta = \{f \vee d \vee r, \neg a \vee \neg f, \neg r \vee l, a \vee k, \neg l \vee m, \neg k \vee \neg f, \neg k \vee \neg l, \neg m \vee k\}$, $\alpha = d \vee q$ and d is a minimal and consistent presupport tree.*



Checking a complete presupport tree (N, A, f) for Δ, α and a for minimality does not necessarily require testing whether each of the subsets of $\{f(x) \mid x \in N\}$ can produce a complete presupport tree for Δ, α and a. In section 5.3 we explain how checking some conditions related to the literals that define the arcs of a presupport tree can help in deciding whether the tree satisfies the definition for a minimal presupport tree.

Putting together all the definitions for proof trees given so far in this section we get the definition for a support tree for Δ, α and a that follows.

Definition 24. *A* **support tree** (N, A, f) *for* Δ, α *and* $a \in \mathsf{Disjuncts}(\alpha)$ *is s presupport tree* (N, A, f) *for* Δ, α *and* a *that is minimal and consistent.*

Example 25. *Proof trees* (N_2, A_2, f_2) *and* (N_3, A_3, f_3) *of example 23 are support trees. The proof tree of example 24 is also a support tree.*

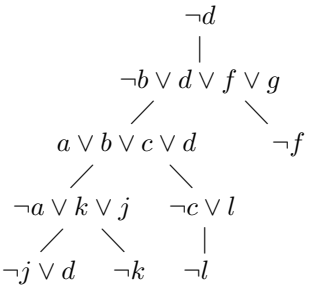
5.2. Theoretical results concerning proof trees

In this section we present some theoretical results associated with the definitions of section 5.1. We start by introducing function $\mathsf{SubtreeRes}(z)$ for a node z of a presupport tree (N, A, f), which we use in propositions that follow later.

Definition 25. *Let* (N, A, f) *be a complete presupport tree for* Δ, α *and* a. *For all* $z \in N$, *if* B *is the set of clauses that corresponds to the clauses that represent* $\mathsf{Subtree}(z)$ *(i.e.* $B = \{f(w) \mid w \in \mathsf{Subtree}(z)\}$*), then*

$$\mathsf{SubtreeRes}(z) = \bigvee \left\{ \left(\mathsf{Literals}(B) \setminus \{\mathsf{Attacks}(f(w), f(w')) \mid w, w' \in \mathsf{Subtree}(z)\} \right) \right\}$$

Example 26. *The following is a support tree for* $\Delta = \{\neg b \vee d \vee f \vee g, a \vee b \vee c \vee d, \neg a \vee k \vee j, \neg j \vee d, \neg k, \neg c \vee l, \neg l, \neg f, \neg d \vee b \vee g, \neg g \vee b, \neg b, \neg d \vee \neg j, j, \neg g, c \vee l\}$, $\alpha = d \vee m \vee g$ *and* d.



For the subtree rooted at z *with* $f(z) = a \vee b \vee c \vee d$, $B = \{f(x) \mid x \in \mathsf{Subtree}(z)\} = \{a \vee b \vee c \vee d, \neg a \vee k \vee j, \neg c \vee l, \neg j \vee d, \neg k, \neg l\}$. *Then,* $\mathsf{Literals}(B) = \{a, b, c, d, \neg a,\ k, j, \neg c, l, \neg j, \neg k, \neg l\}$, $\{\mathsf{Attacks}(f(w), f(w')) \mid w, w' \in \mathsf{Subtree}(z)\} = \{\neg a, a, \neg c, c, \neg j, j, \neg k, k, \neg l, l\}$. $\mathsf{Literals}(B) \setminus \{\mathsf{Attacks}(f(w), f(w')) \mid w, w' \in \mathsf{Subtree}(z)\} = \{b, d\}$ *and* $\mathsf{SubtreeRes}(z) = \bigvee\{b, d\} = b \vee d$.

Proposition 8. *Let (N, A, f) be a complete presupport tree for Δ, α and a. Then for a node $z \in N$, where z' is the parent of z,* $\mathsf{Attacks}(f(z), f(z')) = \mathsf{Attacks}(\mathsf{SubtreeRes}(z), f(z'))$.

Essentially, for a node z, $\mathsf{SubtreeRes}(z)$ gives a formula at z that is obtained by resolving the formula $f(z)$ with $\mathsf{SubtreeRes}(x_1), \ldots, \mathsf{SubtreeRes}(x_n)$ where $x_1, \ldots, x_n$ are the children of z. In this way, we use $\mathsf{SubtreeRes}$ to propagate resolution up the tree.

Proposition 9. *Let (N, A, f) be a complete presupport tree for Δ, α and a. Then for a node $z \in N$ with* $\mathsf{Children}(z) = \{x_1, \ldots, x_n\}$,

$$\mathsf{SubtreeRes}(z) = f(z) \bullet \mathsf{SubtreeRes}(x_1) \bullet \ldots \bullet \mathsf{SubtreeRes}(x_n)$$

From proposition 9 and the way function $\mathsf{SubtreeRes}$ is defined, follows proposition 10, according to which, if (N, A, f) is a complete presupport tree for Δ, α and a, then for all the nodes z of (N, A, f) there is a deduction of $\mathsf{SubtreeRes}(z)$ from the set of clauses assigned to the $\mathsf{Subtree}(z)$.

Proposition 10. *Let (N, A, f) be a complete presupport tree for Δ, α and a. Then for a node $z \in N$, there is a linear deduction $\{\delta_1, ..., \delta_n\} \in \mathsf{Deductions}(\{f(w) \mid w \in \mathsf{Subtree}(z)\})$ where $\mathsf{SubtreeRes}(z) \equiv \delta_n$ and $\{f(w) \mid w \in \mathsf{Subtree}(z)\} \subseteq \{\delta_1, ..., \delta_n\}$.*

Corollary 2. *Let (N, A, f) be a complete presupport tree for Δ, α and a. Then for a node $z \in N$, $\{f(w) \mid w \in \mathsf{Subtree}(z)\} \vdash \mathsf{SubtreeRes}(z)$.*

Using the properties of $\mathsf{SubtreeRes}$ for the nodes of a complete presupport tree (N, A, f) for Δ, α and a we prove some propositions that we then use in order to show how support trees can be used for finding supports for arguments for α.

According to the next proposition, the set of clauses that represent a subtree of a support tree rooted at a node z cannot prove a clause stronger than $\mathsf{SubtreeRes}(z)$. This is a key proposition for showing the consistency of the proof for α indicated by (N, A, f).

Proposition 11. *Let (N, A, f) be a support tree for Δ, α and a. Then for all $z \in N$, there is no $\gamma' \in \mathcal{C}$ with* $\mathsf{Disjuncts}(\gamma') \subset \mathsf{Disjuncts}(\mathsf{SubtreeRes}(z))$ *and* $\{f(w) \mid w \in \mathsf{Subtree}(z)\} \vdash \gamma'$.

Example 27. *Continuing example 26, for $z = a \vee b \vee c \vee d$,* $\mathsf{SubtreeRes}(z) = b \vee d$ *and for $\gamma' = b$ and $\gamma'' = d$, $\{f(x) \mid x \in \mathsf{Subtree}(z)\} \nvdash \gamma'$ and $\{f(x) \mid x \in \mathsf{Subtree}(z)\} \nvdash \gamma''$.*

The proposition that follows illustrates the completeness of our proposal for using support trees in order to obtain arguments for α from Δ.

Proposition 12. *Let $\langle \Phi, \alpha \rangle$ be an argument. Then, there is a support tree (N, A, f) for Δ, α and some $a \in \mathsf{Disjuncts}(\alpha)$ such that $\Phi = \{f(x) \mid x \in N\} \backslash \{\overline{a}\}$.*

The converse of proposition 12 also holds, if (N, A, f) is a support tree for Δ, α and some $a \in \mathsf{Disjuncts}(\alpha)$ and Φ is such that $\Phi = \{f(x) \mid x \in N\} \setminus \{\overline{a}\}$, then $\langle \Phi, \alpha \rangle$ is an argument. This result is presented in the proposition that follows which illustrates the soundness of our proposal of using support trees for Δ, α and a in order to obtain arguments for α from Δ.

Proposition 13. *Let Δ be a set of clauses and α be a clause and let (N, A, f) be a support tree for Δ, α and some $a \in \mathsf{Disjuncts}(\alpha)$. Then, $\langle \Phi, \alpha \rangle$ with $\Phi = \{f(x) \mid x \in N\} \setminus \{\overline{a}\}$ is an argument.*

So, according to proposition 13, if a support tree (N, A, f) is retrieved for Δ, α and some $a \in \mathsf{Disjuncts}(\alpha)$ then an argument $\langle \Phi, \alpha \rangle$ is retrieved for α where Φ is the set of clauses that represent the non-root nodes of (N, A, f). So, proposition 13 together with proposition 12 mean that we can have a sound and complete mechanism for generating arguments that is based on support trees.

The fact that we can use different disjuncts of α to determine the root of a support tree does not require comparing results of search based on different disjuncts of α. This is captured in the next proposition.

Proposition 14. *Let (N, A, f) be a support tree for Δ, α and a. Then, there is no support tree (N', A', f') for Δ, α and some $b \in \mathsf{Disjuncts}(\alpha)$ such that $\{f'(x') \mid x' \in N'\} \setminus \{\overline{b}\} \subset \{f(x) \mid x \in N\} \setminus \{\overline{a}\}$.*

Corollary 3. *Let Δ be a set of clauses α and be a clause. $\langle \Phi, \alpha \rangle$ is an argument iff there is a support tree (N, A, f) for Δ, α and some $a \in \mathsf{Disjuncts}(\alpha)$ such that $\Phi = \{f(x) \mid x \in N\} \setminus \{\overline{a}\}$.*

Hence, according to corollary 3, in order to find all the arguments for α, it is sufficient to find all the support trees for Δ, α and a_i, for all $a_i \in \mathsf{Disjuncts}(\alpha)$. In the next section we motivate the use of this mechanism by explaining how the structure of a presupport tree can be used in order to check for minimality.

5.3. The minimality check

As it was explained earlier, the conditions for a complete presupport tree for Δ, α and a ensure that a proof for α that contains a limited number of clauses is produced. Let (N, A, f) be a complete presupport tree for Δ, α and a. For each node $z \in N$, for each $b \in \mathsf{Disjuncts}(f(z)) \setminus \mathsf{Disjuncts}(\alpha)$ either there is exactly one arc (y, y') such that $y' \in \mathsf{Ancestors}(z)$ and $\mathsf{Attacks}(f(y), f(y')) = b$, and hence there is $y' \in \mathsf{Ancestors}(z)$ with $\overline{b} \in \mathsf{Disjuncts}(f(y'))$, or there is exactly one node $y \in \mathsf{Children}(z)$ such that $\mathsf{Attacks}(f(y), f(z)) = \overline{b}$. This way redundant nodes to resolve with disjunct b of $f(z)$ will be avoided on the branch where z belongs. Avoiding in this way redundant resolution steps though does not necessarily ensure the minimality of the proof indicated by a complete presupport tree. To obtain a minimal and consistent proof for α we need to retrieve a complete presupport tree for Δ, α and a that satisfies the conditions for a minimal and consistent presupport tree. To check whether a complete presupport tree (N, A, f) satisfies the definition for a consistent presupport tree we can simply

check whether there are any arcs $(x, x'), (y, y')$ in A where x' is the parent of x and y' is the parent of y such that $\mathsf{Attacks}(f(x), f(x')) = \overline{\mathsf{Attacks}(f(y), f(y'))}$. To check whether (N, A, f) satisfies the definition for a minimal presupport tree, definition 23 would suggest testing whether subsets of the set of clauses that represent the nodes of (N, A, f) can be used in a complete presupport tree for Δ, α and some $a_i \in \mathsf{Disjuncts}(\alpha)$. In fact the structure of a consistent preupport tree can help deciding whether this indicates a minimal proof for α. So given a consistent presupport tree for Δ, α and a we can decide whether it is minimal by checking some of the properties of the tree.

A complete presupport tree corresponds with a concise linear deduction D from the set of clauses Φ assigned to the non-root nodes. If $\Phi = \{f(x) \mid x \in N\} \setminus \{\overline{a}\}$, then $D \in \mathsf{Deductions}(\Phi)$ where $D = \{\delta_1, \ldots, \delta_n\}$ is such that $\mathsf{Disjuncts}(\delta_n) \subseteq \mathsf{Disjuncts}(\alpha)$ and $\Phi \subseteq D$. D is composed of a limited number of steps, each associated with a node of (N, A, f). Each step uses the preceding steps to calculate $\mathsf{SubtreeRes}(z)$ for a node $z \in N$ and then adds this as a new element of the deduction. With this construction, if z is the child of the root node, then this is the last step of the deduction and $\mathsf{SubtreeRes}(z) = \delta_n$. According to proposition 9, for a node $z \in N$ with $\mathsf{Children}(z) = \{x_1, \ldots, x_n\}$, it holds that $\mathsf{SubtreeRes}(z) = f(z) \bullet \mathsf{SubtreeRes}(x_1) \bullet \ldots \bullet \mathsf{SubtreeRes}(x_n)$. So, each such step can be regarded as a 'sub-deduction' D_z of D that contributes in obtaining δ_n and is obtained from the set of clauses in $\mathsf{Subtree}(z)$: $D_z = \{\gamma_1, \ldots, \gamma_z\}$ where $\gamma_z = \mathsf{SubtreeRes}(z)$, $D_z \in \mathsf{Deductions}(\{f(w) \mid w \in \mathsf{Subtree}(z)\})$ and $\{f(w) \mid w \in \mathsf{Subtree}(z)\} \subseteq D_z$.

To check whether Φ is minimal for entailing α we can check whether some clauses that contribute to each resolution step can be omitted and as a result get a deduction $D' = \{\delta'_1, \ldots, \delta'_m\}$ such that $D' \in \mathsf{Deductions}(\Phi')$ for some $\Phi' \subset \Phi$ and $\mathsf{Disjuncts}(\delta'_m) \subseteq \mathsf{Disjuncts}(\alpha)$. To decide whether it is possible for this to happen we can look at the presupport tree and examine whether for some node x, there is a node y that plays the same role as x in D and so sub-deduction $D_y \subset D$ can be removed from D and D_x can be used instead in its place. If this is the case, then the set $\Phi' = \{f(p) \mid p \in (N \setminus \mathsf{Subtree}(y))\} \setminus \{\overline{a}\}$ is sufficient to entail α. Because the same clauses can be assigned to several nodes of a presupport tree, it can be the case where $\{f(p) \mid p \in (N \setminus \mathsf{Subtree}(y))\} = \{f(q) \mid q \in N\}$. Then, $\Phi' = \Phi$ and the minimality condition is not affected. If it holds though that $\{f(p) \mid p \in (N \setminus \mathsf{Subtree}(y))\} \neq \{f(q) \mid q \in N\}$, then it means that $\{f(p) \mid p \in (N \setminus \mathsf{Subtree}(y))\} \subset \{f(q) \mid q \in N\}$ and since $\overline{a} \in \{f(p) \mid p \in (N \setminus \mathsf{Subtree}(y))\} \cap \{f(q) \mid q \in N\}$, the above can be re-written to $\{f(p) \mid p \in (N \setminus \mathsf{Subtree}(y))\} \setminus \{\overline{a}\} \subset \{f(q) \mid q \in N\} \setminus \{\overline{a}\}$ which is equivalent to $\Phi' \subset \Phi$. So, in the case where $\{f(p) \mid p \in (N \setminus \mathsf{Subtree}(y))\} \neq \{f(q) \mid q \in N\}$, it holds that there is a subset Φ' of Φ such that $\Phi' \vdash \alpha$. Then $\langle \Phi, \alpha \rangle$ is not an argument and by proposition 13 (N, A, f) is not a support tree so since (N, A, f) is a consistent presupport tree follows that it is not a minimal presupport tree. Hence in such a case where we investigate whether there is some redundancy in the proof for α caused by the steps that correspond to two nodes x, y, we need to see whether $\{f(p) \mid p \in (N \setminus \mathsf{Subtree}(y))\} \neq \{f(q) \mid q \in N\}$ holds. The next paragraph explains which pairs of nodes are likely to cause redundancy and thus need to

be investigated.

A pair of nodes x, y can have some overlap in their role in deduction D if $\mathsf{Disjuncts}(\mathsf{SubtreeRes}(x)) \cap \mathsf{Disjuncts}(\mathsf{SubtreeRes}(y)) \neq \emptyset$. This is because for a node x and $D_x \subseteq D$ as defined above, $\mathsf{SubtreeRes}(x)$ is the last element of D_x, and some $b \in \mathsf{Disjuncts}(\mathsf{SubtreeRes}(x))$ will be used to resolve with some other clause in a later step of the deduction. Hence, if this b appears in $\mathsf{SubtreeRes}(y)$ for some $y \in N$, it is possible for D_y to be subtracted from D and re-use D_x in its place. It can be the case though where $\mathsf{Disjuncts}(\mathsf{SubtreeRes}(x)) \cap \mathsf{Disjuncts}(\mathsf{SubtreeRes}(y)) \subseteq \mathsf{Disjuncts}(\alpha)$ and so the disjuncts that $\mathsf{SubtreeRes}(x)$ and $\mathsf{SubtreeRes}(y)$ have in common are only the ones that are in α. In this case there is no other $b \in \mathsf{Disjuncts}(\mathsf{SubtreeRes}(x)) \cap \mathsf{Disjuncts}(\mathsf{SubtreeRes}(y))$ and so $\mathsf{SubtreeRes}(x)$ is used in the deduction to resolve on disjuncts different than the ones $\mathsf{SubtreeRes}(y)$ is used for. Apart from $a \in \mathsf{Disjuncts}(\alpha)$ which indicates the clause $\overline{a}$ for the root of (N, A, f), the rest of the disjuncts $a_i \in \mathsf{Disjuncts}(\alpha) \cap \mathsf{Literals}(\Phi)$ do not play a role in the proof for α and can make it hard to compare the deduction steps of D. Function $\mathsf{Unresolved}$ defined below can be used to deal with this. For an $x \in N$, function $\mathsf{Unresolved}(x)$ gives the clause that consists of the disjuncts of $\mathsf{SubtreeRes}(x)$ excluding the ones that are in α.

Definition 26. *Let (N, A, f) be a complete presupport tree. For a node $x \in N$,*

$$\mathsf{Unresolved}(x) = \bigvee(\mathsf{Disjuncts}(\mathsf{SubtreeRes}(x)) \setminus \mathsf{Disjuncts}(\alpha))$$

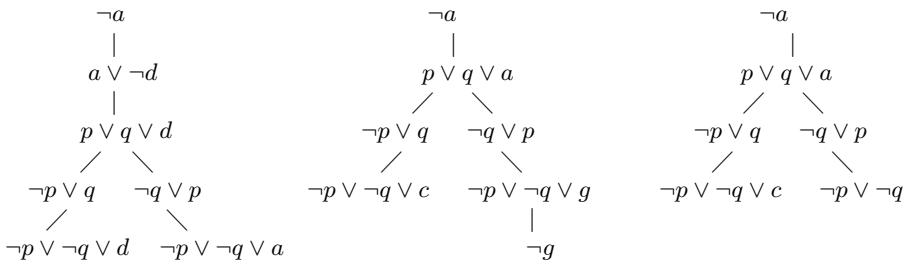
The idea in using function $\mathsf{Unresolved}(x)$ is that it indicates for a node x which literals need to be eliminated at this stage. Checking whether for a pair of nodes $x, y \in N$ there is a $b \in \mathsf{Disjuncts}(\mathsf{Unresolved}(x)) \cap \mathsf{Disjuncts}(\mathsf{Unresolved}(y))$ gives an indication that there is a possibility for D_x to be used instead of D_y in the proof for α or vice versa. So condition $\mathsf{Disjuncts}(\mathsf{Unresolved}(x)) \cap \mathsf{Disjuncts}(\mathsf{Unresolved}(y)) \neq \emptyset$ can be used to locate which pairs of nodes x, y need to be examined on whether they cause some redundancy in the proof that affects its minimality.

If some $b \in \mathsf{Disjuncts}(\mathsf{Unresolved}(x)) \cap \mathsf{Disjuncts}(\mathsf{Unresolved}(y))$, then replacing D_y by D_x in D can lead to a proof for α if for the rest of the disjuncts b_i of $\mathsf{Unresolved}(x)$ there are clauses in $\Phi \setminus \{f(p) \mid p \in (N \setminus \mathsf{Subtree}(y))\}$ that can eliminate b_i. From the way the complete presupport tree is built, if (N, A, f) satisfies the definition for a complete presupport tree, then for all $x \in N$, $\mathsf{Disjuncts}(\mathsf{Unresolved}(x)) \subseteq \mathsf{AncestorLabels}(x)$. If there is a branch on (N, A, f) other than the one where x belongs, where all the disjuncts of $\mathsf{Unresolved}(x)$ are used to label arcs, then it means that there are nodes w_i on this branch for which $\mathsf{Disjuncts}(\mathsf{Unresolved}(x)) \cap \mathsf{Disjuncts}(\mathsf{Unresolved}(w_i)) \neq \emptyset$. In particular, there is an arc (y, y') on that branch where y' is the parent of y s.t. $\mathsf{Attacks}(f(y), f(y')) \in \mathsf{Disjuncts}(\mathsf{Unresolved}(x))$ and $\mathsf{Disjuncts}(\mathsf{Unresolved}(x)) \subseteq \mathsf{AncestorLabels}(y)$. Then, D_x can replace D_y in D and $\Phi' = \{f(p) \mid p \in (N \setminus \mathsf{Subtree}(y))\}$ will be sufficient to provide a proof for α. For the parent y' of y, if $\mathsf{Children}(y') = \{y, y_0, \ldots, y_n\}$, then by proposition 9, $\mathsf{SubtreeRes}(y') = f(y')$ •

$\mathsf{SubtreeRes}(y)\bullet\mathsf{SubtreeRes}(y_0)\bullet\ldots\bullet\mathsf{SubtreeRes}(y_n)$. By replacing in this relation $\mathsf{SubtreeRes}(y)$ by $\mathsf{SubtreeRes}(x)$ we get a valid resolution step where no tautologies are involved. This way we obtain a linear deduction $D'_y = \{\gamma_1, \ldots, \gamma_k\}$ where $\gamma_k = f(y') \bullet \mathsf{SubtreeRes}(x) \bullet \mathsf{SubtreeRes}(y_0) \bullet \ldots \bullet \mathsf{SubtreeRes}(y_n)$ and it holds that $\mathsf{Disjuncts}(\gamma_k) \setminus \mathsf{Disjuncts}(\alpha) \subseteq \mathsf{Disjuncts}(\mathsf{SubtreeRes}(y'))$. Then $D'_y \in \mathsf{Deductions}(\Phi'_y)$ where $\Phi'_y = \{f(p) \mid p \in (\mathsf{Subtree}(y') \setminus \mathsf{Subtree}(y)) \cup \mathsf{Subtree}(x)\}$. Continuing the deduction from y by using D'_y we obtain a deduction $D' \in \mathsf{Deductions}(\Phi')$ where $\Phi' = \{f(p) \mid p \in (N \setminus \mathsf{Subtree}(y))\} \setminus \{\overline{a}\}$, and $D' = \{\delta'_1 \ldots \delta'_m\}$ is such that $\mathsf{Disjuncts}(\delta'_m) \subseteq \mathsf{Disjuncts}(\alpha)$. For a pair of nodes x, y such that $\{f(p) \mid p \in (N \setminus \mathsf{Subtree}(y))\} \neq \{f(q) \mid q \in N\}$ this would mean that there is a $\Phi' \subset \Phi$ such that $\Phi' \vdash \alpha$ and so (N, A, f) is a non-minimal presupport tree for Δ, α and a.

So, in order to decide whether a consistent presupport tree (N, A, f) is minimal, we need to check first if there are pairs of branches that apart from the arc (z, z') where z' is the root node, they have other arcs labeled by common literals. If they do not, then (N, A, f) is a minimal presupport tree and we do not need to investigate further. If they do, then we check whether there is a pair of nodes x, y such that $\mathsf{Attacks}(f(y), f(y')) \in \mathsf{Disjuncts}(\mathsf{Unresolved}(x))$ (where y' is the parent of y) and $\mathsf{Disjuncts}(\mathsf{Unresolved}(x)) \subseteq \mathsf{AncestorLabels}(y)$. If this holds and it also holds that $\{f(p) \mid p \in (N \setminus \mathsf{Subtree}(y))\} \neq \{f(q) \mid q \in N\}$ then this means that for $\Phi' = \{f(p) \mid p \in (N \setminus \mathsf{Subtree}(y))\} \setminus \{\overline{a}\}$ it holds that $\Phi' \subset \Phi$ and $\Phi' \vdash \alpha$ and so $\langle \Phi, \alpha \rangle$ is not an argument and (N, A, f) is not a support tree. Since we have assumed that (N, A, f) is a consistent presupport tree, then if (N, A, f) does not satisfy the definition for a support tree it means that (N, A, f) does not satisfy the definition for a minimal presupport tree for Δ, α and a.

Example 28. *Let* $\Delta = \{a \vee \neg d, p \vee q \vee d, p \vee q \vee a, \neg p \vee q, \neg q \vee p, \neg p \vee \neg q, \neg p \vee \neg q \vee a, \neg p \vee \neg q \vee d, \neg p \vee \neg q \vee c, \neg p \vee \neg q \vee g, \neg g\}$ *and* $\alpha = a \vee c$. *All the following consistent presupport trees for* Δ, α *and* a *are non-minimal.*

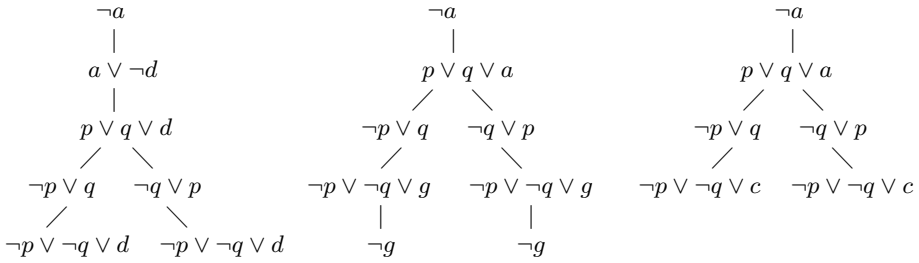


In the first tree, let x *be the node with* $f(x) = \neg p \vee \neg q \vee d$ *and* y *be the node with* $f(y) = \neg p \vee \neg q \vee a$. *Then,* $\mathsf{Unresolved}(x) = \neg p \vee \neg q \vee d$ *and* $\mathsf{Unresolved}(y) = \neg p \vee \neg q$ *and it holds that* $\mathsf{Attacks}(\neg p \vee \neg q \vee a, \neg q \vee p) \in \mathsf{Disjuncts}(\mathsf{Unresolved}(x))$ *and* $\mathsf{Disjuncts}(\mathsf{Unresolved}(x)) \subseteq \mathsf{AncestorLabels}(y)$ *and* $\{f(p) \mid p \in (N \setminus \mathsf{Subtree}(y))\} = \{\neg a, a \vee \neg d, p \vee q \vee d, \neg p \vee q, \neg q \vee p, \neg p \vee \neg q \vee d\} \neq \{f(q) \mid q \in N\}$.

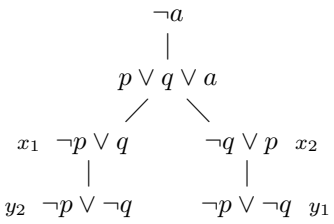
In the second tree, for x such that $f(x) = \neg p \vee \neg q \vee c$ and y such that $f(y) = \neg p \vee \neg q \vee g$, $\mathsf{Unresolved}(x) = \neg p \vee \neg q$ *and* $\mathsf{Unresolved}(y) = \neg p \vee \neg q$ *and it holds that* $\mathsf{Attacks}(\neg p \vee \neg q \vee g, \neg q \vee p) \in \mathsf{Disjuncts}(\mathsf{Unresolved}(x))$ *and* $\mathsf{Disjuncts}(\mathsf{Unresolved}(x)) \subseteq \mathsf{AncestorLabels}(y)$ *and* $\{f(p) \mid p \in (N \backslash \mathsf{Subtree}(y))\} = \{\neg a, p \vee q \vee a, \neg p \vee q, \neg q \vee p, \neg p \vee \neg q \vee c\} \neq \{f(q) \mid q \in N\}$.

In the third tree, for x such that $f(x) = \neg p \vee \neg q \vee c$ and y such that $f(y) = \neg p \vee \neg q$, $\mathsf{Unresolved}(x) = \neg p \vee \neg q$ *and* $\mathsf{Unresolved}(y) = \neg p \vee \neg q$ *and it holds that* $\mathsf{Attacks}(\neg p \vee \neg q, \neg q \vee p) \in \mathsf{Disjuncts}(\mathsf{Unresolved}(x))$ *and* $\mathsf{Disjuncts}(\mathsf{Unresolved}(x) \subseteq \mathsf{AncestorLabels}(y)$ *and* $\{f(p) \mid p \in (N \setminus \mathsf{Subtree}(y))\} = \{\neg a, p \vee q \vee a, \neg p \vee q, \neg q \vee p, \neg p \vee \neg q \vee c\} \neq \{f(q) \mid q \in N\}$.

All the following consistent presupport trees are minimal.



Example 29. *The following presupport tree for $\Delta = \{p \vee q \vee a, \neg p \vee q, \neg q \vee p, \neg p \vee \neg q\}$, $\alpha = a \vee c$ and a is a consistent and minimal presupport tree. For all the pairs of nodes x, y s.t.* $\mathsf{Attacks}(f(y), f(y')) \in \mathsf{Disjuncts}(\mathsf{Unresolved}(x))$ *(where y' is the parent of y) and* $\mathsf{Disjuncts}(\mathsf{Unresolved}(x)) \subseteq \mathsf{AncestorLabels}(y)$ *(i.e. the pairs $x = x_1, y = y_1$ or $x = x_2, y = y_2$ or $x = y_1, y = y_2$ or $x = y_2, y = y_1$) it holds that $\{f(p) \mid p \in (N \setminus \mathsf{Subtree}(y))\} = \{f(q) \mid q \in N\}$ so the minimality condition is not affected for (N, A, f).*



6. Search trees for undercuts

In this section we show how the definitions and theoretical results of section 5 can be extended to generate undercuts for an argument. First we describe how the properties of the language of clauses $\mathcal{C}$ can be taken into account to make the search for undercuts more efficient.

6.1. Reducing the search space for canonical undercuts

Let $A = \langle \Phi, \alpha \rangle$ be an argument where $\Phi = \{\phi_1, \ldots, \phi_n\}$ is a set of clauses and α is a clause. Finding counterarguments (i.e. canonical undercuts) for A requires finding arguments with $\neg \bigwedge \Phi$ as the claim. Taking into account that the clauses from Φ are connected through the $\mathsf{Attacks}$ relation we show that we can focus the search for counterarguments on a particular subset of the resolvents of Φ rather than on Φ. This is the set of strong resolvents defined next and has some interesting properties that can make the search for undercuts more effective.

Definition 27. *Let Φ be a set of clauses. Then the set of* **strong resolvents** *of Φ, denoted $\mathsf{SResolvents}(\Phi)$ is defined as follows.*

$$\mathsf{SResolvents}(\Phi) = \{\psi \in \mathsf{Resolvents}(\Phi) \mid \neg\exists \psi' \in \mathsf{Resolvents}(\Phi) \text{ s.t. } \psi' \neq \psi \text{ and } \mathsf{Disjuncts}(\psi') \subseteq \mathsf{Disjuncts}(\psi)\}$$

Example 30. *Let $\Phi = \{\neg e, e \vee \neg k, \neg j \vee k, \neg l \vee j \vee f\}$. Then, $\mathsf{Resolvents}(\Phi) = \{\neg e, e \vee \neg k, \neg j \vee k, \neg l \vee j \vee f, \neg k, e \vee \neg j, k \vee \neg l \vee f, \neg j, e \vee \neg l \vee f, \neg l \vee f\}$ and $\mathsf{SResolvents}(\Phi) = \{\neg e, \neg j, \neg k, \neg l \vee f\}$.*

Proposition 15. *For a set of clauses Φ, $\neg \bigwedge \Phi \equiv \neg \bigwedge \mathsf{SResolvents}(\Phi)$.*

Example 31. *Continuing example 30, $\mathsf{SResolvents}(\Phi) = \{\neg e, \neg j, \neg k, \neg l \vee f\}$ and $\neg \bigwedge \mathsf{SResolvents}(\Phi) = \neg\big(\neg e \wedge \neg j \wedge \neg k(\neg l \vee f)\big) = (e \vee k \vee j \vee l) \wedge (e \vee k \vee j \vee \neg f)$ which is the conjunctive normal form of $\neg\big(\neg e \wedge (e \vee \neg k) \wedge (\neg j \vee k) \wedge (\neg l \vee j \vee f)\big) = \neg \wedge \Phi$*

As a consequence of proposition 15 we obtain the following corollary.

Corollary 4. *Let Φ, Ψ be sets of clauses. Then $\langle \Psi, \neg \bigwedge \Phi \rangle$ is an argument iff $\langle \Psi, \neg \bigwedge \mathsf{SResolvents}(\Phi) \rangle$ is an argument.*

Hence, searching for a canonical undercut for an argument $\langle \Phi, \alpha \rangle$ is equivalent to searching for an argument for $\neg \bigwedge \mathsf{SResolvents}(\Phi)$. According to the following proposition we can omit in this search the clauses from Δ that are subsumed by some clause from Φ.

Proposition 16. *Let $A = \langle \Phi, \alpha \rangle$ be an argument and let $\langle \Psi, \diamond \rangle$ be a canonical undercut for A. Then, $\forall \psi \in \Psi$, $\forall \phi \in \Phi$, $\mathsf{Disjuncts}(\phi) \not\subseteq \mathsf{Disjuncts}(\psi)$.*

As a consequence of proposition 16 we obtain the following corollary.

Corollary 5. *Let $\langle \Phi, \alpha \rangle$ be an argument and let $\langle \Psi, \diamond \rangle$ be a canonical undercut for $\langle \Phi, \alpha \rangle$. Then there is no $\phi \in \Phi$, such that $\phi \in \Psi$.*

Moreover, as a consequence of proposition 16 we obtain the following corollary.

Corollary 6. *Let $A = \langle \Phi, \alpha \rangle$ be a argument and let $\langle \Psi, \diamond \rangle$ be a canonical undercut for A. Then, $\forall \psi \in \Psi$, $\forall \rho \in \mathsf{SResolvents}(\Phi)$, $\mathsf{Disjuncts}(\rho) \not\subseteq \mathsf{Disjuncts}(\psi)$.*

So, according to corollary 6, when looking for canonical undercuts for an argument $\langle \Phi, \alpha \rangle$ we can remove from the knowledgebase the clauses that are subsumed by some clause from $\mathsf{SResolvents}(\Phi)$ since these cannot be in the premises for a canonical undercut $\langle \Psi, \diamond \rangle$ for $\langle \Phi, \alpha \rangle$.

Corollary 7. *Let $\langle \Phi, \alpha \rangle$ be an argument and $\langle \Psi, \diamond \rangle$ be a canonical undercut for $\langle \Phi, \alpha \rangle$. Then, if $\Delta' = \{\delta \in \Delta \mid \exists \rho \in \mathsf{SResolvents}(\Phi)$ s.t. $\rho \vdash \delta\}$ it holds that $\Psi \subseteq \Delta \setminus \Delta'$.*

We now describe how we can use the results above in the search for canonical undercuts. The definition for a canonical undercut suggests that generating canonical undercuts for an argument $\langle \Phi, \alpha \rangle$ (i.e. finding arguments with claim $\neg\wedge\Phi$) requires converting $\neg\wedge\Phi$ to its CNF $\overline{\Phi}$ and find arguments for $\overline{\Phi}$. However, conversion to CNF would be an inefficient way to deal with the problem as the size of CNF can be exponential to the size of the original clause.

As an alternative we can find the sets of all arguments A_i for each $\neg\rho_i$, $\rho_i \in \mathsf{SResolvents}(\Phi)$ using $(\Delta \setminus \Phi) \cup \mathsf{SResolvents}(\Phi)$ as the knowledgebase. Because $\bigwedge \mathsf{SResolvents}(\Phi)$ is equivalent to $\bigwedge \Phi$, if for some $\Psi \subset \Delta$, $\Psi \cup \{\bigwedge \mathsf{SResolvents}(\Phi)\}$ is a minimal inconsistent set then $\Psi \cup \{\bigwedge \Phi\}$ is also a minimal inconsistent set. Since $\Psi \cup \{\bigwedge \mathsf{SResolvents}(\Phi)\}$ is a minimal inconsistent set then there is a $\Gamma'_i \subseteq \mathsf{SResolvents}(\Phi)$ such that $\Psi \cup \Gamma'_i$ is a minimal inconsistent set and for some $\rho_i \in \Gamma'_i$, if $\Gamma_i = \Gamma'_i \setminus \{\rho_i\}$, then $\langle \Psi \cup \Gamma_i, \neg\rho_i \rangle$ is an argument. Moreover, because by corollary 5 for all the canonical undercuts $\langle \Psi, \diamond \rangle$ for $\langle \Phi, \alpha \rangle$, $\Phi \cap \Psi = \emptyset$, then $(\Delta \setminus \Phi) \cup \mathsf{SResolvents}(\Phi)$ can be used as the knowledgebase to search for a set Ψ that together with $\bigwedge \mathsf{SResolvents}(\Phi)$ is a minimal inconsistent set.

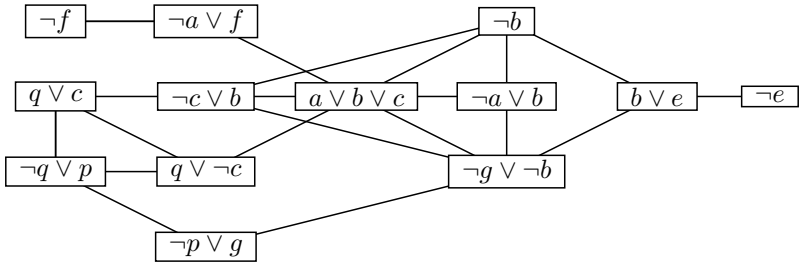
Using an element ρ_i of $\mathsf{SResolvents}(\Phi)$ to find arguments for $\neg\rho_i$ from $(\Delta \setminus \Phi) \cup \mathsf{SResolvents}(\Phi)$ helps reduce the number of non-minimal proofs for $\neg \wedge \Phi$ that may be produced during the search. The fact that the premises in Φ are clauses linked with each other by resolution can be used when looking for canonical undercuts $\langle \Psi, \diamond \rangle$ for $\langle \Phi, \alpha \rangle$ to avoid using premises in Ψ that are subsumed in Φ but are not possible to detect before producing the set $\mathsf{SResolvents}(\Phi)$. For example, assume we are looking for canonical undercuts for $\langle \Phi, \alpha \rangle = \langle \{a \vee b \vee c, \neg c \vee b, \neg b\}, a \vee d \rangle$, hence we are looking for arguments for $\neg\big((a \vee b \vee c) \wedge (\neg c \vee b) \wedge \neg b\big)$. It holds that $\mathsf{SResolvents}(\Phi) = \{a, \neg c, \neg b\}$ so in order to find all the arguments for $\neg\big((a \vee b \vee c) \wedge (\neg c \vee b) \wedge \neg b\big)$ we can alternatively try to find all the arguments for $\neg\big(a \wedge \neg c \wedge \neg b\big)$. For this we can use the negation $\neg\rho_i$ of one of the elements ρ_i of $\mathsf{SResolvents}(\Phi)$ and try to find arguments for $\neg\rho_i$. Not all the arguments for these $\neg\rho_i$ are necessarily canonical undercuts for $\langle \Phi, \alpha \rangle$, but all the canonical undercuts for $\langle \Phi, \alpha \rangle$ are retrieved this way. Although this method generates non-canonical undercuts besides canonical undercuts, it has the advantages that it does not require converting $\neg \wedge \Phi$ to its CNF and that it narrows down the search to what actually needs to be proved. Later we describe how we decide which of these arguments for $\neg\rho_i$ are canonical undercuts.

Besides the benefits mentioned above, using this method when looking for canonical undercuts for $\langle \Phi, \alpha \rangle$ can help reduce the cardinality of the knowledgebase. Any clause $\delta \in \Delta$ that is subsumed by some $\rho \in \mathsf{SResolvents}(\Phi)$ can be removed from the knowledgebase, since by corollary 6, δ cannot be included in the premises of a canonical undercut for $\langle \Phi, \alpha \rangle$. For instance, back to the example where we are looking for canonical undercuts for $\langle \Phi, \alpha \rangle = \langle \{a \vee b \vee c, \neg c \vee b, \neg b\}, a \vee d \rangle$, if we use the negation of $a \in \mathsf{SResolvents}(\Phi)$ as the claim for an argument, then $\Psi = \{\neg a \vee g, \neg g \vee \neg b, b \vee e, \neg e\}$ is a support for an argument for $\neg a$ but it is not a support for a canonical undercut for $\langle \Phi, \alpha \rangle$ because $\Psi' = \{b \vee e, \neg e\}$ is a subset of Ψ sufficient to entail $\neg \wedge \Phi$. Removing $\neg g \vee \neg b$ from Δ because it is subsumed in $\neg b$ where $\neg b \in \mathsf{SResolvents}(\Phi)$ means Ψ cannot be retrieved during the search for canonical undercuts for $\langle \Phi, \alpha \rangle$.

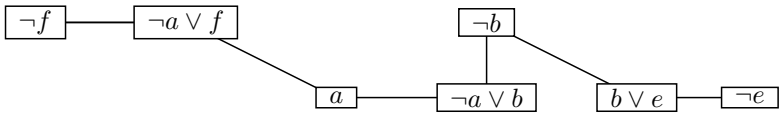
Apart from the fact that removing from Δ the clauses that are subsumed by some clause from $\mathsf{SResolvents}(\Phi)$ can reduce the cardinality of the background knowledge, it can result in a substantially reduced closed graph as the search space for canonical undercuts. Let $\Delta' = \{\delta \in \Delta \mid \exists \rho \in \mathsf{SResolvents}(\Phi) \text{ s.t. } \rho \vdash \delta\}$. Then, by corollary 7, $\Delta \setminus \Delta'$ contains the premises for all the canonical undercuts for $\langle \Phi, \alpha \rangle$. The clauses from $\Delta \setminus \Delta'$ can be linked in the closed graph for Δ to a number of clauses from Δ' that would not satisfy the connectivity condition for belonging to $\mathsf{Closed}(\Delta)$ if it had not been for the clauses of Δ'. As an effect, $\mathsf{Closed}((\Delta \setminus \Delta') \cup \mathsf{SResolvents}(\Phi))$ can contain a smaller number of clauses than $\mathsf{Closed}(\Delta)$. Taking into account the fact that each $\delta \in \Delta'$ contains a larger number of disjuncts than at least one $\rho \in \mathsf{SResolvents}(\Phi)$, removing Δ' from the knowledgebase affects the connectivity of the graph and can result in a substantially reduced graph as the search space for canonical undercuts for $\langle \Phi, \alpha \rangle$. This is demonstrated in the following example.

Example 32. *Let* $\Delta = \{q \vee c, \neg q \vee p, q \vee \neg c, \neg p \vee g, \neg c \vee b, a \vee b \vee c, \neg g \vee \neg b, \neg a \vee b, \neg b, b \vee e, \neg e, \neg a \vee f, \neg f, s \vee t, \neg s \vee m, \neg m \vee n, h \vee j, \neg j \vee i\}$. *Then, for* $\Phi = \{a \vee b \vee c, \neg c \vee b, \neg b\}$ *and* $\alpha = a \vee d$, $\langle \Phi, \alpha \rangle$ *is an argument. Assume we want to find all the canonical undercuts for* $\langle \Phi, \alpha \rangle$. *It holds that* $\mathsf{SResolvents}(\Phi) = \{a, \neg c, \neg b\}$. *Let* $\Delta' = \{\delta \in \Delta \mid \exists \rho \in \mathsf{SResolvents}(\Phi) \; s \; .t. \; \rho \vdash \delta\}$. *Then,* $\Delta' = \{\neg c \vee b, a \vee b \vee c, q \vee \neg c, \neg g \vee \neg b, \neg b\}$ *and* $\Delta \setminus \Delta' = \{q \vee c, \neg f, \neg a \vee f, \neg a \vee b, b \vee e, \neg e, \neg q \vee p, \neg p \vee g, s \vee t, \neg s \vee m, \neg m \vee n, h \vee j, \neg j \vee i\}$ *contains the supports for all the canonical undercuts for* $\langle \Phi, \alpha \rangle$. *In order to retrieve them we can use the closed graph for* $\Delta'' = (\Delta \setminus \Delta') \cup \mathsf{SResolvents}(\Phi) = \{q \vee c, \neg f, \neg a \vee f, \neg a \vee b, b \vee e, \neg e, \neg q \vee p, \neg p \vee g, s \vee t, \neg s \vee m, \neg m \vee n, h \vee j, \neg j \vee i, a, \neg b, \neg c\}$.

The closed graph for Δ is



and the closed graph for Δ'' is



The closed graph for Δ'' contains the supports for $A_1 = \langle\{\neg f, \neg a \vee f\}, \diamond\rangle$, $A_2 = \langle\{b \vee e, \neg e\}, \diamond\rangle$ and $A_3 = \langle\{\neg a \vee b\}, \diamond\rangle$ that are all the canonical undercuts for $\langle\Phi, \alpha\rangle$ from Δ.

So, in order to find all the canonical undercuts for $\langle\Phi, \alpha\rangle$, we can find arguments for each $\neg\rho_i$ where $\rho_i \in \mathsf{SResolvents}(\Phi)$, using $\Delta \setminus \{\delta \in \Delta \mid \exists\rho \in \mathsf{SResolvents}(\Phi)$ s.t. $\rho \vdash \delta\} \cup \mathsf{SResolvents}(\Phi)$ as the background knowledge. Then, we obtain for each $\neg\rho_i$, arguments $\langle\Psi_i \cup \Gamma_i, \neg\rho_i\rangle$ where $\Psi_i \subseteq \Delta$ and $\Gamma_i \subset \mathsf{SResolvents}(\Phi)$ (where Γ_i can be the empty set).

Example 33. *In order to obtain the canonical undercuts of example 32, using the negation of $\rho_1 = a$ and $\rho_2 = \neg b$ from $\mathsf{SResolvents}(\Phi)$ as claims for arguments we obtain*
$A'_1 = \langle\Psi_1 \cup \Gamma_1, \neg\rho_1\rangle = \langle\{\neg f, \neg a \vee f\} \cup \emptyset, \neg a\rangle$,
$A'_2 = \langle\Psi_2 \cup \Gamma_2, \neg\rho_2\rangle = \langle\{b \vee e, \neg e\} \cup \emptyset, b\rangle$,
$A'_3 = \langle\Psi_3 \cup \Gamma_3, \neg\rho_1\rangle = \langle\{\neg a \vee b\} \cup \{\neg b\}, \neg a\rangle$
$A'_4 = \langle\Psi_4 \cup \Gamma_4, \neg\rho_2\rangle = \langle\{\neg a \vee b\} \cup \{a\}, b\rangle$

The part Ψ_i (i=1...4) of each the support sets $\Psi_i \cup \Gamma_i$ of the arguments above, which contains clauses from Δ, gives a support for a canonical undercut for $\langle\Phi, \alpha\rangle = \langle\{a \vee b \vee c, \neg c \vee b, \neg b\}, a \vee d\rangle$.

It is not always the case though that an argument $\langle\Psi_i \cup \Gamma_i, \neg\rho_i\rangle$ where $\Psi_i \subseteq \Delta$, $\Gamma_i \subset \mathsf{SResolvents}(\Phi)$ and $\rho_i \in \mathsf{SResolvents}(\Phi)$ indicates a support set Ψ_i for a canonical undercut $\langle\Psi_i, \diamond\rangle$ for $\langle\Phi, \alpha\rangle$. For Ψ_i, it holds that it is consistent and $\Psi_i \vdash \neg \bigwedge \mathsf{SResolvents}(\Phi)$ (and hence $\Psi_i \vdash \neg\wedge\Phi$). It can be the case though where

Ψ_i is not minimal for entailing $\neg \bigwedge \mathsf{SResolvents}(\Phi)$ so $\langle \Psi_i, \neg \bigwedge \mathsf{SResolvents}(\Phi)\rangle$ is not an argument and so there is a $\Psi_j \subset \Psi_i$ such that $\langle \Psi_j, \neg \bigwedge \mathsf{SResolvents}(\Phi)\rangle$ is an argument. In this case, there is a $\Gamma_j \subset \mathsf{SResolvents}(\Phi)$ such that $\Psi_j \cup \Gamma_j$ is a minimal inconsistent set. Then, for some $\rho_j \in \Gamma_j$ (where possibly $\rho_j = \rho_i$) it holds that $\langle (\Psi_j \cup \Gamma_j) \setminus \{\rho_j\}, \neg\rho_j\rangle$ is an argument. After having found all the arguments for all the $\neg\rho_i$ we can decide which ones indicate canonical undercuts for $\langle \Phi, \alpha\rangle$ by comparing them and checking whether for some $\langle \Psi_i \cup \Gamma_i, \neg\rho_i\rangle$ there is some $\langle \Psi_j \cup \Gamma_j, \neg\rho_j\rangle$ (where ρ_j may or may not be equal to ρ_i) such that $\Psi_j \subset \Psi_i$. The Ψ_i for which this condition does not hold for any Ψ_j are canonical undercuts for $\langle \Phi, \alpha\rangle$. We capture the correctness of using strong resolvents for finding arguments in the following result.

Proposition 17. *Let $\langle \Phi, \alpha\rangle$ be an argument. $\langle \Psi, \diamond\rangle$ is a canonical undercut for $\langle \Phi, \alpha\rangle$ iff there is a $\rho_i \in \mathsf{SResolvents}(\Phi)$ and a $\Gamma_i \subset \mathsf{SResolvents}(\Phi)$ (possibly the empty set) such that $\langle \Psi \cup \Gamma_i, \neg\rho_i\rangle$ is an argument and there is no $\Psi' \subset \Psi$ and $\Gamma_j \subset \mathsf{SResolvents}(\Phi)$ and $\rho_j \in \mathsf{SResolvents}(\Phi)$ (where ρ_j can be equal to ρ_i) such that such that $\langle \Psi' \cup \Gamma_j, \neg\rho_j\rangle$ is an argument.*

Example 34. *Let $\Delta = \{e \vee q \vee f, \neg f \vee e, \neg e \vee \neg f, f, e \vee \neg q\}$. Then $\langle \Phi, \alpha\rangle = \langle \{e \vee q \vee f, \neg f \vee e\}, e \vee q\rangle$ is an argument and $\mathsf{SResolvents}(\Phi) = \{\neg f \vee e, e \vee q\}$. Then, the following are arguments:*
$A_1 = \langle \Psi_1 \cup \Gamma_1, \neg\rho_1\rangle = \langle \{f, \neg e \vee \neg f, e \vee \neg q\} \cup \emptyset, \neg(e \vee q)\rangle$
$A_2 = \langle \Psi_2 \cup \Gamma_2, \neg\rho_2\rangle = \langle \{f, \neg e \vee \neg f\} \cup \emptyset, \neg(\neg f \vee e)\rangle$,
and $\Psi_2 \subset \Psi_1$ so $\langle \Psi_1, \diamond\rangle$ is not a canonical undercut for $\langle \Phi, \alpha\rangle$. $\langle \Psi_2, \diamond\rangle$ is a canonical undercut for $\langle \Phi, \alpha\rangle$.

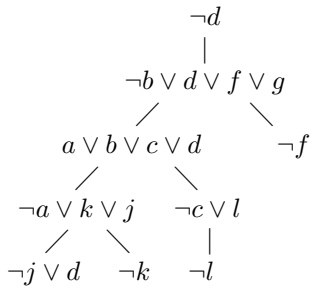
6.2. Using a support tree to generate canonical undercuts

In section 6.1 we described how by using the clauses from $\mathsf{SResolvents}(\Phi)$ and finding arguments for each $\neg\rho_i$, where $\rho_i \in \mathsf{SResolvents}(\Phi)$, we can generate undercuts for an argument $\langle \Phi, \beta\rangle$. This requires finding arguments for the negated clause $\neg\rho_i$. We have proposed the support tree as a way to find arguments for a claim that is a clause. In this section we describe how the support tree can be used again as the structure to provide the arguments for $\neg\rho_i$.

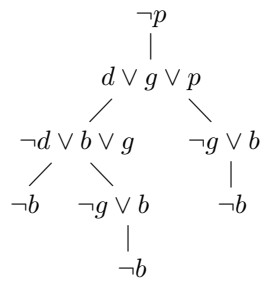
Let (N, A, f) be a support tree for Δ, α and a where α is a clause consisting of a unique literal. Then, $\mathsf{Disjuncts}(\alpha) = \{a\}$, and $\overline{a}$ represents the root node of (N, A, f). If Γ' is the set of non-root nodes of (N, A, f) then by proposition 13, $\langle \Gamma', \alpha\rangle$ is an argument and for $\Gamma = \Gamma' \cup \{\neg\alpha\}$ it holds that Γ is a minimal inconsistent set. Because $\neg\alpha = \overline{a}$ it holds that Γ is equal to the set of clauses assigned to the nodes of the tree, i.e. $\Gamma = \{f(x) \mid x \in N\}$. Then for all $\gamma \in \Gamma$ it holds that $\langle \Gamma \setminus \{\gamma\}, \neg\gamma\rangle$ is an argument. Hence, in order to find an argument for a negated clause $\neg\gamma$, we can find a support tree (N, A, f) for a claim α that consists of a unique disjunct, where γ is assigned to a node in (N, A, f) and obtain in this way a minimal inconsistent set that contains γ. For this we construct a clause γ' that consists of the literals of γ together with an arbitrary literal p whose atom does not appear anywhere in the knowledgebase and the disjuncts of γ. i.e. $\gamma' = \bigvee(\mathsf{Disjuncts}(\gamma) \cup \{p\})$ where $p \notin \mathsf{Literals}(\Delta \cup \{\gamma\})$ and

$\overline{p} \notin \mathsf{Literals}(\Delta \cup \{\gamma\})$. Using $\alpha' = p$ as the clause for which a support tree will be constructed from $\Delta \cup \{\gamma'\}$, ensures that γ' will be contained in all the minimal inconsistent sets that also contain $\overline{p}$ and all the minimal inconsistent sets that contain γ' are the ones that also contain $\overline{p}$. If (N, A, f) is a support tree for $\Delta \cup \{\gamma'\}$, $\alpha' = p$ and $p \in \mathsf{Disjuncts}(\alpha')$, then $\{f(x) \mid x \in N\}$ is a minimal inconsistent set and $\{\gamma', \neg p\} \subset \{f(x) \mid x \in N\}$. From the way γ' is constructed it holds that $\gamma' \wedge \neg p \equiv \gamma$. Then, $\{f(x) \mid x \in N\} \setminus \{\gamma', \neg p\} \cup \{\gamma\}$ is a minimal inconsistent set, so $\langle \{f(x) \mid x \in N\} \setminus \{\gamma', \neg p\}, \neg\gamma \rangle$ is an argument and $\{f(x) \mid x \in N\} \setminus \{\gamma', \neg p\} \subseteq \Delta$. Therefore, the nodes of (N, A, f) that are located below the node represented by γ' correspond to a support for an argument for $\neg\gamma$.

Example 35. *The support tree (N, A, f) of example 26 from section 5.2, gives an argument for $\alpha = d \vee m \vee g$ from $\Delta = \{\neg b \vee d \vee f \vee g, a \vee b \vee c \vee d, \neg a \vee k \vee j, \neg j \vee d, \neg k, \neg c \vee l, \neg l, \neg f, \neg d \vee b \vee g, \neg g \vee b, \neg b, \neg d \vee \neg j, j, \neg g, c \vee l\}$.*



The set of clauses corresponding to the non-root nodes of (N, A, f) is $\Phi = \{\neg b \vee d \vee f \vee g, a \vee b \vee c \vee d, \neg a \vee k \vee j, \neg j \vee d, \neg k, \neg c \vee l, \neg l, \neg f\}$. Then, $\langle \Phi, \alpha \rangle$ is an argument and $\mathsf{SResolvents}(\Phi) = \{\neg f, \neg k, \neg a \vee j, d \vee \neg a, d \vee g, d \vee b, \neg c, \neg l, \neg j \vee d\}$. Assume we want to find a canonical undercut for $\langle \Phi, \alpha \rangle$ and for this we look for an argument for the negation of $\gamma = d \vee g \in \mathsf{SResolvents}(\Phi)$. If $\gamma' = d \vee g \vee p$ and $\alpha' = p$, and $\Delta' = \Delta \cup \{\gamma'\} \cup \mathsf{SResolvents}(\Phi)$, the following is a support tree for Δ', α' and p.



Then for z such that $f(z) = \gamma' = d \vee g \vee p$, if $\Psi = \{f(x) \mid x \in \mathsf{Subtree}(z) \setminus \{z\}\}$, then $\Psi = \{\neg b, \neg g \vee b, \neg d \vee b \vee g\}$, and $\langle \Psi, \neg d \wedge \neg g \rangle$ is an argument and is equal to $\langle \Psi, \neg\gamma \rangle$.

7. Algorithms

In this section we present the algorithms that implement the theory presented in sections 5 and 6. First we give the algorithms related to the search for arguments for a given claim as described in section 5 and then the algorithms related to the search for canonical undercuts for a given argument as described in section 6.

7.1. Algorithms generating arguments

In this section we present the algorithm that searches for all the support trees for a clause α and an $a \in \mathsf{Disjuncts}(\alpha)$ from a knowledgebase Δ. Then, according to proposition 13, each of the support trees retrieved gives an argument for α. If this algorithm is applied for all $a_i \in \mathsf{Disjuncts}(\alpha)$, then according to proposition 12, for all the arguments for α from Δ there is a support tree for Δ, α and some $a_i \in \mathsf{Disjuncts}(\alpha)$ retrieved by the algorithm.

Algorithm 1 builds a depth-first search tree T that represents the steps of the search for arguments for a claim α from a knowledgebase Δ. Every node v' in T is a presupport tree and is an extension of the presupport tree in its parent node v in T. The leaf node of every completed accepted branch is a support tree.

The search is based on the query graph of α in Δ (for the algorithm that produces the query graph see [17]). The starting point for the search is the complement $\overline{a}$ of one of the disjuncts of α which will also be the root of the retrieved presupport trees. The idea in building presupport trees by using the structure of the query graph, is to start from $\overline{a}$ and walk over the graph by following the links in a way that the conditions for a complete presupport tree indicate. After all the complete presupport trees have been generated, the ones that satisfy the conditions for a minimal and consistent presupport tree will be returned. Every step of algorithm SearchTree, which extends the search tree T by a node, consists of extending the presupport tree (N, A, f) that represents the current leaf node v by one level when condition *iii*) of definition 21 needs to be satisfied for some of the current leaf nodes of (N, A, f). Function $\mathsf{Extensions}(v)$ gives all the possible extensions of the search tree i.e. all the possible presupport trees that extend the presupport tree in v by one level. If for a node v of the search tree $\mathsf{Extensions}(v) = \emptyset$, then it means that the current branch of the search tree cannot be expanded any further below node v. This happens when for the presupport tree (N, A, f) that is contained in node v

1. either (N, A, f) is a complete presupport tree
2. or for a node x in (N, A, f) a child node has to be added but all the clauses that can be assigned to a child node of x have already been assigned to the ancestors of x
3. or for a node x in (N, A, f) and some $b \in \mathsf{Disjuncts}(f(x))$ there is an arc (w, w'), s.t. $w' \in \mathsf{Ancestors}(x)$ and $\mathsf{Attacks}(f(w), f(w')) = \overline{b}$ and so according to proposition 7 (N, A, f) cannot be extended to a complete presupport tree.

In such a case where a leaf node v of the search tree has been reached, the algorithm uses boolean function $\mathsf{Accept}(v)$ which either stores or rejects the presupport tree (N, A, f) in the leaf node v of the currently built branch. $\mathsf{Accept}(v)$ rejects v in cases 2 and 3 given above. If case 1 holds and the presupport tree (N, A, f) that represents v is a complete presupport tree, then $\mathsf{Accept}(v)$ tests further, whether (N, A, f) is a support tree. First it tests whether (N, A, f) satisfies definition 22 for a consistent presupport tree. If it does not satisfy this definition it rejects this node. If (N, A, f) does satisfy the consistency condition, then $\mathsf{Accept}(v)$ tests whether (N, A, f) is a minimal presupport tree by applying the method described in section 5.3. If the minimality check is also satisfied, (N, A, f) is a support tree for Δ, α and a and so is stored in the set $SupportTrees$ of support trees that will be returned in the end. Because function $\mathsf{Extensions}(v)$ is extending the presupport tree of each next new node in T according to the conditions of definition 21, all the non-leaf nodes of a presupport tree (N, A, f) of a node v from T satisfy the conditions for a complete presupport and $\mathsf{Accept}(v)$ only tests the leaf nodes of (N, A, f). After $\mathsf{Accept}(T)$ has either rejected the current branch of T, or stored the result of its leaf node, the algorithm backtracks and continues to the next node of T to be expanded.

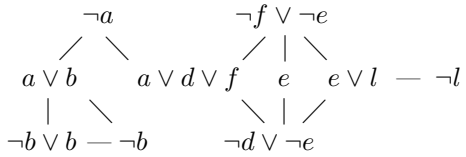
After this algorithm has been applied for all $a \in \mathsf{Disjuncts}(\alpha)$, all the arguments for α from Δ are obtained.

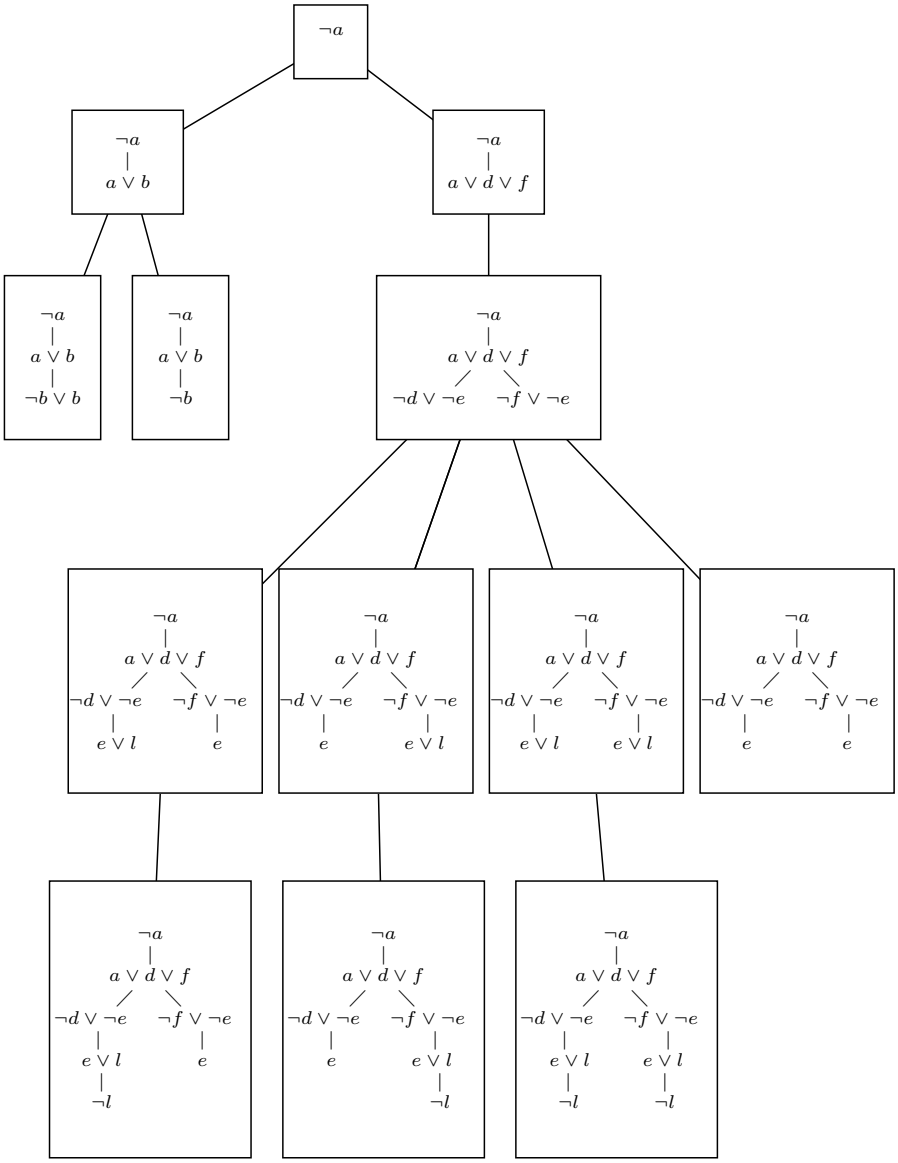
Algorithm 1 $\mathsf{SearchTree}(a)$

Let $SupportTrees = \emptyset$
Let r be a node that contains (N_0, A_0, f_0) s.t. $A_0 = \emptyset$ and $N_0 = \{x\}$ with $f_0(x) = \overline{a}$
Let S be an empty stack
Push r onto S
while S is non-empty **do**
  Let v be the top of S
  pop S
  if $\mathsf{Extensions}(v) \neq \emptyset$ **then**
    for all $y \in \mathsf{Extensions}(v)$ **do**
      push y onto S
    end for
  else
    if $\mathsf{Accept}(v)$ **then**
      $SupportTrees = SupportTrees \cup \{v\}$
    end if
  end if
end while
return $SupportTrees$

Example 36. *Let* $\alpha = a \vee m$ *and* $\Delta = \{a \vee b, \neg b \vee b, \neg b, a \vee d \vee f, \neg f \vee \neg e, e, \neg d \vee \neg e, e \vee l, \neg l, m \vee k, \neg s \vee g, r \vee j, j \vee \neg s, \neg s \vee k, p\}$. *The query graph of* α *in* Δ

is given below and contains the negation $\neg a$ *of* $a \in \mathsf{Disjuncts}(\alpha)$ *which is the unique starting point for the search for arguments for* α*, since the negation* $\neg m$ *of* $m \in \mathsf{Disjuncts}(\alpha)$ *does not appear in the graph. Figure 1 illustrates how algorithm 1 walks over this query graph starting from* $\neg a$*.*





The result of applying algorithm SearchTree for $\alpha = a \vee m$ using the query graph of example 36 where the starting node is $\neg a$. The leaf node of the first branch is rejected because for y with $f(y) = \neg b \vee b$, $b \in \mathsf{Disjuncts}(f(y))$ and $\neg b \in \mathsf{AncestorLabels}(y)$ so by proposition 7 this cannot be expanded to a complete presupport tree. The result of the second branch is accepted. The results of the third and fourth branch are not accepted because the presupport trees they represent are non-minimal and the results of the fifth and sixth branch are accepted.

7.2. Using a support tree to generate resolvents

Let $\langle \Phi, \alpha \rangle$ be an argument, where α and the formulae from Φ are clauses. In section 5.2 we proved that there is an $a \in \mathsf{Disjuncts}(\alpha)$ such that (N, A, f) is a support tree for Δ, α and some $a \in \mathsf{Disjuncts}(\alpha)$, and $\Phi = \{f(x) \mid x \in N\} \setminus \{\overline{a}\}$. In section 6 we described how using the negation of each of the clauses from $\mathsf{SResolvents}(\Phi)$ as claims for arguments can be useful for generating canonical undercuts for $\langle \Phi, \alpha \rangle$. In this section we give the algorithm that generates $\mathsf{SResolvents}(\Phi)$ based on the structure of the suppport tree (N, A, f) that corresponds to Φ.

In section 7 we gave the algorithms that given Δ and α return the support trees for Δ and α and $a \in \mathsf{Disjuncts}(\alpha)$. Apart from providing a minimal and consistent proof for a claim α, a support tree can also be used to generate $\mathsf{SResolvents}(\Phi)$ efficiently. The way the support tree is built helps minimizing the number of disjuncts of the resolvents produced when the order in which the resolution of clauses happens corresponds to a post-order traversal of the support tree.

Given a support tree (N, A, f) for Δ, α and a that corresponds to the support Φ of an argument $\langle \Phi, \alpha \rangle$, i.e. $\Phi = \{f(x) \mid x \in N\} \setminus \{\overline{a}\}$, algorithm $\mathsf{GetSResolvents}((N, A, f))$ returns the set $\mathsf{SResolvents}(\Phi)$. Using function $\mathsf{ResolveSubtree}(y)$ the algorithm assigns a set of clauses to each node y, that represents the set of strong resolvents produced by $\{f(x) \mid x \in \mathsf{Subtree}(y)\}$. Initially, before the traversal takes place, the value for $\mathsf{ResolveSubtree}(y)$ for each non-root node y from N is initialized to $\mathsf{ResolveSubtree}(y) = \{f(y)\}$ and for the root node is initialized to be equal to the empty set (because $\overline{a} \notin \Phi$). Each node is visited after all its children have been visited. Every time a child x of node y is visited, the value of $\mathsf{Resolvents}(\mathsf{ResolveSubtree}(x) \cup \mathsf{ResolveSubtree}(y))$ is assigned to $\mathsf{ResolveSubtree}(y)$. Every time set $\mathsf{ResolveSubtree}(y)$ is updated, the clauses for which there is a stronger clause in $\mathsf{ResolveSubtree}(y)$ are removed using function $\mathsf{RemoveSubsumed}$. In this way the number of resolvents produced at each step is controlled and the comparison for subsumed clauses is focused on each node and happens locally. Hence for each node y that has been visited and processed during the traversal, $\mathsf{ResolveSubtree}(y)$ gives the set of strong resolvents of the set of clauses that represent its subtree. When the root node *root* is reached during the traversal, the algorithm outputs $\mathsf{ResolveSubtree}(root)$ which is the set $\mathsf{SResolvents}(\{f(x) \mid x \in N\} \setminus \{\overline{a}\}) = \mathsf{SResolvents}(\Phi)$. In order to traverse the tree the algorithm is using functions $\mathsf{FirstChild}(x)$, $\mathsf{Parent}(x)$, and $\mathsf{NextSibling}(x)$ that given a node x, return respectively the first child of x, the parent of x and the next sibling of x in (N, A, f). Boolean function $\mathsf{Visited}(x)$ indicates whether node x has been visited so far during the traversal.

In the case where (N, A, f) is a support tree that has been retrieved while looking for a canonical undercut $\langle \Psi, \diamond \rangle$ for some argument that has a support Ψ', then apart from clauses from Ψ, (N, A, f) can also contain clauses from $\mathsf{SResolvents}(\Psi')$. As we explained in section 6, in order to find a canonical undercut for an argument with support Ψ' we look for arguments $\langle \Psi \cup \Gamma, \neg\rho \rangle$ for all the $\rho \in \mathsf{SResolvents}(\Psi')$ from $(\Delta \setminus \Delta') \cup \mathsf{SResolvents}(\Psi')$ where Δ' is

the set of clauses that are subsumed by some clauses from $\mathsf{SResolvents}(\Psi')$, $\Psi' \subset \Delta$ and $\Gamma \subset \mathsf{SResolvents}(\Psi')$. In this case, the presupport tree contains the clauses of $\Psi \cup \Gamma$, where Γ may or may not be equal to the empty set. Then, the traversal would produce $\mathsf{SResolvents}(\Psi \cup \Gamma)$ rather than $\mathsf{SResolvents}(\Psi)$. In order to avoid this, the nodes z of (N, A, f) for which $f(z) \in \Gamma$ have the value for $\mathsf{ResolveSubtree}(z)$ initialized to be the empty set. This way, no clauses from Γ are involved in the resolvents produced by the traversal of (N, A, f). For the same reason, in all presupport trees the root node has its values for ResolveSubtree initialized to be equal to the empty set since it does not represent a clause from the set for which the set of strong resolvents has to be generated.

Algorithm 2 $\mathsf{GetSResolvents}((N, A, f))$

1: $x = root$
2: **while** $x \neq null$ **do**
3: **if** $\mathsf{FirstChild}(x) \neq null$ & $\mathsf{Visited}(x) == false$ **then**
4: $\mathsf{Visited}(x) = true$
5: $x = \mathsf{FirstChild}(x)$
6: **else**
7: $y = \mathsf{Parent}(x)$
8: $\mathsf{ResolveSubtree}(y) = \mathsf{Resolvents}(\mathsf{ResolveSubtree}(y) \cup \mathsf{ResolveSubtree}(x))$
9: $\mathsf{ResolveSubtree}(y) = \mathsf{RemoveSubsumed}(\mathsf{ResolveSubtree}(y))$
10: **if** $\mathsf{NextSibling}(x) \neq null$ **then**
11: $x = \mathsf{NextSibling}(x)$
12: **else**
13: $x = \mathsf{Parent}(x)$
14: **end if**
15: **end if**
16: **if** $x == root$ **then**
17: return $\mathsf{ResolveSubtree}(x)$
18: **end if**
19: **end while**

The following example illustrates how algorithm GetSResolvents works using the support tree of example 26. For simplicity we use the clause representation of each node x to denote x, and each corresponding line of the algorithm operation is given on the right hand side of each step. This tree represents the search for an argument $\langle \Phi, \alpha \rangle$ for a clause α and not a canonical undercut for some argument, hence the set of non-root nodes from (N, A, f) is equal to Φ and all the non-root nodes x have their value for $\mathsf{ResolveSubtree}(x)$ initialized to be equal to $\{f(x)\}$. For the root node, since it does not have its clause representation in Φ, its value for ResolveSubtree is initialised to be equal to the empty set.

Example 37. *Let (N, A, f) be the support tree of example 26 where $\Phi = \{f(x) \mid x \in N\} \backslash \{\overline{a}\} = \{\neg b \vee d \vee f \vee g, a \vee b \vee c \vee d, \neg a \vee k \vee j, \neg c \vee l, \neg j \vee d, \neg k, \neg l, \neg c \vee l, \neg l\}$. The following represents the sequence of operations that corresponds to the application of algorithm 7.2 on (N, A, f).*

GetSResolvents$((N, A, f))$

$x = \neg d$ *(1)*

Visited$(\neg d) = true,\ x = \neg b \vee d \vee f \vee g$ *(4),(5)*

Visited$(\neg b \vee d \vee f \vee g) = true,\ x = a \vee b \vee c \vee d$ *(4),(5)*

Visited$(a \vee b \vee c \vee d) = true,\ x = \neg a \vee k \vee j$ *(4),(5)*

Visited$(\neg a \vee k \vee j) = true,\ x = \neg j \vee d$ *(4)(5)*

$y = \neg a \vee k \vee j$ *(7)*

ResolveSubtree$(\neg a \vee k \vee j) = \{\neg a \vee k \vee j, \neg j \vee d, \neg a \vee k \vee d\}$ *(8),(9)*

$x = \neg k$ *(11)*

$y = \neg a \vee k \vee j$ *(7)*

ResolveSubtree$(\neg a \vee k \vee j) = \{\neg a \vee k \vee j, \neg j \vee d, \neg a \vee k \vee d, \neg k, \neg a \vee j,$
$\neg a \vee d\}$ *(8)*

ResolveSubtree$(\neg a \vee k \vee j) = \{\neg j \vee d, \neg k, \neg a \vee j, \neg a \vee d\}$ *(9)*

$x = \neg a \vee k \vee j$ *(13)*

$y = a \vee b \vee c \vee d$ *(7)*

ResolveSubtree$(a \vee b \vee c \vee d) = \{a \vee b \vee c \vee d, \neg j \vee d, \neg k, \neg a \vee j, \neg a \vee d,$
$b \vee c \vee d \vee j, b \vee c \vee d\}$ *(8)*

ResolveSubtree$(a \vee b \vee c \vee d) = \{\neg j \vee d, \neg k, \neg a \vee j, \neg a \vee d, b \vee c \vee d\}$ *(9)*

$x = \neg c \vee l$ *(11)*

Visited$(\neg c \vee l) = true,\ x = \neg l$ *(4),(5)*

$y = \neg c \vee l$ *(7)*

ResolveSubtree$(\neg c \vee l) = \{\neg c \vee l, \neg l, \neg c\}$ *(8)*

ResolveSubtree$(\neg c \vee l) = \{\neg l, \neg c\}$ *(9)*

$x = \neg c \vee l$ *(13)*

$y = a \vee b \vee c \vee d$ *(7)*

ResolveSubtree$(a \vee b \vee c \vee d) = \{\neg j \vee d, \neg k, \neg a \vee j, \neg a \vee d, b \vee c \vee d, \neg l,$
$\neg c, b \vee d\}$ *(8)*

ResolveSubtree$(a \vee b \vee c \vee d) = \{\neg j \vee d, \neg k, \neg a \vee j, \neg a \vee d, \neg l, \neg c, b \vee d\}$ *(9)*

$x = a \vee b \vee c \vee d$ *(13)*

$y = \neg b \vee d \vee f \vee g$ *(7)*

ResolveSubtree$(\neg b \vee d \vee f \vee g) = \{\neg b \vee d \vee f \vee g, \neg j \vee d, \neg k, \neg a \vee j, \neg a \vee d, \neg l,$
$\neg c, b \vee d, d \vee f \vee g\}$ *(8)*

ResolveSubtree$(\neg b \vee d \vee f \vee g) = \{\neg j \vee d, \neg k, \neg a \vee j, \neg a \vee d, \neg l, \neg c, b \vee d,$
$d \vee f \vee g\}$ *(9)*

$x = \neg f$ *(11)*

$y = \neg b \vee d \vee f \vee g$ *(7)*

ResolveSubtree$(\neg b \vee d \vee f \vee g) = \{\neg j \vee d, \neg k, \neg a \vee j, \neg a \vee d, \neg l, \neg c, b \vee d,$
$d \vee f \vee g, \neg f, d \vee g\}$ *(8)*

ResolveSubtree$(\neg b \vee d \vee f \vee g) = \{\neg j \vee d, \neg k, \neg a \vee j, \neg a \vee d, \neg l, \neg c, b \vee d,$
$\neg f, d \vee g\}$ (9)

$x = \neg b \vee d \vee f \vee g$ *(13)*

$y = root$ *(7)*

ResolveSubtree$(root) = \{\neg j \vee d, \neg k, \neg a \vee j, \neg a \vee d, \neg l, \neg c, b \vee d, \neg f, d \vee g\}$ *(8),(9)*

return ResolveSubtree$(root)$ *(17)*

So, the set of clauses returned by the algorithm is $\{\neg j \vee d, \neg k, \neg a \vee j, \neg a \vee d, \neg l, \neg c, b \vee d, \neg f, d \vee g\}$ *and is the set* $\mathsf{SResolvents}(\Phi)$ *of strong resolvents of* Φ.

7.3. Algorithms for generating counterarguments

As we described in section 6, given an argument $\langle \Phi, \alpha \rangle$, and the requirement to find all the canonical undercuts for $\langle \Phi, \alpha \rangle$ from Δ, we can start by looking for all the arguments

$\mathsf{Args}(\neg\rho_1) = \langle \Psi_1^1 \cup \Gamma_1^1, \neg\rho_1 \rangle, \ldots \langle \Psi_k^1 \cup \Gamma_k^1, \neg\rho_1 \rangle$

$\vdots$

$\mathsf{Args}(\neg\rho_n) = \langle \Psi_1^n \cup \Gamma_1^n, \neg\rho_n \rangle, \ldots \langle \Psi_m^n \cup \Gamma_m^n, \neg\rho_n \rangle$

for all $\rho_1, \ldots, \rho_n \in \mathsf{SResolvents}(\Phi)$ from $\Delta \cup \mathsf{SResolvents}(\Phi)$ where $\Psi_i \subseteq \Delta$ and $\Gamma_i \subset \mathsf{SResolvents}(\Phi)$. As explained in section 6, we can use the set $\Delta'' = (\Delta \setminus \{\delta \in \Delta \mid \exists \rho \in \mathsf{SResolvents}(\Phi) \text{ s .t. } \rho \vdash \delta\}) \cup \mathsf{SResolvents}(\Phi)$ as the knowledgebase from which all these arguments are generated. Also, as explained in section 6.2, although the support tree is defined with respect to a disjunctive clause and is related to finding arguments for claims that are clauses, we can use it to find arguments for a negated clause $\neg\rho_i$. For this we construct a clause ρ_i' consisting of the literals of ρ_i together with an arbitrary literal p whose atom does not appear anywhere in Δ or α, and hence anywhere in Δ''.

Algorithm 3 $\mathsf{GetCounterarguments}((N, A, f))$

$Undercuts = \emptyset$, $Canonical = \emptyset$
$\mathsf{SResolvents}(\Phi) = \mathsf{GetSResolvents}((N, A, f))$
$\Delta'' = \mathsf{RemoveSubsumedOf}(\Delta, \mathsf{SResolvents}(\Phi))$
$\Delta'' = \Delta'' \cup \mathsf{SResolvents}(\Phi)$
Let v be node containing (N, A, f) s.t. $N = \{x\}, A = \emptyset, \{f(x) \mid x \in N\} = \{p\}$
for all $\rho_i \in \mathsf{SResolvents}(\Phi)$ **do**
  $\rho_i' = \mathsf{Augment}(\rho_i, p)$
  $\Delta'' = \Delta'' \cup \{\rho_i'\}$
  $Undercuts = Undercuts \cup \mathsf{SearchTree}(v)$
  $\Delta'' = \Delta'' \setminus \{\rho_i'\}$
end for
$Canonical = \mathsf{getCanonical}(Undercuts)$
return *Canonical*

Then, a support tree (N, A, f) for $\Delta'' \cup \{\rho_i'\}$, $\alpha' = p$ and p indicates an argument for $\neg\rho_i$ where the support of this argument is the set of clauses that represent (N, A, f) excluding clauses $\alpha' = p$ and ρ_i'. Let function $\mathsf{Augment}(\rho_i, p)$ return ρ_i' as described above, and let $\mathsf{RemoveSubumedOf}(\Delta, \mathsf{SResolvents}(\Phi))$ return the set $\Delta \setminus \{\delta \in \Delta \mid \exists \rho \in \mathsf{SResolvents}(\Phi) \text{ s .t. } \rho \vdash \delta\}$. Then, given the support tree (N, A, f) that corresponds to argument $\langle \Phi, \alpha \rangle$, algorithm $\mathsf{GetCounterarguments}((N, A, f))$ returns the set of all canonical undercuts for

$\langle \Phi, \alpha \rangle$. For this, for each $\rho_i \in \mathsf{SResolvents}(\Phi)$, a search tree T_i is generated by algorithm SearchTree of section 7.1 for $\alpha' = p$ where the knowledgebase is $\Delta'' \cup \{\rho'_i\}$. Each of T_i is then added to set $Undercuts$ and the algorithm proceeds in the same way by building a tree T_j associated to the next $\rho_j \in \mathsf{SResolvents}(\Phi)$ using $\Delta'' \cup \{\rho'_j\}$ (where $\rho'_j = \mathsf{Augment}(\rho_j, p)$) as the knowledgebase until all the clauses from $\mathsf{SResolvents}(\Phi)$ have been examined.

After having generated all these arguments, the algorithm refines which indicate canonical undercuts for $\langle \Phi, \alpha \rangle$ by comparing them with function getCanonical. The $\langle \Psi^i_s \cup \Gamma^i_s, \neg\rho_i \rangle$ for which there is no $\langle \Psi^j_l \cup \Gamma^j_l, \neg\rho_j \rangle$ (where ρ_j may be equal to ρ_i) such that $\Psi^j_l \subset \Psi^i_s$, indicate a canonical undercut $\langle \Psi^i_s, \diamond \rangle$ for $\langle \Phi, \alpha \rangle$. Because the literals from each ρ_i are linked to the literals of the clauses of each support $\Psi^i_s \cup \Gamma^i_s$ for $\langle \Psi^i_s \cup \Gamma^i_s, \neg\rho_i \rangle$, it is more likely for a Ψ^j_l from $\langle \Psi^j_l \cup \Gamma^j_l, \neg\rho_j \rangle$ to be contained in Ψ^i_s from $\langle \Psi^i_s \cup \Gamma^i_s, \neg\rho_i \rangle$ when these are arguments for the same claim i.e. when it holds that $\rho_i = \rho_j$. For this reason, function getCanonical first compares the supports for arguments that correspond to each $\rho_i \in \mathsf{SResolvents}(\Phi)$ individually and reject the ones that cannot indicate a canonical undercut for $\langle \Phi, \alpha \rangle$ as described above. Then, it compares the supports for the remaining arguments of different ρ_i, ρ_j and similarly it rejects the ones that according to proposition 17 do not indicate a canonical undercut for $\langle \Phi, \alpha \rangle$.

7.4. Algorithm for argument trees

In this section we introduce an algorithm which, putting together the algorithms given so far generates an argument tree. Given the support tree (N_0, A_0, f_0) for Δ, α and a, that corresponds to an argument for α, algorithm $\mathsf{ArgumentTree}((N_0, A_0, f_0))$ generates an argument tree that has $\langle \{f_0(x) \mid x_0 \in N_0\} \setminus \{\overline{a}\}, \alpha \rangle$ as its root. It uses function $\mathsf{Node}((N, A, f), parentNode)$ which, given a support tree (N, A, f), creates an argument tree node structure that is identified by (N, A, f) and has $parentNode$ as its parent in the argument tree. With the method described in section 6, when looking for canonical undercuts for an argument that has support Ψ', a canonical undercut $\langle \Psi, \diamond \rangle$ is retrieved for this argument by finding arguments $\langle \Psi \cup \Gamma, \neg\rho_i \rangle$ for some $\rho_i \in \mathsf{SResolvents}(\Psi)$ where $\Gamma \subset \mathsf{SResolvents}(\Psi')$. Therefore, apart from the support tree (N_0, A_0, f_0) of the root argument, the rest of the nodes of the argument tree are identified by support trees (N, A, f) that may contain clauses from the set of strong resolvents of their parent node in the argument tree. So, the support for each undercut $\langle \Psi, \diamond \rangle$ is given by the relation $\Psi = (\{f(x) \mid x \in N \setminus root\} \cap \Delta) \setminus \Gamma$ where $root$ is the root of (N, A, f).

The algorithm generates the tree in a depth-first way and a branch stops expanding when either there is no canonical undercut for the current leaf from the knowledgebase, or the canonical undercuts for the current leaf have supports whose clauses have all been used in the supports of the arguments of the branch and so the branch cannot be extended further because this would violate condition (2) of the definition of the argument tree. Function BranchClauses gives for each node of the argument tree the set of clauses that appear in the supports of its ancestor nodes in the branch where it belongs and is used by the algorithm

Algorithm 4 ArgumentTree$((N_0, A_0, f_0))$

$Arcs = \emptyset$, $Nodes = \emptyset$
$root = Node((N_0, A_0, f_0), null)$
S is an empty Stack
push $root$ onto S
while S is not empty **do**
  $topNode = ((N_t, A_t, f_t), t')$ is the top of S
  pop S
  $Canonical =$ GetCounterarguments$((N_t, A_t, f_t))$
  $\Gamma =$ GetSResolvents$((N_t, A_t, f_t))$
  for all $(N, A, f) \in Canonical$ **do**
    $\Psi = (\{f(x) \mid x \in N \setminus root\} \cap \Delta) \setminus \Gamma$
    if $\Psi \not\subseteq$ BranchClauses$(topNode)$ **then**
      $newNode = Node((N, A, f), topNode)$
      BranchClauses$(newNode) =$ BranchClauses$(topNode) \cup \Psi$
      $Arcs = Arcs \cup \{(newNode, topNode)\}$
      $Nodes = Nodes \cup \{newNode\}$
      push $newNode$ onto S
    end if
  end for
end while
return $(Arcs, Nodes)$

to monitor whether a new node can be added on the tree without violating condition (2) of the definition of the argument tree.

7.5. Implementation

We have implemented a prototype software system based on the algorithms presented in this section. The software system JArgue which is implemented in Java provides a tool for generating argument trees in classical propositional logic given knowledgebases and claims that are from $\mathcal{C}$. Running on an ordinary PC ((Core2 Duo 2.13GHz, RAM 3GB), given a knowledgebase of 40 clauses that consist of 1,2 and 3 disjuncts, the system can generate an argument tree of 60 nodes and 20 branches in less than 2.5 seconds whereas an argument tree of 90 nodes and 30 branches can be generated in less than 4 seconds. The supports for these nodes consist of an average of 4 clauses. So, with this kind of data and an average time per node ratio of 40 milliseconds, an argument tree with branching factor 2 and height 5 that consists of $2^{(5+1)} - 1 = 63$ nodes arranged in 32 branches, can be generated in 2.5 seconds.

Further experiments with input knowledgebases containing clauses of 1 and 3 disjuncts gave argument trees of 700 nodes on average with an average node support of 3.15 clauses in 1.8 minutes. Argument trees that were produced using test sets consisting of 1 and 2 place clauses had smaller supports in their nodes on average. For this data, an argument tree with an average node support of 2.32

clauses, consisting of 700 nodes on average was produced in 0.5 minutes, while trees of 3,500 nodes were produced in 3.6 minutes. The largest tree produced in this experiment consisted of 9,137 nodes and was generated in 13.5 minutes.

The examples of this paper are produced in negligible time (i.e. less than 0.1 seconds). In future work we plan to evaluate the performance of the system through systematic empirical experimentation. For more details, see [19] and [26].

8. Discussion

Argumentation is a subject that is of much interest at the moment with numerous applications including conflict resolution in multiagent systems (e.g. [3]), reasoning with clinical knowledge (e.g. [22, 24]) and legal applications (e.g. [32]).

Most proposals for logic based argumentation are either based on classical logic or defeasible logic. (e.g. [5, 1, 23, 15]). Classical logic has many advantages over defeasible logic programming for representing and reasoning with knowledge including syntax, proof theory and semantics for the intuitive language incorporating negation, conjunction, disjunction and implication. However, for argumentation, it is computationally challenging to generate arguments from a knowledgebase 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 [20]. The difficult nature of argumentation has been underlined by studies concerning the complexity of finding individual arguments [31] and the complexity of finding argument trees [25]. Furthermore, encodation of these tasks as quantified Boolean formulae also indicate that development of algorithms is a difficult challenge [9].

There exist techniques based on Assumption Truth Maintenance Systems (ATMS) that are closely related to the problem of generating arguments. In [2], it has been suggested that the Davis and Putnam method for theorem proving could be harnessed for generating arguments. The Davis and Putnam method is related to the connection graph method in that they are based on resolution. Whilst, we are not aware of algorithms for generating arguments and counterarguments using the Davis and Putnam, it has been shown, in [11], that the Davis and Putnam method can be used for finding minimal inconsistent sets of formulae.

Algorithms for generating arguments and counterarguments (canonical undercuts) have been given in a proposal that is based on a SAT solver [7]. As with our approach, their approach is based on finding proofs by contradiction. However, an advantage with our approach is that the combination of connection graph method and support tree construction offers a means to generate arguments and counterarguments for the first-order case.

In this paper, we have proposed the use of a connection graph approach as a way of ameliorating the computation cost. The framework we have presented

focuses the search for arguments on a part of the knowledgebase rather that the whole knowledgebase without affecting completeness. Using graph structures for this part of the knowledgebase it applies a search strategy to generate arguments for claims that are clauses and also extends to generating counterarguments. We have provided theoretical results to ensure the correctness of the proposal, and we have provided a prototype software inplementation of this proposal to indicate the potential advantages of the approach. The work in this paper therefore substantially extends our previous papers on using connection graphs to cut down the search space for arguments [17] and a framework for finding arguments with claims that are restricted to literals [16]. This paper therefore subsumes these previous papers by generating arguments with claims that are clauses, and by generating canonical undercuts for an argument. In future work, we will extend the empirical evaluation, and extend the theory and algorithms for dealing with arbitrary formulae as claims of arguments. We will also extend the framework to generate arguments and canonical undercuts from first-order knowledge based on the generalised notion of a support tree. In previous work [18] we have developed our framework to provide arguments for first-order literals, where the knowledgebase is first-order clauses. So, the next step is to combine that work with the work in this paper to generate arguments where the claim is a prenex clause.

Appendix

Proposition 1. *Let Ψ be a set of clauses and $\alpha = a_1 \vee \ldots \vee a_n$ be a clause. Then, $\Psi \vdash \alpha$ iff there is a linear deduction $\{\delta_1, \ldots, \delta_n\} \in \mathsf{Deductions}(\Psi \cup \{\overline{a_1}, \ldots, \overline{a_n}\})$ and δ_n is the empty clause.*

Proof: Follows from theorem 1. □

Lemma 1. *Let Ψ be a set of clauses and let $D \in \mathsf{Deductions}(\Psi)$. Then, for all $a \in \mathsf{Literals}(D)$ there is an $\alpha \in \Psi \cap D$ such that $a \in \mathsf{Disjuncts}(\alpha)$.*

Proof: Follows from condition (1) of the definition of a linear deduction. □

Lemma 2. *Let $\langle \Phi, \alpha \rangle$ be an argument where Φ is a set of clauses and $\alpha = a_1 \vee \ldots \vee a_n$ is a clause. Then, there is at least one $a_j \in \mathsf{Disjuncts}(\alpha)$ such that there is a $\beta \in \Phi$ with $a_j \in \mathsf{Disjuncts}(\beta)$.*

Proof: $\Phi \vdash \alpha$ and so there is a linear deduction D for α from Φ. Then by lemma 1, there is a $a_j \in \mathsf{Disjuncts}(\alpha)$ such that there is a $\beta \in \Phi$ with $a_j \in \mathsf{Disjuncts}(\beta)$. □

Lemma 3. *Let Ψ be a set of clauses and let $D \in \mathsf{Deductions}(\Psi)$ be such that $D = \{\delta_1, \ldots, \delta_n\}$. Then, for all $a \in \mathsf{Literals}(D) \setminus \mathsf{Disjuncts}(\delta_n)$ there is an $\alpha \in D \cap \Psi$ and an $\alpha' \in D \cap \Psi$ such that $\mathsf{Attacks}(\alpha, \alpha') = a$.*

Proof: Follows from condition (2) of definition 13. □

Lemma 4. *Let $\langle \Phi, \alpha \rangle$ be an argument. Then, there is no $\beta \in \Phi$ such that β is a tautology.*

Proof: Let $\langle \Phi, \alpha \rangle$ be an argument and let $\beta \in \Phi$ be a tautology. Then $\Phi^+ \equiv \Phi \cup \{\neg\alpha\}$ is a minimal inconsistent set and also $\beta \in \Phi^+$ and $\beta \neq \neg\alpha$ because β is a clause consisting of a disjunction of more than one literals while $\neg\alpha$ is a conjuction of one or more literals. So, $\Phi^+ \setminus \{\beta\} \subset \Phi^+$ and $\Phi^+ \setminus \{\beta\} \vdash \neg\beta \vdash \bot$, hence Φ^+ is not a minimal inconsistent set which contradicts the assumption that $\langle \Phi, \alpha \rangle$ is an argument. □

Lemma 5. *Let Φ be a minimal inconsistent set of clauses. Then for all $\psi \in \Phi$, for all $a \in \mathsf{Disjuncts}(\psi)$ there is a $\Phi' \subset \Phi$ such that $\langle \Phi', \overline{a} \rangle$ is an argument. For this Φ' it holds that $\Phi' \subseteq \Phi \setminus \{\psi\}$.*

Proof: Let Φ be a minimal inconsistent set of clauses. Let for a clause ψ, $\overline{\psi}$ be the CNF of $\neg\psi$. Then, for all $\langle \Phi \setminus \{\psi\}, \overline{\psi} \rangle$ is an argument hence $\Phi \setminus \{\psi\}$ is consistent and $\Phi \setminus \{\psi\} \vdash \overline{\psi}$. Let for $\psi \in \Phi$, $a \in \mathsf{Disjuncts}(\psi)$. Then $\overline{a} \in \mathsf{Conjuncts}(\overline{\psi})$ so either $\langle \Phi \setminus \{\psi\}, \overline{a} \rangle$ is an argument or there is a subset Φ' of $\Phi \setminus \{\psi\}$ that is minimal for entailing $\overline{a}$ and is also consistent hence $\langle \Phi', \overline{a} \rangle$ is an argument. □

Lemma 6. *If Φ is a minimal inconsistent set of clauses then for all $\phi \in \Phi$, for all $a \in \mathsf{Disjuncts}(\phi)$, there is a $\phi' \in \Phi$ with $\overline{a} \in \mathsf{Disjuncts}(\phi)$.*

Proof: Let Φ be a minimal inconsistent set of clauses and let $\phi \in \Phi$ be such that $a \in \mathsf{Disjuncts}(\phi)$. To give a proof by contradiction suppose there is no $\phi' \in \Phi$ with $\overline{a} \in \mathsf{Disjuncts}(\phi')$. By lemma 4, $\overline{a} \notin \mathsf{Disjuncts}(\phi)$. Let $\overline{\phi}$ be the CNF of $\neg\phi$. Then, $\langle \Phi \setminus \{\phi\}, \overline{\phi} \rangle$ is an argument and $\overline{a} \in \mathsf{Conjuncts}(\overline{\phi})$, so $\Phi \setminus \{\phi\} \vdash \overline{a}$. If $\Phi' \subseteq \Phi \setminus \{\phi\}$ is a minimal set that entails $\overline{a}$, then $\langle \Phi', \overline{a} \rangle$ is an argument, but there is no $\phi' \in \Phi'$ such that $\overline{a} \in \mathsf{Disjuncts}(\phi')$ which contradicts lemma 2. □

Corollary 8. *If Φ is a minimal inconsistent set of clauses then for all $a \in \mathsf{Literals}(\Phi)$ there is a $\Phi' \subset \Phi$ such that $\langle \Phi', a \rangle$ is an argument and a $\Phi'' \subset \Phi$ such that $\langle \Phi'', \overline{a} \rangle$ is an argument.*

Proof: Follows from lemmas 5 and 6. □

Proposition 2. *Let α be a tautology. Then, $\langle \Phi, \alpha \rangle$ is an argument iff $\Phi = \emptyset$.*

Proof: Let $\langle \Phi, \alpha \rangle$ be an argument where α is a tautology. Then $\Phi^+ \equiv \Phi \cup \{\neg\alpha\}$ is a minimal inconsistent set, but $\neg\alpha \vdash \bot$, so $\{\neg\alpha\}$ is a minimal inconsistent set and hence $\Phi^+ = \Phi \cup \{\neg\alpha\} = \{\neg\alpha\}$. Also, $\neg\alpha \notin \Phi$ because Φ is consistent since it is a support for an argument and $\neg\alpha \vdash \bot$, so $\Phi \cap \{\neg\alpha\} = \emptyset$ and $\Phi \cup \{\neg\alpha\} = \{\neg\alpha\}$ holds iff $\Phi = \emptyset$. □

Proposition 3. *Let Ψ be a minimal inconsistent set of clauses and let $(N, A) = \mathsf{Closed}(\Psi)$. Then $N = \Psi$.*

Proof: Ψ is an inconsistent set and so there is a Γ in $\mathsf{Deductions}(\Psi)$, $\Gamma = \{\gamma_1, \ldots, \gamma_n\}$, where $\gamma = \gamma_n$ is the empty clause. Also because Ψ is a minimal inconsistent set, $\Psi \subseteq \Gamma$. Then:
(1) For all $\gamma_i \in \Gamma$, for all $a \in \mathsf{Disjuncts}(\gamma_i)$ there is a $\gamma_i' \in \Gamma \cap \Psi$ (possibly $\gamma_i' = \gamma_i$) such that $a \in \mathsf{Disjuncts}(\gamma_i')$.
(2) For all $\gamma_i \in \Gamma$, for all $a \in \mathsf{Disjuncts}(\gamma_i)$ there is a $\gamma_j \in \Gamma$ with $\overline{a} \in \mathsf{Disjuncts}(\gamma_j)$.

From (1) and (2) follows

(3) For all $\gamma_i \in \Gamma$ and for all $a \in \mathsf{Disjuncts}(\gamma_i)$ there are $\gamma_i' \in \Gamma \cap \Psi$ and $\gamma_j' \in \Gamma \cap \Psi$ such that $a \in \mathsf{Disjuncts}(\gamma_i')$ and $\overline{a} \in \mathsf{Disjuncts}(\gamma_j')$

Using the above we show that the following holds:

(4) For all $\gamma_i' \in \Gamma \cap \Psi$ and for all $a \in \mathsf{Disjuncts}(\gamma_i')$ there is a $\gamma_j' \in \Gamma \cap \Psi$ such that $\mathsf{Attacks}(\gamma_i', \gamma_j') = a$

From (3), for all $\gamma_i' \in \Gamma \cap \Psi$ and for all $a \in \mathsf{Disjuncts}(\gamma_i')$ there is a $\gamma_j' \in \Gamma \cap \Psi$ such that $a \in \mathsf{Preattacks}(\gamma_i', \gamma_j')$. If for all such γ_i', γ_j', $\mathsf{Attacks}(\gamma_i', \gamma_j') = \mathit{null}$, then for all γ_j' with $\overline{a} \in \mathsf{Disjuncts}(\gamma_j')$, $|\mathsf{Preattacks}(\gamma_i', \gamma_j')| > 1$, and so for all γ_j' with $\overline{a} \in \mathsf{Disjuncts}(\gamma_j')$ there is a $b \in \mathsf{Disjuncts}(\gamma_i')$ such that $\overline{b} \in \mathsf{Disjuncts}(\gamma_j')$. Since $\Gamma \cap \Psi$ is a minimal inconsistent set, then $\Gamma' = \Gamma \cap \Psi \setminus \{\gamma_i'\} \vdash \neg\gamma_i'$ and Γ' is consistent. So, $\Gamma' \vdash \overline{a} \in \mathsf{Conjuncts}(\neg\gamma_i)$ and there are $\gamma_j' \in \Gamma'$ with $\overline{a} \in \mathsf{Disjuncts}(\gamma_j')$. For all the disjuncts b' of $\mathsf{Disjuncts}(\gamma_j') \setminus \{\overline{a}\}$, $\Gamma' \vdash \overline{b}'$. But since for all γ_j' with $\overline{a} \in \mathsf{Disjuncts}(\gamma_j')$ there is a $b \in \mathsf{Disjuncts}(\gamma_i')$ such that $\overline{b} \in \mathsf{Disjuncts}(\gamma_j')$ then for $b' = \overline{b}$ holds that $\Gamma' \vdash \overline{\overline{b}} = b$. Also, since $b \in \mathsf{Disjuncts}(\gamma_i')$, and $\Gamma' \vdash \neg\gamma_i$ then $\Gamma' \vdash \overline{b} \in \mathsf{Conjuncts}(\neg\gamma_i)$ which contradicts the assumption that Γ' is consistent. Hence, the assumption that for some $\gamma_i' \in \Gamma \cap \Psi$ there is an $a \in \mathsf{Disjuncts}(\gamma_i')$ for which there is no $\gamma_j' \in \Gamma \cap \Psi$ such that $\mathsf{Attacks}(\gamma_i', \gamma_j') = a$, cannot hold and hence for all $\gamma_i' \in \Gamma \cap \Psi$, for all $a \in \mathsf{Disjuncts}(\gamma_i')$, there is $\gamma_j' \in \Gamma \cap \Psi$ such that $\mathsf{Attacks}(\gamma_i', \gamma_j') = a$ and so (4) holds.
From (4) and the definition of the closed graph follows that $N = \Psi$. □

Proposition 4. *Let $\langle \Phi, \alpha \rangle$ be an argument where α is a clause. If $\mathsf{Query}(\Delta, \alpha)$ is such that $\mathsf{Nodes}(\mathsf{Query}(\Delta, \alpha)) \neq \emptyset$, then $\Phi \subset \mathsf{Nodes}(\mathsf{Query}(\Delta, \alpha))$.*

Proof: $\langle \Phi, \alpha \rangle$ is an argument, hence $\Phi \cup \{\neg\alpha\}$ is a minimal inconsistent set and so for some $C \subseteq \mathsf{Conjuncts}(\neg\alpha)$, $\Phi \cup C$ is a minimal inconsistent set. Then by proposition 3 if $\mathsf{Closed}(\Phi \cup C) = (N, A)$ then $N = \Phi \cup C$. By the definition of the query graph, if $(N', A') = \mathsf{Query}(\Phi, \alpha)$ then (N', A') is the attack graph for $N' = \bigcup_{a_j \in \mathsf{Disjuncts}(\alpha)} \mathsf{Nodes}(\mathsf{Focal}(\Delta \cup \{\overline{a}_i \mid a_i \in \mathsf{Disjuncts}(\alpha)\}, \overline{a}_j))$ and so since $C \subseteq \{\overline{a}_i \mid a_i \in \mathsf{Disjuncts}(\alpha)\}$ and $\Phi \subseteq \Delta$ and in both (N, A), (N', A') the $\mathsf{Attacks}$ function defines the arcs, we get that (1) is a subset of (2) where (1) and (2) are defined as follows:

$$(1) = \bigcup_{a_k \in C} \mathsf{Nodes}(\mathsf{Focal}(\Phi \cup C, \overline{a}_k))$$
$$(2) = \bigcup_{a_j \in \mathsf{Disjuncts}(\alpha)} \mathsf{Nodes}(\mathsf{Focal}(\Delta \cup \{\overline{a}_i \mid a_i \in \mathsf{Disjuncts}(\alpha)\}, \overline{a}_j))$$

so $N \subseteq N'$ and because $C \subseteq \{\overline{a}_i \mid a_i \in \mathsf{Disjuncts}(\alpha)\}$ it holds that $N \setminus C \subseteq N'$ and hence $\Phi \subset \mathsf{Nodes}(\mathsf{Query}(\Delta, \alpha))$. □

Corollary 1. *Let Ψ be a minimal inconsistent set of clauses. Then for all $\gamma \in \Psi$, $\mathsf{Focal}(\Psi, \gamma)$ is non-empty.*

Proof: From proposition 3 follows that if $\mathsf{Closed}(\Psi) = (N, A)$ then $\Psi = N$ and so for all $\gamma \in \Psi$, $\gamma \in N$ and since $N \neq \emptyset$ then by the definition of the focal graph $\mathsf{Focal}(\Psi, \gamma)$ is a non-empty graph. □

Proposition 5. *Let (N, A, f) be a complete presupport tree for Δ, α and a. Then, there is no $\overline{b} \in \mathsf{Literals}(\{f(x) \mid x \in N\} \setminus \{\overline{a}\})$ such that $b \in \mathsf{Disjuncts}(\alpha)$.*

Proof: Let (N, A, f) be a complete presupport tree for Δ, α and a. If for some $x \in N$, $\overline{b} \in \mathsf{Disjuncts}(f(x))$ is such that $b \in \mathsf{Disjuncts}(\alpha)$ then for $\overline{b} \in \mathsf{Disjuncts}(f(x))$ exactly one of conditions *i*), *ii*) and *iii*) of definition 21 holds. *i*) cannot hold for $\overline{b} \in \mathsf{Disjuncts}(f(x))$ because if it did then according to the condition, $\overline{b} \in \mathsf{Disjuncts}(\alpha)$ and also by assumption $b \in \mathsf{Disjuncts}(\alpha)$ and this contradicts the assumption that α is not a tautology. If condition *ii*) holds for $\overline{b} \in \mathsf{Disjuncts}(f(x))$ then $\overline{b} \in \mathsf{AncestorLabels}(x)$ and so there is an arc $(w, w') \in A$ where $w' \in \mathsf{Ancestors}(x)$ such that $\mathsf{Attacks}(f(w), f(w')) = \overline{b}$. Then, for w', and $b \in \mathsf{Disjuncts}(f(w'))$ it holds that there is a child w if w' such that $\mathsf{Attacks}(f(w), f(w')) = \overline{b}$ and also that $b \in \mathsf{Disjuncts}(\alpha)$ so both conditions *i*) and *iii*) of definition 21 hold for $b \in \mathsf{Disjuncts}(f(w'))$ which contradicts the assumption that (N, A, f) is a complete presupport tree for Δ, α and a. If condition *iii*) holds for $\overline{b} \in \mathsf{Disjuncts}(f(x))$ then there is a child x' of x such that $\mathsf{Attacks}(f(x'), f(x)) = b$ and then for x' and $b \in \mathsf{Disjuncts}(f(x'))$ both conditions *i*) and *ii*) of definition 21 hold which contradicts the assumption that (N, A, f) is a complete presupport tree for Δ, α and a. □

Proposition 6. *Let (N, A, f) be a complete presupport tree for Δ, α and a. Then, there is no $x \in N$ such that both $b \in \mathsf{AncestorLabels}(x)$ and $\overline{b} \in \mathsf{AncestorLabels}(x)$ hold.*

Proof: Let literal b and its complement $\overline{b}$ be such that there are arcs (w, w'), (y, y') on the same branch of (N, A, f) where w' is the parent of w and y' is the parent of y such that $\mathsf{Attacks}(f(w), f(w')) = b$ and $\mathsf{Attacks}(f(y), f(y')) = \overline{b}$. To give a proof by contradiction we show that under these conditions (N, A, f) cannot satisfy the conditions for a complete presupport tree. Assume that $w \in \mathsf{Ancestors}(y)$. If w' is the root then $\overline{b} = \overline{a}$ and $b \in \mathsf{Disjuncts}(\alpha)$ and so for $b \in \mathsf{Disjuncts}(f(y'))$, since there is a child y such that $\mathsf{Attacks}(f(y), f(y')) = \overline{b}$ both conditions *i*) and *iii*) of definition 21 hold and so (N, A, f) is not complete. If w' is not the root then $b \notin \mathsf{Disjuncts}(\alpha)$ and so condition *iii*) of definition 21 holds for $\overline{b} \in \mathsf{Disjuncts}(f(w'))$ and there is a child w of w' such that $\mathsf{Attacks}(f(w), f(w')) = b$. Then, for $b \in \mathsf{Disjuncts}(f(y'))$ it holds that there is a $w' \in \mathsf{Ancestors}(y')$ such that $\mathsf{Attacks}(f(w), f(w')) = b$ and also a child y such

that $\mathsf{Attacks}(f(y), f(y')) = \overline{b}$ and so both conditions *ii*) and *iii*) of definition 21 hold for $b \in \mathsf{Disjuncts}(f(y'))$ and so (N, A, f) is not complete. □

Proposition 7. *Let (N, A, f) be a complete presupport tree for Δ, α and a. Then, there is no $x \in N$ such that $b \in \mathsf{Disjuncts}(f(x))$ and $\overline{b} \in \mathsf{AncestorLabels}(x)$.*

Proof: Let (N, A, f) be a complete presupport tree for Δ, α and a and let $x \in N$ be such that $b \in \mathsf{Disjuncts}(f(x))$ and $\overline{b} \in \mathsf{AncestorLabels}(x)$. Then for $b \in \mathsf{Disjuncts}(f(x))$ at least one of the conditions of definition 21 holds. By proposition 5, $b \notin \mathsf{Disjuncts}(\alpha)$ and so condition *i*) cannot hold for $b \in \mathsf{Disjuncts}(f(x))$. Then for $b \in \mathsf{Disjuncts}(f(x))$ one of conditions *ii*) or *iii*) of definition 21 holds. Condition *ii*) cannot hold for $b \in \mathsf{Disjuncts}(f(x))$ because then $b \in \mathsf{AncestorLabels}(x)$ and $\overline{b} \in \mathsf{AncestorLabels}(x)$ which contadicts proposition 6. Condition *iii*) cannot hold either because then there is a child x' of x such that $\mathsf{Attacks}(f(x'), f(x)) = \overline{b}$ and for x' there is more than one $w \in \mathsf{Ancestors}(x')$ such that $\mathsf{Attacks}(f(w), f(w')) = \overline{b}$. Hence, no condition of definition 21 can hold for $\overline{b} \in \mathsf{Disjuncts}(f(x'))$ when there is an $x \in N$ such that $b \in \mathsf{Disjuncts}(f(x))$ and $\overline{b} \in \mathsf{AncestorLabels}(x)$ and this contradicts the assumption that (N, A, f) is a complete presupport tree for Δ, α and a. □

Proposition 8. *Let (N, A, f) be a complete presupport tree for Δ, α and a. Then for a node $x \in N$, where x' is the parent of z, $\mathsf{Attacks}(f(x), f(x')) = \mathsf{Attacks}(\mathsf{SubtreeRes}(x), f(x'))$.*

Proof: Let (N, A, f) be a complete presupport tree for Δ, α and a. From the way $\mathsf{SubtreeRes}$ is defined follows that for all x, x' where x' is the parent of x, $\mathsf{Attacks}(f(x), f(x')) \in \mathsf{Disjuncts}(\mathsf{SubtreeRes}(x))$. Then, in order to prove that $\mathsf{Attacks}(f(x), f(x')) = \mathsf{Attacks}(\mathsf{SubtreeRes}(x), f(x'))$ it is sufficient to prove that it cannot hold that $|\mathsf{Preattacks}(\mathsf{SubtreeRes}(x), f(x'))| > 1$. To give a proof by contradiction assume that (N, A, f) is a complete presupport tree for Δ, α and a and for an arc $(x, x') \in A$ which is such that $\mathsf{Attacks}(f(x), f(x')) = b$, it holds that $|\mathsf{Preattacks}(\mathsf{SubtreeRes}(x), f(x'))| > 1$. Then, it holds that $b \in \mathsf{Preattacks}(\mathsf{SubtreeRes}(x), f(x'))$ and there is a $c \neq b$ which is such that $c \in \mathsf{Preattacks}(\mathsf{SubtreeRes}(x), f(x'))$. Since $c \in \mathsf{Disjuncts}(\mathsf{SubtreeRes}(x))$, there is a $w \in \mathsf{Subtree}(x)$ such that $c \in \mathsf{Disjuncts}(f(w))$ and there is no arc (w, w') where in $w, w' \in \mathsf{Subtree}(x)$ such that $\mathsf{Attacks}(f(w), f(w')) = c$ or $\mathsf{Attacks}(f(w'), f(w)) = \overline{c}$. Because (N, A, f) is a complete presupport tree, for $c \in \mathsf{Disjuncts}(f(w))$ either $c \in \mathsf{AncestorLabels}(w)$ or $c \in \mathsf{Disjuncts}(\alpha)$ holds. If $c \in \mathsf{AncestorLabels}(w)$ then it holds that $c \in \mathsf{AncestorLabels}(x')$ because c does not label any arc in $\mathsf{Subtree}(x')$ and if $c \in \mathsf{AncestorLabels}(x')$ holds it contradicts proposition 7 because $\overline{c} \in \mathsf{Disjuncts}(f(x'))$. Hence, $c \in \mathsf{AncestorLabels}(w)$ does not hold and so $c \in \mathsf{Disjuncts}(\alpha)$ must hold in order for (N, A, f) to be complete. If x' is not the root node of (N, A, f) then since $\overline{c} \in \mathsf{Disjuncts}(f(x'))$ it holds that $\overline{c} \in \mathsf{Literals}(\{f(x) \mid x \in N\} \setminus \{\overline{a}\})$ and if $c \in \mathsf{Disjuncts}(\alpha)$ then this contradicts proposition 5. If x' is the root then $f(x')$ is a literal and so it cannot hold that $|\mathsf{Preattacks}(\mathsf{SubtreeRes}(x), f(x'))| > 1$. Hence, w with $c \in$

$\mathsf{Disjuncts}(f(w))$ does not satisfy any of the conditions of definition 21 which contradicts the assumption that (N, A, f) is a complete presupport tree. So it cannot hold that $|\mathsf{Preattacks}(\mathsf{SubtreeRes}(x), f(x'))| > 1$ and so it follows that $\mathsf{Attacks}(f(x), f(x')) = \mathsf{Attacks}(\mathsf{SubtreeRes}(x), f(x'))$. □

Proposition 9. *Let (N, A, f) be a complete presupport tree for Δ, α and a. Then for a node $z \in N$ with $\mathsf{Children}(z) = \{x_1, \ldots, x_n\}$, $\mathsf{SubtreeRes}(z) = f(z) \bullet \mathsf{SubtreeRes}(x_1) \bullet \ldots \bullet \mathsf{SubtreeRes}(x_n)$.*

Proof: Let (N, A, f) be a complete presupport tree for Δ, α and a and let $z \in N$. Then for every $v_i \in \mathsf{Children}(z)$ it holds that $\mathsf{Attacks}(f(v_i), f(z)) = b_i$ for some $b_i \in \mathsf{Disjuncts}(f(v_i))$ and because of the constraints for a complete presupport tree there is no other arc (w, w') on the branch where v_i belongs such that $\mathsf{Attacks}(f(w), f(w')) = b_i$ or $\mathsf{Attacks}(f(w), f(w')) = \overline{b}_i$. Hence, for neither b_i nor $\overline{b}_i$ are there any $w, w' \in \mathsf{Subtree}(v_i)$ such that $\mathsf{Attacks}(f(w), f(w')) = b_i$ or $\mathsf{Attacks}(f(w), f(w')) = \overline{b}_i$ and so by the definition of $\mathsf{SubtreeRes}(v_i)$, it holds that $b_i \in \mathsf{Disjuncts}(\mathsf{SubtreeRes}(v_i))$ for all $v_i \in \mathsf{Children}(z)$. Then for $b_i = \mathsf{Attacks}(f(v_i), f(z))$ for all $v_i \in \mathsf{Children}(z)$, $\mathsf{SubtreeRes}(v_1) \bullet \ldots \bullet \mathsf{SubtreeRes}(v_j) \bullet f(z)$ can be re-written to:

$$\bigvee \bigcup_{v_i} \big((\mathsf{Disjuncts}(\mathsf{SubtreeRes}(v_i)) \cup \mathsf{Disjuncts}(f(z))) \setminus \{b_i, \overline{b}_i\} \big)$$

which for $B_i = \{f(w_i) \mid w_i \in \mathsf{Subtree}(v_i)\}$ and $A_i = \{\mathsf{Attacks}(f(w_i), f(w_i')) \mid w_i, w_i' \in \mathsf{Subtree}(v_i)\}$ for all $v_i \in \mathsf{Children}(z)$ can be re-written to

$$\bigvee \bigcup_{v_i} \big((((\mathsf{Literals}(B_i) \setminus A_i) \cup \mathsf{Disjuncts}(f(z))) \setminus \{b_i, \overline{b}_i\} \big) =$$

and $A_i \cap \mathsf{Disjuncts}(f(z)) = \emptyset$ otherwise the conditions for a complete presupport tree would not be satisfied, so the equation above can be re-written to

$$\bigvee \bigcup_{v_i} \big(\mathsf{Literals}(B_i) \cup \mathsf{Disjuncts}(f(z))) \setminus (A_i \cup \{b_i, \overline{b}_i\}) \big) =$$

$$\bigvee \big(\mathsf{Literals}(\{f(w) \mid w \in \mathsf{Subtree}(z)\}) \setminus \{\mathsf{Attacks}(f(w), f(w')) \mid w, w' \in \mathsf{Subtree}(z)\} \big)$$

which by definition is equal to $\mathsf{SubtreeRes}(z)$. □

Proposition 10. *Let (N, A, f) be a complete presupport tree for Δ, α and a. Then for a node $z \in N$, there is a linear deduction $\{\delta_1, ..., \delta_n\} \in \mathsf{Deductions}(\{f(w) \mid w \in \mathsf{Subtree}(z)\})$ where $\mathsf{SubtreeRes}(z) \equiv \delta_n$ and $\{f(w) \mid w \in \mathsf{Subtree}(z)\} \subseteq \{\delta_1, ..., \delta_n\}$.*

Proof: We use a proof by induction. For the base case, let $z \in N$ be a leaf and let $f(z) = \delta_1$. So $\mathsf{Subtree}(z) = \{z\}$, and $\{f(w) \mid w \in \mathsf{Subtree}(z)\} = \{f(z)\} = \{\delta_1\}$ and by definition, for a leaf node $\mathsf{SubtreeRes}(z) = f(z)$. Therefore, there is a linear deduction $\{\delta_1\} \in \mathsf{Deductions}(\{f(w) \mid w \in \mathsf{Subtree}(z)\}) =$

$\mathsf{Deductions}(\{\delta_1\})$ where $\mathsf{SubtreeRes}(z) = \delta_1$ and $\{f(w) \mid w \in \mathsf{Subtree}(z)\} = \{\delta_1\} \subseteq \{\delta_1\}$. Therefore the proposition holds for the base case.
For the inductive step, let $z \in N$ be a non-leaf node with children $v_1, .., v_j$, and let $f(z) = \delta_k$. Also, assume that the proposition holds for each child $v_i \in \{v_1, ..., v_j\}$. Hence, for each child v_i, we assume that there is a linear deduction $\{\delta_1^{v_i}, ..., \delta_{n_i}^{v_i}\} \in \mathsf{Deductions}(\{f(w) \mid w \in \mathsf{Subtree}(v_i)\})$ where $\mathsf{SubtreeRes}(v_i) \equiv \delta_{n_i}^{v_i}$ and $\{f(w) \mid w \in \mathsf{Subtree}(v_i)\} \subseteq \{\delta_1^{v_i}, ..., \delta_{n_i}^{v_i}\}$.
Therefore, each of these linear deductions can be put into a linear deduction for δ_n as follows,

$$\{\delta_1^{v_1}, ..., \delta_{n_1}^{v_1}, ..., \delta_1^{v_j}, ..., \delta_{n_j}^{v_j}, \delta_k, \delta_n\}$$

where $\delta_n \equiv \delta_{n_1}^{v_1} \bullet ... \bullet \delta_{n_j}^{v_j} \bullet \delta_k$. Then by proposition 9 follows that $\delta_n = \mathsf{SubtreeRes}(z)$. From the structure of the tree, we have that $\{f(w) \mid w \in \mathsf{Subtree}(v_i)\} \subseteq \{f(w) \mid w \in \mathsf{Subtree}(z)\}$. Then, from the constraints on linear deduction, we have the following.

$$\{f(w) \mid w \in \mathsf{Subtree}(z)\} \subseteq \{\delta_1^{v_1}, ..., \delta_{n_1}^{v_1}, ..., \delta_1^{v_j}, ..., \delta_{n_j}^{v_j}, \delta_k, \delta_n\}$$

Therefore, there is a linear deduction, as follows, where $\mathsf{SubtreeRes}(z) \equiv \delta_n$.

$$\{\delta_1^{v_1}, ..., \delta_{n_1}^{v_1}, ..., \delta_1^{v_j}, ..., \delta_{n_j}^{v_j}, ..., \delta_k, \delta_n\} \in \mathsf{Deductions}(\{f(w) \mid w \in \mathsf{Subtree}(z)\})$$

so the proposition holds for the inductive case. □

Corollary 2 *Let (N, A, f) be a complete presupport tree for Δ, α and a. Then for a node $z \in N$, $\{f(w) \mid w \in \mathsf{Subtree}(z)\} \vdash \mathsf{SubtreeRes}(z)$.*

Proof: Follows from proposition 10. □

Lemma 7. *Let $\langle \Phi, \alpha \rangle$ be an argument. Then, there is a consistent presupport tree (N, A, f) for Δ, α and a such that $\Phi = \{f(x) \mid x \in N\} \setminus \{\overline{a}\}$.*

Proof: Let $\langle \Phi, \alpha \rangle$ be an argument. Then there is a clause α' such that $\Phi \vdash \alpha'$ where $\mathsf{Disjuncts}(\alpha') \subseteq \mathsf{Disjuncts}(\alpha)$ and there is no α'' such that $\mathsf{Disjuncts}(\alpha'') \subset \mathsf{Disjuncts}(\alpha')$ and $\Phi \vdash \alpha''$. Then, $\Phi \cup \{\neg\alpha'\}$ is a minimal inconsistent set. Let C be the set of complements of the disjuncts of α' i.e. $C = \{\overline{b} \mid b \in \mathsf{Disjuncts}(\alpha')\}$. Then, C contains the complements of the literals of α that appear in Φ. $C = \{\overline{b} \mid b \in \mathsf{Disjuncts}(\alpha) \cap \mathsf{Literals}(\Phi)\}$ and for all $\overline{b} \in C$, $\overline{b} \in \mathsf{Conjuncts}(\neg\alpha')$. Because $\Phi \cup \{\neg\alpha'\}$ is a minimal inconsistent set, and by the way C is defined, then if $\Gamma = \Phi \cup C$, it holds that Γ is a minimal inconsistent set of clauses.
By corollary 8, for all $b \in \mathsf{Literals}(\Gamma)$ there is a $\Gamma' \subset \Gamma$ such that $\langle \Gamma', b \rangle$ is an argument and so there is a linear deduction $D = \{\beta_1, \ldots, \beta_n\}$ such that $\beta_n = b$ and $\Gamma' \subseteq D$. By lemma 2 there is a $\beta \in \Gamma' \cap D$ such that $b \in \mathsf{Disjuncts}(\beta)$.
Let for $b \in \mathsf{Literals}(\Gamma)$, $\mathcal{D}(b) = \{\beta_1, \ldots, \beta_n\} \in \mathsf{Deductions}(\Gamma')$ be one of these deductions where $\beta_n = b$ and $\beta_{n-1} = \beta \in \Gamma'$ with $b \in \mathsf{Disjuncts}(\beta)$. Also, let for each such $\mathcal{D}(b)$, $\mathsf{Formula}(\mathcal{D}(b))$ be the function that returns β_{n-1} i.e. $\mathsf{Formula}(\mathcal{D}(b)) = \beta_{n-1}$. Hence for all $\mathcal{D}(b)$, $\mathsf{Formula}(\mathcal{D}(b))$ is a clause from the original set of clauses that contains b as a disjunct: $\mathsf{Formula}(\mathcal{D}(b)) \in \Gamma$, and

$b \in \mathsf{Disjuncts}(\mathsf{Formula}(\mathcal{D}(b)))$.
Let N be a set of nodes each of which is defined using function $\mathsf{TreeNode}$ as follows. For a node $x \in N$ and a linear deduction $\mathcal{D}(b)$ which is as described above, $\mathsf{TreeNode}(\mathcal{D}(b), x)$ defines a node y that contains $\mathcal{D}(b)$ and has x as its parent. For a node $x \in N$, $\mathsf{Ancestors}(x)$ returns the set of nodes that precede x in a branch. We use this mapping to construct a consistent presupport tree where $\mathsf{Formula}(\mathcal{D}(b))$ will give the clause representation of each node of the tree.
Since Γ is a minimal inconsistent set, there is a $\mathcal{D}(\bot)$ such that $\Gamma \subseteq \mathcal{D}(\bot)$. Because $\mathsf{Disjuncts}(\bot) = \emptyset$ it holds that $\mathsf{Disjuncts}(\bot) \subset \mathsf{Disjuncts}(\gamma)$ for any $\gamma \in \Gamma$ then any clause could be $\mathsf{Formula}(\mathcal{D}(\bot))$. Let $\mathsf{Formula}(\mathcal{D}(\bot)) = \overline{a}$ for some $\overline{a} \in \Gamma \cap C$. Then, let $x_0 = \mathsf{TreeNode}(\mathcal{D}(\bot), null)$ be the root node of a tree (N, A).
By corollary 8, for all the disjuncts b of a clause $\gamma \in \Gamma$ there is a $\Gamma_{\overline{b}} \subset \Gamma$ such that $\langle \Gamma_{\overline{b}}, \overline{b} \rangle$ is an argument and so there is a $\mathcal{D}(\overline{b}) \in \mathsf{Deductions}(\Gamma_{\overline{b}})$. Then, for $\mathsf{Formula}(\mathcal{D}(\bot)) = \overline{a} \in \Gamma$ there is a $\mathcal{D}(\overline{\overline{a}}) = \mathcal{D}(a) = \{\alpha_1, \ldots, \alpha_n\}$ where $\alpha_n = \overline{\overline{a}} = a$ and $\mathsf{Formula}(\mathcal{D}(a)) = \alpha_{n-1} \in \Gamma$ is such that $a \in \mathsf{Disjuncts}(a_{n-1})$. Then let $x_a = \mathsf{TreeNode}(\mathcal{D}(a), x_0)$ be the child node of x_0. Then, so far $N = \{x_a, x_0\}$ and $A = \{(x_a, x_0)\}$.
In the same way, for $\mathsf{Formula}(\mathcal{D}(a))$, for all $b_i \in \mathsf{Disjuncts}(\mathsf{Formula}(\mathcal{D}(a))$ there are arguments $\langle \Gamma_{\overline{b}_i}, \overline{b}_i \rangle$ and $\mathcal{D}(\overline{b}_i) \in \mathsf{Deductions}(\Gamma_{\overline{b}_i})$ as described above. Then for all $b_i \in \mathsf{Disjuncts}(\mathsf{Formula}(\mathcal{D}(a))) \setminus \{a\}$, let $x_{\overline{b}_i} = \mathsf{TreeNode}(\mathcal{D}(\overline{b}_i), x_a)$ where $\mathsf{Attacks}(\mathsf{Formula}(\mathcal{D}(\overline{b}_i)), \mathsf{Formula}(\mathcal{D}(a))) = \overline{b}_i$. Then, if $\mathsf{Disjuncts}(\mathsf{Formula}(\mathcal{D}(a))) \setminus \{a\} = \{x_{\overline{b}_1}, \ldots, x_{\overline{b}_m}\}$, the tree so far consists of nodes $N = \{x_{\overline{b}_1}, \ldots, x_{\overline{b}_m}, x_a, x_0\}$ and arcs $A = \{(x_{\overline{b}_1}, x_a), \ldots, (x_{\overline{b}_m}, x_a), (x_a, x_0)\}$.
Continuing constructing the tree in the same way, where for a node $x_{c_i} = \mathsf{TreeNode}(\mathcal{D}(c_i), x_{d_k})$ a child node $x_{\overline{p}_j} = (\mathcal{D}(\overline{p}_j), x_{c_i})$ is created for each disjuct p_j of $\mathsf{Formula}(\mathcal{D}(c_i))$ such that:

(1) there is no $x \in \mathsf{Ancestors}(x_{c_i})$ such that $x = (\mathcal{D}(\overline{p}_j), x')$ for some $x' \in \mathsf{Ancestors}(x_{c_i})$
(2) there is no $x \in \mathsf{Ancestors}(x_{c_i})$ such that $x = (\mathcal{D}(\overline{q}), x')$ for some $x' \in \mathsf{Ancestors}(x_{c_i})$ and $\mathsf{Formula}(\mathcal{D}(\overline{q})) = \mathsf{Formula}(\mathcal{D}(c_i))$
(3) $\mathsf{Attacks}(\mathsf{Formula}(\mathcal{D}(\overline{p}_j)), \mathsf{Formula}(\mathcal{D}(c_i))) = \overline{p}_j$
(4) there is no $x \in N$ such that $x_{p_j} = (\mathcal{D}(p_j), x')$ for some $x' \in N$

we obtain a tree (N, A) which is isomorphic to a complete and consistent presupport tree (N', A', f) for $\Delta \cup C$, $\theta = a$ and $a \in \mathsf{Disjuncts}(\theta)$ where for each $x_{c_i} \in N, x_{c_i} = \mathsf{TreeNode}(\mathcal{D}(c_i), x_{d_k})$ there is a $x \in N'$ such that $f(x) = \mathsf{Formula}(\mathcal{D}(c_i))$ and $\{f(x) \mid x \in N'\} = \Gamma$.
Conditions (1) to (3) in the construction of (N, A) ensure that the conditions of the definition for a presupport tree and conditions *ii*) and *iii*) of the definition for a complete presupport tree are satisfied for tree (N', A', f) while condition (4) ensures that the definition for a consistent presupport tree is satisfied for (N', A', f). Because θ is a unit clause, conditions *ii*) and *iii*) of definition 21 are sufficient for (N', A', f) to satisfy the definition for a complete presupport tree. For the nodes $x \in N'$ that are such that $f(x) \in C$

it holds that apart from $\overline{a}$ that represents the root the rest have to be leaf nodes because they are literals and for each $\{x \in N' \mid f(x) \in C\}$, $f(x) = \overline{b}$ where $\overline{b}$ is a literal and $\mathsf{Attacks}(f(x), f(x')) = \overline{b}$ where x' is the parent of x and condition *ii*) if definition 21 is satisfied for $\overline{b} \in \mathsf{Disjuncts}(f(x))$ and condition *iii*) of the definition is satisfied for $b \in \mathsf{Disjuncts}(f(x'))$. Then, if the leaf nodes that have a literal from C assigned as their clause representation i.e. $\{x \in N' \mid f(x) \in C\}$ are removed from the tree, then the resulting tree is a complete presupport tree for Δ, α and a where for the parents x' of each removed leaf x with $f(x) = \overline{b}$ condition *i*) of the definition for a complete presupport tree holds for $b \in \mathsf{Disjuncts}(f(x'))$ because by the way C is defined it holds that $b \in \mathsf{Disjuncts}(\alpha)$. Apart from the the parents x' of each such removed leaf x no other node in the tree could be affected by removing these leaf nodes. Then the resulting tree (N'', A'', f) is a complete and consistent presupport tree for Δ, α and a and $\{f(x) \mid x \in N''\} = \{f(y) \mid y \in N'\} \setminus C = \Gamma \setminus C = \Phi$. □

Proposition 11. *Let (N, A, f) be a support tree for Δ, α and a. Then for all $z \in N$, there is no $\gamma' \in \mathcal{C}$ with $\mathsf{Disjuncts}(\gamma') \subset \mathsf{Disjuncts}(\mathsf{SubtreeRes}(z))$ and $\{f(w) \mid w \in \mathsf{Subtree}(z)\} \vdash \gamma'$.*

Proof: Let $\gamma = \mathsf{SubtreeRes}(z)$ and let $B = \{f(w) \mid w \in \mathsf{Subtree}(z)\}$. Then by proposition 10, $B \vdash \gamma$. To give a proof by contradiction assume $B \vdash \gamma'$ for some $\gamma' \in \mathcal{C}$ such that $\mathsf{Disjuncts}(\gamma') \subset \mathsf{Disjuncts}(\gamma)$. Then there is a literal $m \in \mathsf{Disjuncts}(\gamma) \setminus \mathsf{Disjuncts}(\gamma')$. Also, there is a linear deduction of γ from B, $D_\gamma \in \mathsf{Deductions}(B)$ and a linear deduction of γ' from B, $D_{\gamma'} \in \mathsf{Deductions}(B)$. Then there is a $\rho_m \in D_\gamma$ such that $m \in \mathsf{Disjuncts}(\rho_m)$ and so there is $w \in \mathsf{Subtree}(z)$ such that $f(w) \in D_\gamma$ and $m \in \mathsf{Disjuncts}(f(w))$. Since $m \in \mathsf{Disjuncts}(\gamma)$ then by the way γ is defined there is no arc $(w, w') \in A$ where $w, w' \in \mathsf{Subtree}(z)$ such that $\mathsf{Attacks}(w, w') = m$ or $\mathsf{Attacks}(w, w') = \overline{m}$. Because $m \notin \mathsf{Disjuncts}(\gamma')$ then by lemmas 1 and 3 respectively either
(1) There is no $\rho_m \in D_{\gamma'}$ with $m \in \mathsf{Disjuncts}(\rho_m)$ or
(2) There is a $\rho_m \in D_{\gamma'}$ with $m \in \mathsf{Disjuncts}(\rho_m)$ and there is a $\rho_{\overline{m}} \in D_{\gamma'}$ such that $\overline{m} \in \mathsf{Disjuncts}(\rho_{\overline{m}})$
If (1) holds then there is a $w \in \mathsf{Subtree}(z)$ with $m \in \mathsf{Disjuncts}(f(w))$ and $f(w) \notin D_{\gamma'}$. By lemma 3, for all $k \in \mathsf{Literals}(D_{\gamma'}) \setminus \mathsf{Disjuncts}(\gamma')$ there are $f(y) \in D_{\gamma'} \cap B$ and $f(y') \in D_{\gamma'} \cap B$ such that $k \in \mathsf{Disjuncts}(f(y))$ and $\overline{k} \in \mathsf{Disjuncts}(f(y'))$ and it holds that $\mathsf{Attacks}(f(y), f(y')) = k$. Also for each such y and for the rest of the disjuncts $p \in (\mathsf{Disjuncts}(f(y)) \setminus \{k\}) \setminus \mathsf{Disjuncts}(\gamma')$ because $p \in \mathsf{Literals}(D_{\gamma'}) \setminus \mathsf{Disjuncts}(\gamma')$, then by lemma 3 there is a $f(y'') \in D \cap B$ with $\overline{p} \in \mathsf{Disjuncts}(f(y''))$ such that $\mathsf{Attacks}(f(y''), f(y)) = \overline{p}$ and the same holds for the rest of the disjuncts of $f(y'')$ and also for all $f(x) \in D_{\gamma'} \cap B$ and the disjuncts of these $f(x)$ that do not appear as disjuncts of γ'. Then, for all $x \in \mathsf{Subtree}(z)$ for which $f(x) \in D_{\gamma'} \cap B$ the conditions for a complete presupport tree hold which means that there is a complete presupport tree (N', A', f') for Δ, α and a such that $\{f'(x') \mid x' \in N'\} \subseteq \{f(x) \mid x \in N\} \setminus \{f(w), f(w')\}$ and so $\{f'(x') \mid x' \in N'\} \subset \{f(x) \mid x \in N\}$ which contradicts the assumption that (N, A, f) is a minimal presupport tree for Δ, α and a.

If (2) holds, there is a $w \in \mathsf{Subtree}(z)$ with $m \in \mathsf{Disjuncts}(f(w))$ and $f(w) \in D_\gamma$ and $f(w) \in D_{\gamma'}$ and there is a $w' \in \mathsf{Subtree}(z)$ such that $f(w') \in D_{\gamma'}$ and $\overline{m} \in \mathsf{Disjuncts}(f(w'))$. Hence there are $w, w' \in \mathsf{Subtree}(z)$ with $m \in \mathsf{Disjuncts}(f(w))$ and $\overline{m} \in \mathsf{Disjuncts}(f(w'))$ but there is no arc (w, w'') where $w'' \in \mathsf{Subtree}(z)$ such that $\mathsf{Attacks}(w, w') = m$ or $\mathsf{Attacks}(w', w'') = \overline{m}$. Hence condition iii) of the definition for a complete presupport tree does not hold for either of $m \in \mathsf{Disjuncts}(f(w))$ and $\overline{m} \in \mathsf{Disjuncts}(f(w'))$. Then, because (N, A, f) is a complete presuport tree for each of $m \in \mathsf{Disjuncts}(f(w))$ and $\overline{m} \in \mathsf{Disjuncts}(f(w'))$ either condition i) or condition ii) of definition 21 must hold. Condition i) cannot hold for both $m \in \mathsf{Disjuncts}(f(w))$ and $\overline{m} \in \mathsf{Disjuncts}(f(w'))$ because then $\{m, \overline{m}\} \subset \mathsf{Disjuncts}(\alpha)$ and by the definition for a presupport tree α cannot be a tautology. Condition ii) cannot hold for both $m \in \mathsf{Disjuncts}(f(w))$ and $\overline{m} \in \mathsf{Disjuncts}(f(w'))$ because then $\{m, \overline{m}\} \subset \mathsf{AncestorLabels}(z)$ and this contradicts the assumption that (N, A, f) is a consistent presupport tree. If condition i) holds for $m \in \mathsf{Disjuncts}(f(w))$ and condition ii) holds for $\overline{m} \in \mathsf{Disjuncts}(f(w'))$, then $m \in \mathsf{Disjuncts}(\alpha)$ and there is some $w'' \in N$ (possibly $w = w''$) such that $(w', w'') \in N$ and $\mathsf{Attacks}(f(w'), f(w'')) = \overline{m}$. Then $m \in \mathsf{Disjuncts}(f(w''))$ and for $m \in \mathsf{Disjuncts}(f(w''))$ both conditions i) and iii) hold which contradicts the assumption that (N, A, f) is a complete presupport tree. Similarly if condition ii) holds for $m \in \mathsf{Disjuncts}(f(w))$ and condition i) holds for $\overline{m} \in \mathsf{Disjuncts}(f(w'))$ we get contradiction.

Because either of (1) and (2) lead to contradiction, then it cannot hold that $B \vdash \gamma'$ for some $\gamma' \in \mathcal{C}$ such that $\mathsf{Disjuncts}(\gamma') \subset \mathsf{Disjuncts}(\gamma)$. □

Proposition 12. *Let $\langle \Phi, \alpha \rangle$ be an argument. Then, there is a support tree (N, A, f) for Δ, α and a for some $a \in \mathsf{Disjuncts}(\alpha)$ such that $\Phi = \{f(x) \mid x \in N\} \setminus \{\overline{a}\}$.*

Proof: By lemma 7 there is a consistent presupport tree (N, A, f) for Δ, α and a such that $\Phi = \{f(x) \mid x \in N\} \setminus \{\overline{a}\}$. For a proof by contradiction assume that for all the consistent presupport trees (N, A, f) for Δ, α and a such that $\Phi = \{f(x) \mid x \in N\} \setminus \{\overline{a}\}$ it holds that they are non-minimal. If (N, A, f) is a non-minimal presupport tree then there is a complete presupport tree (N', A', f') for Δ, α and a such that $\{f'(x') \mid x' \in N'\} \subset \{f(x) \mid x \in N\}$. Let be z be the child of the root node of (N', A', f'). Then, because (N', A', f') is a complete presupport tree, from the conditions of definition 21 the only literals b that appear in the tree for which there are no arcs $(w, w') \in A$ such that $\mathsf{Attacks}(f(w), f(w')) = b$ or $\mathsf{Attacks}(f(w), f(w')) = \overline{b}$ are the ones that are in the disjuncts of α. Also $a \in \mathsf{Disjuncts}(f(z))$ and $a \in \mathsf{Disjuncts}(\mathsf{SubtreeRes}(z))$ and by the definition for a presupport tree a is also in the disjuncts of α. Then by the way $\mathsf{SubtreeRes}(z)$ is defined, it holds that for all $b \in \mathsf{Disjuncts}(\mathsf{SubtreeRes}(z))$, $b \in \mathsf{Disjuncts}(\alpha)$ and so $\mathsf{Disjuncts}(\mathsf{SubtreeRes}(z)) \subseteq \mathsf{Disjuncts}(\alpha)$. By proposition 10, for $z \in N'$ it holds that there is a linear deduction $\{\delta_1, \ldots, \delta_n\} \in \mathsf{Deductions}(\{f'(w) \mid w \in \mathsf{Subtree}(z)\})$ where $\delta_n = \mathsf{SubtreeRes}(z)$ and $\{f'(w) \mid w \in \mathsf{Subtree}(z)\} \subseteq \{\delta_1, \ldots, \delta_n\}$. Then, $\{f'(w) \mid w \in \mathsf{Subtree}(z)\} \vdash \mathsf{SubtreeRes}(z)$ and since it holds that $\mathsf{Disjuncts}(\mathsf{SubtreeRes}(z)) \subseteq \mathsf{Disjuncts}(\alpha)$ then it also holds that $\{f'(w) \mid$

$w \in \mathsf{Subtree}(z)\} \vdash \alpha$ and if $\Phi' = \{f'(w) \mid w \in \mathsf{Subtree}(z)\}$ then $\Phi' \subset \Phi$ and $\Phi' \vdash \alpha$ which contradicts the assumption that $\langle \Phi, \alpha \rangle$ is an argument. □

Proposition 13. *Let Δ be a set of clauses and α be a clause and let (N, A, f) be a support tree for Δ, α and a. Then, $\langle \Gamma, \alpha \rangle$ with $\Gamma = \{f(x) \mid x \in N\} \setminus \{\overline{a}\}$ is an argument.*

Proof: Let z be the unique child of the root of (N, A, f) where the root z_0 of (N, A, f) is such that $f(z_0) = \overline{a}$. Then $\Gamma = \{f(x) \mid x \in N\} \setminus \{\overline{a}\} = \{f(x) \mid x \in \mathsf{Subtree}(z)\}$.
(1) $\Gamma \vdash \alpha$. Follows from corollary 2. Since (N, A, f) is a support tree, it holds that it is a complete presupport tree. For z it holds that $\mathsf{Disjuncts}(\mathsf{SubtreeRes}(z)) \subseteq \mathsf{Disjuncts}(\alpha)$. This is because according to the definition for a complete presupport tree, for all $x \in N$ the disjuncts of $\mathsf{Subtree}(x)$ are those b that appear in the clauses of the subtree for which either condition i) or condition ii) of the definition holds. Since z is the child of the root z_0 where $f(z_0) = \overline{a}$, then the only disjunct of $f(z)$ for which condition ii) holds is a, which by definition 21 is also a disjunct of α and so for $a \in \mathsf{Disjuncts}(\mathsf{SubtreeRes}(x))$ it holds that $a \in \mathsf{Disjuncts}(\alpha)$. For the rest of the disjuncts b of $\mathsf{SubtreeRes}(z)$, condition i) of definition 21 holds and so $b \in \mathsf{Disjuncts}(\alpha)$, hence all the disjuncts of $\mathsf{SubtreeRes}(z)$ are disjuncts of α so $\mathsf{Disjuncts}(\mathsf{SubtreeRes}(z)) \subseteq \mathsf{Disjuncts}(\alpha)$ and if $\Gamma = \{f(x) \mid x \in \mathsf{Subtree}(z)\}$ then by proposition 10, $\Gamma \vdash \mathsf{SubtreeRes}(z)$ and so $\Gamma \vdash \alpha$.

(2) $\Gamma \nvdash \bot$. Follows from proposition 11. It holds that $a \in \mathsf{Disjuncts}(\mathsf{SubtreeRes}(z))$, so $\mathsf{Disjuncts}(\mathsf{SubtreeRes}(z)) \neq \emptyset$. If $\Gamma \vdash \bot$ then for $\gamma' = \bot, \mathsf{Disjuncts}(\gamma') = \emptyset \subset \mathsf{Disjuncts}(\mathsf{SubtreeRes}(z))$ and so by proposition 11 it cannot hold that $\Gamma \vdash \gamma'$.

(3) There is no $\Gamma' \subset \Gamma$ such that $\Gamma' \vdash \alpha$: For a proof by contradiction assume there is some $\Gamma' \subset \Gamma$ such that $\Gamma' \vdash \alpha$. Without loss of generality assume Γ' is minimal for entailing α. Then, by proposition 12 there is support tree (N', A', f') for Δ, α and some $b \in \mathsf{Disjuncts}(\alpha)$ such that $\Gamma' = \{f(x') \mid x' \in N'\} \setminus \{\overline{b}\}$. We show first that if there is a support tree (N', A', f') for Δ, α and some $b \in \mathsf{Disjuncts}(\alpha)$ then then we can construct a support tree (N'', A'', f'') for $\Delta \cup \{\overline{c} \mid c \in \mathsf{Disjuncts}(\gamma)\}$, $\alpha'' = a$ and a such that $\{f''(x'') \mid x'' \in N''\} = \Gamma' \cup \{\overline{c} \mid c \in \mathsf{Disjuncts}(\gamma)\}$. Then we show that using the structure of (N'', A'', f'') we can construct a complete presupport tree for Δ, α and a where the set of clauses assigned to its non-root nodes is equal to Γ'. Then (N, A, f) does not satisfy the definition for a minimal presupport tree and this contradicts the fact that (N, A, f) is a support tree for Δ, α and a.

Let (N', A', f') be a support tree for Δ, α and some $b \in \mathsf{Disjuncts}(\alpha)$ such that $\Gamma' = \{f(x') \mid x' \in N'\} \setminus \{\overline{b}\}$. Then, it holds that $b \neq a$. Otherwise (N', A', f') would be a support tree for Δ, α and a and so (N', A', f') would be a complete presupport tree for Δ, α and a such that $\{f(x') \mid x' \in N'\} \subset \{f(x) \mid x \in N\}$. Then (N, A, f) could not be a minimal presupport tree for Δ, α and a and this contradicts the assumption that (N, A, f) is a support tree for Δ, α

and a. So (N', A', f') is a support tree for Δ, α and some $b \in \mathsf{Disjuncts}(\alpha)$ such that $b \neq a$.

Let z be the unique child of the root in (N, A, f). Then, $\Gamma = \{f(x) \mid x \in \mathsf{Subtree}(z)\}$. Let $\gamma = \mathsf{SubtreeRes}(z)$. Then, as explained in (1), it holds that $\mathsf{Disjuncts}(\gamma) = \mathsf{Disjuncts}(\alpha) \cap \mathsf{Literals}(\Gamma)$ and also $a \in \mathsf{Disjuncts}(\gamma)$. By corollary 2, $\Gamma \vdash \gamma$. By proposition 11 for z, it holds that there is no clause γ'' such $\mathsf{Disjuncts}(\gamma'') \subset \mathsf{Disjuncts}(\gamma)$ and $\Gamma \vdash \gamma''$. Since $\Gamma \nvdash \gamma''$ for any such γ'', then the following holds:

(i) for all $\Gamma'' \subseteq \Gamma$, and for all γ'' such that $\mathsf{Disjuncts}(\gamma'') \subset \mathsf{Disjuncts}(\gamma)$, $\Gamma'' \nvdash \gamma''$.

Let z' be the unique child of the root of (N', A', f'). Then, $\Gamma' = \{f'(x') \mid x' \in \mathsf{Subtree}(z')\}$. Let $\gamma' = \mathsf{SubtreeRes}(z')$. Then, $\mathsf{Disjuncts}(\gamma') = \mathsf{Disjuncts}(\alpha) \cap \mathsf{Literals}(\Gamma')$. Then, $\Gamma' \vdash \gamma'$ and by (i) follows that $\mathsf{Disjuncts}(\gamma') \not\subset \mathsf{Disjuncts}(\gamma)$. Moreover, it cannot hold that $\mathsf{Disjuncts}(\gamma) \not\subset \mathsf{Disjuncts}(\gamma')$ because then it should hold that $\mathsf{Disjuncts}(\alpha) \cap \mathsf{Literals}(\Gamma) \subset \mathsf{Disjuncts}(\alpha) \cap \mathsf{Literals}(\Gamma')$ which cannot hold because $\mathsf{Literals}(\Gamma') \subseteq \mathsf{Literals}(\Gamma)$ (since $\Gamma' \subset \Gamma$). Then it holds that $\mathsf{Disjuncts}(\gamma') = \mathsf{Disjuncts}(\gamma)$ and so $a \in \mathsf{Disjuncts}(\gamma')$. From the way γ' is defined it holds that $\Gamma' \cup \{\overline{c} \mid c \in \mathsf{Disjuncts}(\gamma')\}$ is a minimal inconsistent set and $\overline{a} \in \{\overline{c} \mid c \in \mathsf{Disjuncts}(\gamma')\}$. Then if $\Gamma'' = (\Gamma' \cup \{\overline{c} \mid c \in \mathsf{Disjuncts}(\gamma)\}) \setminus \{\overline{a}\}$, it holds that $\langle \Gamma', a \rangle$ is an argument. Then, by proposition 12, there is a support tree (N'', A'', f'') for $\Delta \cup \{\overline{c} \mid c \in \mathsf{Disjuncts}(\gamma)\}$, $\alpha'' = a$ and some $a'' \in \mathsf{Disjuncts}(\alpha'')$ such that $\Gamma'' = \{f''(x'') \mid x'' \in N''\} \setminus \{\overline{a''}\}$. Since the unique disjunct of α'' is a, then it follows that $a'' = a$ and so (N'', A'', f'') is a support tree for $\Delta \cup \{\overline{c} \mid c \in \mathsf{Disjuncts}(\gamma)\}$, $\alpha'' = a$ and a. Because α'' is a unit clause, conditions *ii*) and *iii*) of the definition for a complete presupport tree are sufficient for (N'', A'', f'') to satisfy the definition for a complete presupport tree. For the nodes $x'' \in N''$ that are such that $f''(x'') \in \{\overline{c} \mid c \in \mathsf{Disjuncts}(\gamma)\}$ it holds that, apart from $\overline{a}$ that represents the root, the rest have to be leaf nodes. Because for each $x'' \in N''$ such that $f''(x'') \in \{\overline{c} \mid c \in \mathsf{Disjuncts}(\gamma)\} \setminus \{\overline{a}\}$, it holds that $f''(x'') = \overline{c}$ for some $\overline{c}$, $f''(x'')$ consists of a unique disjunct. Then, for this $\overline{c} \in \mathsf{Disjuncts}(f''(x''))$, $\mathsf{Attacks}(f''(x''), f(y'')) = \overline{c}$ where y'' is the parent of x'' and condition *ii*) of definition 21 is satisfied for $\overline{c} \in \mathsf{Disjuncts}(f''(x''))$ and x'' cannot have a child, so it is a leaf node. For y'' and $c \in \mathsf{Disjuncts}(f''(y''))$ condition *iii*) of definition 21 is satisfied. Then, we show that if the leaf nodes that have a literal from $\{\overline{c} \mid c \in \mathsf{Disjuncts}(\gamma)\} \setminus \{\overline{a}\}$ assigned as their clause representation are removed from (N'', A'', f''), the resulting tree is a complete presupport tree for Δ, α and a where the set of non-root nodes is equal $\Gamma'' \setminus \{\overline{c} \mid c \in \mathsf{Disjuncts}(\gamma)\}$ which is equal to Γ'. Since $\Gamma' \subset \Gamma$, and $\Gamma = \{f(x) \mid x \in N\} \setminus \{\overline{a}\}$ this contradicts the assumption that (N, A, f) is a minimal presupport tree. By removing these leaf nodes of (N'', A'', f'') the resulting tree is complete for Δ, α and a because for the parents y'' of each removed leaf x'' with $f''(x'') = \overline{c}$ condition *i*) of the definition for a complete presupport tree holds for $c \in \mathsf{Disjuncts}(f''(y''))$ because for all $\{\overline{c} \mid c \in \mathsf{Disjuncts}(\gamma)\}$ it holds that $c \in \mathsf{Disjuncts}(\alpha)$. Apart from the the parents y'' of each such removed leaf x'', no other node in the tree could be affected by removing these leaf nodes of (N'', A'', f'') and the resulting tree is a

complete presupport tree for Δ, α and a. Also, the set of clauses assigned to its nodes exluding the root is equal to Γ' because $\mathsf{Disjuncts}(\gamma) \subseteq \mathsf{Disjuncts}(\alpha)$ and so by proposition 5, $\Gamma' \cap \{\overline{c} \mid c \in \mathsf{Disjuncts}(\gamma)\} = \emptyset$ so $\Gamma' \setminus \{\overline{c} \mid c \in \mathsf{Disjuncts}(\gamma)\} = \Gamma'$. Since $\Gamma' \subset \Gamma$ it holds that (N, A, f) is not a minimal presupport tree for Δ, α and a and this contradicts the assumption that (N, A, f) is a support tree for Δ, α and a. So, there cannot be a $\Gamma' \subset \Gamma$ such that $\Gamma' \vdash \alpha$. □

Proposition 14. *Let (N, A, f) be a support tree for Δ, α and a. Then, there is no support tree (N', A', f') for Δ, α and some $b \in \mathsf{Disjuncts}(\alpha)$ such that $\{f'(x') \mid x' \in N'\} \setminus \{\overline{b}\} \subset \{f(x) \mid x \in N\} \setminus \{\overline{a}\}$.*

Proof: For a proof by contradiction assume (N, A, f) is a support tree for Δ, α and a and there is a support tree (N', A', f') for Δ, α and some $b \in \mathsf{Disjuncts}(\alpha)$ such that $\{f'(x') \mid x' \in N'\} \setminus \{\overline{b}\} \subset \{f(x) \mid x \in N\} \setminus \{\overline{a}\}$. By proposition 13 for (N, A, f) it holds that $\langle \{f(x) \mid x \in N\} \setminus \{\overline{a}\}, \alpha \rangle$ is an argument and by proposition 13 for (N', A', f') it also holds that $\langle \{f'(x') \mid x' \in N'\} \setminus \{\overline{b}\}, \alpha \rangle$ is an argument so $\{f'(x') \mid x' \in N'\} \setminus \{\overline{b}\} \vdash \alpha$ and by assumption $\{f'(x') \mid x' \in N'\} \setminus \{\overline{b}\} \subset \{f(x) \mid x \in N\} \setminus \{\overline{a}\}$ and this contradicts that $\langle \{f(x) \mid x \in N\} \setminus \{\overline{a}\}, \alpha \rangle$ is argument. □

Corollary 3. *Let Δ be a set of clauses and α be a clause. $\langle \Gamma, \alpha \rangle$ is an argument iff there is an $a \in \mathsf{Disjuncts}(\alpha)$ such that there is a support tree (N, A, f) for Δ, α and a with $\Gamma = \{f(x) \mid x \in N\} \setminus \{\overline{a}\}$.*

Proof: Follows from propositions 12 and 13. □

Lemma 8. *Let (N, A, f) be a consistent presupport tree. Then, for all $x, y \in N$ such that $f(x) = f(y)$, it holds that $\mathsf{Disjuncts}(f(x)) \cap \mathsf{AncestorLabels}(x) = \mathsf{Disjuncts}(f(y)) \cap \mathsf{AncestorLabels}(y)$.*

Proof: Let (N, A, f) be a consistent presupport tree. If for some nodes x, y such that $f(x) = f(y)$ it holds that $\mathsf{AncestorLabels}(x) \cap \mathsf{Disjuncts}(f(x)) \neq \mathsf{AncestorLabels}(y) \cap \mathsf{Disjuncts}(f(y))$ then one of the following holds:
either $\mathsf{AncestorLabels}(x) \cap \mathsf{Disjuncts}(f(x)) \subset \mathsf{AncestorLabels}(y) \cap \mathsf{Disjuncts}(f(y))$ or $\mathsf{AncestorLabels}(y) \cap \mathsf{Disjuncts}(f(y)) \subset \mathsf{AncestorLabels}(x) \cap \mathsf{Disjuncts}(f(x))$.
For a proof by contradiction assume (N, A, f) is a consistent presupport tree for Δ, α and a and $x, y \in N$ are such that $f(x) = f(y)$ and $\mathsf{AncestorLabels}(y) \cap \mathsf{Disjuncts}(f(y)) \subset \mathsf{AncestorLabels}(x) \cap \mathsf{Disjuncts}(f(x))$. Then there is some $b \in \mathsf{AncestorLabels}(x) \cap \mathsf{Disjuncts}(f(x))$ s.t. $b \notin \mathsf{AncestorLabels}(y) \cap \mathsf{Disjuncts}(f(y))$. Because $b \in \mathsf{Disjuncts}(f(x))$ and $f(x) = f(y)$ it holds that $b \in \mathsf{Disjuncts}(f(y))$. For b it holds that $b \notin \mathsf{Disjuncts}(\alpha)$. Since there is an arc (w, w') where $w' \in \mathsf{Ancestors}(x)$ for which it holds that $\mathsf{Attacks}(f(w), f(w')) = b$ then by the definition of a complete presupport tree the only case where b can be in the disjuncts of α is when w' is the root of the tree, w is x and $b = \overline{\overline{a}}$. In this case, $y \in \mathsf{Subtree}(w)$ and $f(y) = f(w)$ which violates the conditions of the definition of a presupport tree. Hence, $b \notin \mathsf{Disjuncts}(\alpha)$ and $b \notin \mathsf{AncestorLabels}(y)$ so conditions i) and ii) of definition 21 do not hold for for $b \in \mathsf{Disjuncts}(f(y))$ and

so in order for (N, A, f) to be a complete presupport tree condition *iii*) of the definition holds. Hence, there is a child y' of y such that $\mathsf{Attacks}(f(y'), f(y)) = \bar{b}$ which contradicts the assumption that (N, A, f) is a consistent presupport tree for Δ, α and a. $\square$.

Proposition 15. *For a set of clauses* Φ, $\neg \bigwedge \Phi \equiv \neg \bigwedge \mathsf{SResolvents}(\Phi)$.

Proof: Clearly, $\Phi \equiv \mathsf{Resolvents}(\Phi)$ and $\mathsf{Resolvents}(\Phi) \equiv \mathsf{SResolvents}(\Phi)$. Therefore, we get $\neg \bigwedge \Phi \equiv \neg \bigwedge \mathsf{SResolvents}(\Phi)$. $\square$

Corollary 4. *Let* Φ, Ψ *be sets of clauses. Then* $\langle \Psi, \neg \bigwedge \Phi \rangle$ *is an argument iff* $\langle \Psi, \neg \bigwedge \mathsf{SResolvents}(\Phi) \rangle$ *is an argument.*

Proof: Follows from proposition 15. $\square$

Proposition 16. *Let* $A = \langle \Phi, \alpha \rangle$ *be an argument and let* $\langle \Psi, \diamond \rangle$ *be a canonical undercut for* A. *Then,* $\forall \psi \in \Psi$, $\forall \phi \in \Phi$, $\mathsf{Disjuncts}(\phi) \not\subseteq \mathsf{Disjuncts}(\psi)$.

Proof: Let $\langle \Psi, \diamond \rangle$ be a canonical undercut for an argument A where $A = \langle \Phi, \alpha \rangle$. Then $\Psi \vdash \neg \bigwedge \Phi$ and $\Psi \cup \{\bigwedge \Phi\}$ is a minimal inconsistent set. To show a contradiction suppose for some $\psi \in \Psi$ and $\phi \in \Phi$, $\mathsf{Disjuncts}(\phi) \subseteq \mathsf{Disjuncts}(\psi)$ (and hence $\phi \vdash \psi$). Then, $(\Psi \setminus \{\psi\}) \cup \{\phi\} \cup \{\bigwedge \Phi\} \vdash \bot$ but because $\phi \in \mathsf{Conjuncts}(\bigwedge \Phi)$, then $(\Psi \setminus \{\psi\}) \cup \{\bigwedge \Phi\} \vdash \bot$ which contradicts the fact that $\Psi \cup \{\bigwedge \Phi\}$ is a minimal inconsistent set. $\square$

Corollary 5. *Let* $\langle \Phi, \alpha \rangle$ *be an argument and let* $\langle \Psi, \diamond \rangle$ *be a canonical undercut for* $\langle \Phi, \alpha \rangle$. *Then there is no* $\phi \in \Phi$, *such that* $\phi \in \Psi$.

Proof: To give a proof by contradiction, let $\langle \Phi, \alpha \rangle$ be an argument, $\langle \Psi, \diamond \rangle$ be a canonical undercut for $\langle \Phi, \alpha \rangle$ and let for some $\phi \in \Phi$, $\phi \in \Psi$. It holds that $\langle \Psi, \neg \wedge \Phi \rangle$ is an argument and so $\Psi \cup \{\wedge \Phi\}$ is a minimal inconsistent set, $\Psi \cup \{\wedge \Phi\} \vdash \bot$ and there is no subset Ψ' of $\Psi \cup \{\wedge \Phi\}$ such that $\Psi' \vdash \bot$. Then, since $\phi \in \Phi \cap \Psi$, it holds that $(\Psi \setminus \{\phi\}) \cup \{\wedge \Phi\} \vdash \bot$ and also $(\Psi \setminus \{\phi\}) \cup \{\wedge \Phi\} \subset \Psi \cup \{\wedge \Phi\}$, which contradicts the assumption that $\Psi \cup \{\wedge \Phi\}$ is a minimal inconsistent set. $\square$

Corollary 6. *Let* $A = \langle \Phi, \alpha \rangle$ *be a argument and let* $\langle \Psi, \diamond \rangle$ *be a canonical undercut for* A. *Then,* $\forall \psi \in \Psi$, $\forall \rho \in \mathsf{SResolvents}(\Phi)$, $\mathsf{Disjuncts}(\rho) \not\subseteq \mathsf{Disjuncts}(\psi)$.

Proof: Let $\langle \Psi, \diamond \rangle$ be a canonical undercut for $\langle \Phi, \alpha \rangle$. By corollary 4 follows that $\langle \Psi, \neg \bigwedge \mathsf{SResolvents}(\Phi) \rangle$ is an argument and so $\Psi \cup \{\bigwedge \mathsf{SResolvents}(\Phi)\}$ is a minimal inconsistent set. Let for some $\psi \in \Psi$ and $\rho \in \mathsf{SResolvents}(\Phi)$, $\mathsf{Disjuncts}(\rho) \subseteq \mathsf{Disjuncts}(\psi)$ (and hence $\rho \vdash \psi$). Then, $\big((\Psi \setminus \{\psi\}) \cup \{\rho\}\big) \cup \{\bigwedge \mathsf{SResolvents}(\Phi)\} \vdash \bot$ and $\rho \in \mathsf{Conjuncts}(\bigwedge \mathsf{SResolvents}(\Phi))$, so $(\Psi \setminus \{\psi\}) \cup \{\bigwedge \mathsf{SResolvents}(\Phi)\} \vdash \bot$ which contradicts the fact that $\Psi \cup \{\bigwedge \mathsf{SResolvents}(\Phi)\}$ is a minimal inconsistent set. $\square$

Corollary 7. *Let $\langle \Phi, \alpha \rangle$ be an argument and $\langle \Psi, \diamond \rangle$ be a canonical undercut for $\langle \Phi, \alpha \rangle$. Then, if $\Delta' = \{\delta \in \Delta \mid \nexists \rho \in \mathsf{SResolvents}(\Phi) \text{ s.t. } \rho \vdash \delta\}$ it holds that $\Psi \subseteq \Delta \setminus \Delta'$.*

Proof: Follows from corollary 6. □

Proposition 17 *Let $\langle \Phi, \alpha \rangle$ be an argument. $\langle \Psi, \diamond \rangle$ is a canonical undercut for $\langle \Phi, \alpha \rangle$ iff there is a $\rho_i \in \mathsf{SResolvents}(\Phi)$ and a $\Gamma_i \subset \mathsf{SResolvents}(\Phi)$ (possibly the empty set) such that $\langle \Psi \cup \Gamma_i, \neg\rho_i \rangle$ is an argument and there is no $\Psi' \subset \Psi$ and $\Gamma_j \subset \mathsf{SResolvents}(\Phi)$ and $\rho_j \in \mathsf{SResolvents}(\Phi)$ (where ρ_j can be equal to ρ_i) such that such that $\langle \Psi' \cup \Gamma_j, \neg\rho_j \rangle$ is an argument.*

Proof: ($\rightarrow$) Let $A = \langle \Phi, \alpha \rangle$ be an argument and $\langle \Psi, \diamond \rangle$ be a canonical undercut for A. Then $\langle \Psi, \neg \wedge \Phi \rangle$ is an argument and $\langle \Psi, \neg \bigwedge \mathsf{SResolvents}(\Phi) \rangle$ is also an argument. Then, $\Psi \cup \{\bigwedge \mathsf{SResolvents}(\Phi)\}$ is a minimal inconsistent set and so there is an $\Gamma_i' \subseteq \mathsf{SResolvents}(\Phi)$ such that $\Psi \cup \Gamma_i'$ is a minimal inconsistent set and $\Gamma_i' \neq \emptyset$ because otherwise $\Psi \vdash \bot$ would hold and $\langle \Psi, \diamond \rangle$ would not be a canonical undercut for A. Then, there is a $\rho_i \in \Gamma_i'$, and if $\Gamma_i = \Gamma_i' \setminus \{\rho_i\}$ then $\langle \Psi \cup \Gamma_i, \neg\rho_i \rangle$ is an argument, so there exists a $\rho_i \in \mathsf{SResolvents}(\Phi)$ and a $\Gamma_i \subset \mathsf{SResolvents}(\Phi)$ (possibly the empty set) such that $\langle \Psi \cup \Gamma_i, \neg\rho_i \rangle$ is an argument. Moreover, there is no $\Psi' \subset \Psi$, $\Gamma_j \subset \mathsf{SResolvents}(\Phi)$ and $\rho_j \in \mathsf{SResolvents}(\Phi)$ such that $\langle \Psi' \cup \Gamma_j, \neg\rho_j \rangle$ is an argument because then $\Psi' \cup \Gamma_j \cup \{\rho_j\}$ would be a minimal inconsistent set and $\Psi' \vdash \neg \bigwedge(\Gamma_j \cup \{\rho_j\})$ from which follows that $\Psi' \vdash \neg \bigwedge(\mathsf{SResolvents}(\Phi))$ and so $\Psi' \vdash \neg \wedge \Phi$ and this contradicts the assumption that $\langle \Psi, \diamond \rangle$ is a canonical undercut for $\langle \Phi, \alpha \rangle$.
($\leftarrow$) Assume that for some $\Psi \subseteq \Delta$, $\rho_i \in \mathsf{SResolvents}(\Phi)$, $\Gamma_i \subset \mathsf{SResolvents}(\Phi)$ (possibly the empty set) $\langle \Psi \cup \Gamma_i, \neg\rho_i \rangle$ is an argument. Then Ψ is consistent and $\Psi \vdash \neg \bigwedge(\Gamma_i \cup \{\rho_i\})$ from which follows that $\Psi \vdash \neg \bigwedge \mathsf{SResolvents}(\Phi)$. Also assume and there are no $\Psi' \subset \Psi$, $\Gamma_j \subset \mathsf{SResolvents}(\Phi)$ and $\rho_j \in \mathsf{SResolvents}(\Phi)$ (where ρ_j can be equal to ρ_i) such that $\langle \Psi' \cup \Gamma_j, \neg\rho_j \rangle$ is an argument. Then, $\Psi \cup \Gamma_i \cup \{\rho_i\}$ is a minimal inconsistent set and there is no $\Psi' \subset \Psi$ and $\Gamma_j' \subseteq \mathsf{SResolvents}(\Phi)$ such that $\Psi' \cup \Gamma_j'$ is a minimal inconsistent set, so there is no $\Psi' \subset \Psi$ such that $\Psi' \cup \{\bigwedge \mathsf{SResolvents}(\Phi)\} \vdash \bot$, and so there is no $\Psi' \subset \Psi$ such that $\Psi' \vdash \neg \bigwedge \mathsf{SResolvents}(\Phi)$, and so $\langle \Psi, \neg \bigwedge \mathsf{SResolvents}(\Phi) \rangle$ is an argument and hence $\langle \Psi, \diamond \rangle$ is a canonical undercut for $\langle \Phi, \alpha \rangle$. □

Bibliography

[1] L. Amgoud and C. Cayrol. A model of reasoning based on the production of acceptable arguments. *Annals of Math. and A.I.*, 34:197–216, 2002.

[2] L. Amgoud, C. Cayrol, D. Le Berre, and P. Sabatier. Comparing arguments using preference orderings for argument-based reasoning. *IEEE International Conference on Tools with Artificial Intelligence*, 0:400, 1996.

[3] L. Amgoud and S. Kaci. An argumentation framework for merging conflicting knowledge bases. *International Journal of Approximate Reasoning*, 45(2):321 – 340, 2007.

[4] S. Benferhat, D. Dubois, and H. Prade. Argumentative inference in uncertain and inconsistent knowledge bases. In *Proceedings of the 9th Annual Conference on Uncertainty in Artificial Intelligence (UAI 1993)*, pages 1449–1445, 1993.

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

[6] Ph. Besnard and A. Hunter. *Elements of Argumentation*. MIT Press, 2008.

[7] Ph. Besnard, E. Grégoire, C. Piette, and B. Raddaoui. Mus-based generation of arguments and counter-arguments. In *11th IEEE International Conference on Information Reuse and Integration(IRI'10)*, pages 239–244, Las Vegas, NV, USA, 2010. IEEE.

[8] Ph. Besnard and A. Hunter. Argumentation based on classical logic. *Argumentation in Artificial Intelligence*, pages 133–152, 2009.

[9] Ph. Besnard, A. Hunter, and S. Woltran. Encoding deductive argumentation in quantified boolean formulae. *Artif. Intell.*, 173(15):1406–1423, 2009.

[10] D. Bryant, P. Krause, and G. Vreeswijk. Argue tuProlog: A lightweight argumentation engine for agent applications. In *Computational Models of Argument (Comma'06)*, pages 27–32. IOS Press, 2006.

[11] Th. Castell, C. Cayrol, M. Cayrol, and D. Le Berre. Using the davis and putnam procedure for an efficient computation of preferred models. In *12th European Conference on Artificial Intelligence (ECAI'96)*, pages 350–354. Wiley & Sons, Ltd, 1996.

[12] 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*, pages 668–679. Springer, 2001.

[13] C. Chesñevar, A. Maguitman, and R. Loui. Logical models of argument. *ACM Computing Surveys*, 32:337–383, 2000.

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

[15] P. Dung, R. Kowalski, and F. Toni. Dialectical proof procedures for assumption-based admissible argumentation. *Artificial Intelligence*, 170:114–159, 2006.

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

[17] V. Efstathiou and A. Hunter. Focused search for arguments from propositional knowlegde. In *Proceedings of the Second International Conference on Computational Models of Argument (COMMA'08)*. IOS Press, 2008.

[18] V. Efstathiou and A. Hunter. An algorithm for generating arguments in classical predicate logic. In *Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU'09)*, pages 119–130. Springer, 2009.

[19] V. Efstathiou and A.Hunter. JArgue: An implemented argumentation system for classical propositional logic (software demo). `http://www.ing.unibs.it/comma2010/demos/Efstathiou_etal.pdf`, 2010.

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

[21] M. Elvang-Gøransson, P. Krause, and J. Fox. Dialectic reasoning with classically inconsistent information. In *Proceedings of the 9th Conference on Uncertainty in Artificial Intelligence (UAI 1993)*, pages 114–121. Morgan Kaufmann, 1993.

[22] J. Fox, D. Glasspool, D. Grecu, S. Modgil, M. South, and V. Patkar. Argumentation-based inference and decision making–a medical perspective. *IEEE Intelligent Systems*, 22(6):34–41, 2007.

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

[24] N. Gorogiannis, A. Hunter, and M. Williams. An argument-based approach to reasoning with clinical knowledge. *International Journal of Approximate Reasoning*, 51(1):1 – 22, 2009.

[25] R. Hirsch and N. Gorogiannis. The Complexity of the Warranted Formula Problem in Propositional Argumentation. *J Logic Computation*, page exp074, 2009.

[26] A. Hunter. Argumentation Factory. Algorithms and Software for Industrial Strength Inconsistency Tolerance. `http://www.cs.ucl.ac.uk/staff/a.hunter/projects/af/index.html`.

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

[28] R. Kowalski. A proof procedure using connection graphs. *Journal of the ACM*, 22:572–595, 1975.

[29] R. Kowalski. *Logic for problem solving*. North-Holland Publishing, 1979.

[30] D. W. Loveland. A linear format for resolution. In *Symposium on Automatic Demonstration*, pages 147–162. Springer, 1970.

[31] S. Parsons, M. Wooldridge, and L. Amgoud. Properties and complexity of some formal inter-agent dialogues. *Journal of Logic and Computation*, 13(3):347–376, 2003.

[32] H. Prakken. *Logical Tools for Modelling Legal Argument*. Law and Philosophy Library. Springer, 1997.

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

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

[35] J. A. Robinson. A machine-oriented logic based on the resolution principle. *J. ACM*, 12(1):23–41, 1965.

[36] G. Vreeswijk. An algorithm to compute minimally grounded and admissible defence sets in argument systems. In *Computational Models of Argument (Comma'06)*, pages 109–120. IOS Press, 2006.