A Tableau Algorithm for the Clique Guarded Fragment

Preliminary Version

Colin Hirsch
Mathematische Grundlagen der Informatik, RWTH Aachen
hirsch@informatik.rwth-aachen.de

Stephan Tobies
LuFG Theoretical Computer Science, RWTH Aachen
tobies@informatik.rwth-aachen.de

LTCS-Report 00-03

A Tableau Algorithm for the Clique Guarded Fragment

Preliminary Version

Colin Hirsch
Mathematische Grundlagen der Informatik, RWTH Aachen
hirsch@informatik.rwth-aachen.de

Stephan Tobies
LuFG Theoretical Computer Science, RWTH Aachen
tobies@informatik.rwth-aachen.de

May 19, 2000

1 Introduction

The Guarded Fragment of first-order logic, introduced by Andréka, van Benthem, and Németi [1], has been a successful attempt to transfer many good properties of modal, temporal, and description logics to a larger fragment of predicate logic. Among these are decidability, the finite model property, invariance under an appropriate variant of bisimulation, and other nice model theoretic properties [1, 4].

The Guarded Fragment (GF) is obtained from full first-order logic through relativisation of quantifiers by so-called guard formulas. Every appearance of a quantifier in the GF must be of the form

$$\exists \mathbf{y}(\alpha(\mathbf{x},\mathbf{y}) \land \varphi(\mathbf{x},\mathbf{y})) \text{ or } \forall \mathbf{y}(\alpha(\mathbf{x},\mathbf{y}) \to \varphi(\mathbf{x},\mathbf{y})),$$

where α is a positive atomic formula, the *guard*, that contains all free variables of ψ. This generalises quantification in modal and temporal logics, where quantification is restricted to those elements reachable via some accessibility relation.

By allowing for more general formulas as guards while preserving the idea of quantification only over elements that are *close together* in the model, one obtains generalisations of GF which are still well-behaved in the above sense. Most importantly, one can obtain the *loosely guarded fragment* (LGF) [13] and the *clique guarded fragment* (CGF) [5], for which decidability, invariance under clique guarded bisimulation, and some other properties have been shown in [5]. The question whether CGF and LGF have the finite model property has been open until now.

GF, LGF, and CGF are decidable and known to be 2-ExpTime complete, which is shown in [4, 5] using game and automata-based approaches. While these approaches yield optimal worst-case complexity results for many logics, they appear to be unsuitable as a starting point for an efficient implementation—their worst-case complexity is actually their any-case complexity. Many decidability results for modal or description logics are based on tableau algorithms [11, 7, 2, 10]. Some of the fastest implementations of modal satisfiability procedures are based on tableaux calculi [9]. Unlike automata algorithms, the average-case behaviour in practice is so good that finding *really* hard problems to test these implementations has become a problem itself.

In this paper, we generalise the principles usually found in tableau algorithms for modal logics to develop a tableau algorithm for CGF. To the best of our knowledge, this is the first algorithm for CGF that can be used as the basis for an efficient implementation[1]. As a corollary of the constructions used to show the soundness of our algorithm, we obtain that CGF, and hence LGF and GF have the finite model property. Also, we obtain an alternative proof for the fact that every satisfiable CGF formula of width k has a model of tree width at most $k-1$ [5].

In the current version, there is still a gap in the proof Lemma 3.14, which is necessary to establish the finite model property. This does not cause a gap in the proof of Theorem 3.5 because it can alternatively be established using Lemma 3.17.

2 Preliminaries

For the definitions of GF and LGF we refer the reader to [5]. The *clique guarded fragment* CGF of first-order logic can be obtained in two equivalent ways, by either semantically or syntactically restricting the range of the first-order quantifiers. In the following we will use bold letters to refer to tuples of elements of the universe $(\mathbf{a}, \mathbf{b}, \dots)$ resp. tuples of variables $(\mathbf{x}, \mathbf{y}, \dots)$.

Definition 2.1 (Semantic CGF). *Let τ be a relational vocabulary. For a τ-structure $\mathfrak{A}$ with universe A, the* Gaifman graph *of $\mathfrak{A}$ is defined as the undirected graph $G(\mathfrak{A}) = (A, E^{\mathfrak{A}})$ with*

$$E^{\mathfrak{A}} = \{(a, a') \mid a \neq a', \text{there exists } R \in \tau \text{ and} \mathbf{a} \in R^{\mathfrak{A}} \text{ which contains both } a \text{ and } a'\}.$$

Under clique guarded semantics *we understand the modification of standard first order semantics, where, instead of ranging over all elements of the universe, a quantifier is restricted to elements that form a clique in the Gaifman graph, including the binding for the free variables of the matrix formula. More precisely, let $\mathfrak{A}$ be a σ-structure and β an environment mapping variables to elements of A. We define the model relation inductively over the structure of formulas as the usual* FO *semantics with the exception*

$$\mathfrak{A}, \beta \models \forall y.\varphi(\mathbf{x}, y) \text{ iff for all } a \in A \text{ such that } \beta(\mathbf{x}) \cup \{a\} \text{ forms a clique in } G(\mathfrak{A}), \mathfrak{A}, \beta[x \mapsto a] \models \varphi,$$

and a similar definition for the existential case. With CGF *we denote first order logic restricted to clique guarded semantics.*

Definition 2.2 (Syntactic CGF). *Let τ be a relational vocabulary. A formula α is said to be a* clique-formula *for a set $\mathbf{x} \subseteq \text{free}(\alpha)$ if α is a conjunction of atoms such that each two elements from $\mathbf{x}$ coexist in at least one atom, each atom contains at least two element from $\mathbf{x}$, and each element from $\text{free}(\alpha) \setminus \mathbf{x}$ occurs exactly once in one atom. In the following, we will identify a clique-formula α with the set of its conjuncts.*

The syntactic CGF *is inductively defined as follows.*

1. *Every relational atomic formula $Rx_{i_1} \dots x_{i_m}$ or $x_i = x_j$ belongs to* CGF.
2. CGF *is closed under boolean operations.*

[1]There are resolution based decision procedures for GF and LGF [3] that are readily implemented using the saturation theorem prover SPASS [14]. It is unclear if this approach can be extended to CGF.

3. *If* $\mathbf{x}, \mathbf{y}, \mathbf{z}$ *are tuples of variables,* $\alpha(\mathbf{x}, \mathbf{y}, \mathbf{z})$ *is a* clique-formula *for* $\mathbf{x} \cup \mathbf{y}$ *and* $\varphi(\mathbf{x}, \mathbf{y})$ *is a formula in* CGF *such that* $\text{free}(\varphi) \subseteq \mathbf{x} \cup \mathbf{y}$*, then*

$$\exists \mathbf{yz}.(\alpha(\mathbf{x}, \mathbf{y}, \mathbf{z}) \wedge \varphi(\mathbf{x}, \mathbf{y})) \quad \textit{and} \quad \forall \mathbf{yz}.(\alpha(\mathbf{x}, \mathbf{y}, \mathbf{z}) \rightarrow \varphi(\mathbf{x}, \mathbf{y}))$$

belong to CGF.

We will use $(\exists \mathbf{yz}.(\alpha(\mathbf{x}, \mathbf{y}, \mathbf{z}))\varphi(\mathbf{x}, \mathbf{y})$ *and* $(\forall \mathbf{yz}.(\alpha(\mathbf{x}, \mathbf{y}, \mathbf{z}))\varphi(\mathbf{x}, \mathbf{y})$ *as alternative notations for* $\exists \mathbf{yz}.(\alpha(\mathbf{x}, \mathbf{y}, \mathbf{z}) \wedge \varphi(\mathbf{x}, \mathbf{y}))$ *and* $\forall \mathbf{yz}.(\alpha(\mathbf{x}, \mathbf{y}, \mathbf{z}) \rightarrow \varphi(\mathbf{x}, \mathbf{y}))$ *respectively.*

The following lemma can be shown by elementary manipulation.

Lemma 2.3. *Let* $\alpha(\mathbf{x}, \mathbf{y}, \mathbf{z})$ *be a clique-formula for* $\mathbf{x}, \mathbf{y}$*. Then*

$$\forall \mathbf{yz}.(\alpha(\mathbf{x}, \mathbf{y}, \mathbf{z}) \rightarrow \varphi(\mathbf{x}, \mathbf{y})) \equiv \forall \mathbf{y}.(\exists \mathbf{z}.\alpha(\mathbf{x}, \mathbf{y}, \mathbf{z}) \rightarrow \varphi(\mathbf{x}, \mathbf{y})).$$

The use of the name CGF both for the semantic and the syntactic clique guarded fragment is justified by the following Lemma.

Lemma 2.4. *Over any finite relational vocabulary the syntactic and semantic versions of the* CGF *are equally expressive.*

Proof sketch: By some elementary equivalence transformations, every syntactically clique guarded formula can be brought into a form where switching from standard semantics to clique guarded semantics does not change its meaning. Conversely, for any finite signature there is a finite disjunction $\mathit{clique}(\mathbf{x}, y, \mathbf{z})$ of clique-formulas for $\mathbf{x}, y$ such that $\mathbf{a}, b$ form a clique in $G(\mathfrak{A})$ iff $\mathfrak{A} \models \exists \mathbf{z}.\mathit{clique}(\mathbf{a}, b, \mathbf{z})$. By guarding every quantifier with such a formula and applying some elementary formula transformations and Lemma 2.3, we get, for every FO formula ψ, a syntactically clique guarded formula that is equivalent to ψ under clique guarded semantics.

In the following we will only consider the syntactic variant of the clique guarded fragment.

At a first glance the expressiveness of CGF and the loosely guarded fragment LGF are incomparable. While the auxiliary variables of the CGF allow additional expressiveness, there are also LGF-formulas that are not (syntactically) clique guarded. In CGF, a guard α in $Q\mathbf{yz}.\alpha(\mathbf{x}, \mathbf{y}, \mathbf{z})$ necessarily contains every pair from $\mathbf{x} \cup \mathbf{y}$ in (at least) one atom. In LGF a guard β in $Q\mathbf{y}.\beta(\mathbf{x}, \mathbf{y})$ need only contain all combinations of a variable from $\mathbf{x}$ with one from $\mathbf{x} \cup \mathbf{y}$ in (at least) one guard atom. An example for a loosely guarded formula that is not (syntactically) clique guarded is

$$\psi = (\exists xy.Rxy)(\forall z.(Rxz \wedge Ryz))\varphi(x, y, z)$$

because x and y do not coexist in the guard of the universal quantifier. Yet, ψ can be turned into a clique guarded formula by adding the guard Rxy of the existential quantifier to the guard of the universal quantifier. This yields the guard $Rxy \wedge Rxz \wedge Ryz$, a clique formula for x, y, z. Since it is always possible to clique-guard a loosely guarded formula in this way, LGF is contained in CGF. It is also possible to show that CGF is strictly more expressive than LGF [5].

Definition 2.5 (NNF, Closure, Width). *Let* $\psi \in$ CGF *be closed. In the following, we assume all formulas to be in negation normal form (NNF), where negation occurs only in front of atomic formulas. Every formula in* CGF *can be transformed into NNF in linear time by pushing negation inwards using DeMorgan's law and the duality of the quantifiers.*

For a formula $\psi \in$ CGF in NNF, let $\mathrm{cl}(\psi)$ be the smallest set that contains ψ and is closed under sub-formulas. Let C be a set of constants. With $\mathrm{cl}(\psi, C)$ we denote the set

$$\mathrm{cl}(\psi, C) = \{\varphi(\mathbf{a}) \mid \mathbf{a} \subseteq C, \varphi(\mathbf{x}) \in \mathrm{cl}(\psi)\}.$$

The width *of a formula $\psi \in$ CGF is defined by*

$$\mathrm{width}(\psi) := \max\{|\mathrm{free}(\varphi)| \mid \varphi \in \mathrm{cl}(\psi)\}.$$

3 A Tableaux Algorithm for CGF

For various modal and description logics, decidability can be shown by means of tableaux algorithms, where satisfiability of a formula ψ is decided by a syntactically guided search for a model for ψ. Examples for these kind of algorithms can be found, e.g., in [11, 12, 7, 10]. Models are usually represented by a graph in which the nodes correspond to worlds and the edges correspond to the accessibility relations in the model. Each node is labeled with a set formulas that this node must satisfy, and new edges and nodes are created as required by existential modalities. Since many modal and description logics have the tree model property, the graphs generated by these algorithms are trees, which allows for simpler algorithms and easier implementation and optimisation of these algorithms. Indeed, some of the fastest implementations of modal and description logics satisfiability algorithms are based on tableau calculi [9].

For many modal or description logics, e.g. K or $\mathcal{ALC}$, termination of these algorithms is due to the fact that the modal depth of the formulas appearing at a node strictly decreases with every step from the root of the tree. For other logics, e.g., $\mathsf{K4}$, K with the universal modality, or the expressive DL $\mathcal{SHIQ}$, this is no longer true and termination has to be enforced by other means. One possibility for this is *blocking*, i.e., stopping the creation of new successor nodes below a node v if there already is an ancestor node w that is labeled with similar formulas as v. Intuitively, in this case the model can fold back from the predecessor of v to w, creating a cycle. Unraveling of these cycles recovers an (infinite) tree model. Since the algorithms guarantee that the formulas occurring in the label of the nodes stem from a finite set (usually the sub-formulas of the input formula), every growing path will eventually contain a blocked node, preventing further growth of this path and (together with a bound on the degree of the tree) ensuring termination of the algorithm.

Our investigation of a tableaux algorithm for CGF starts with the observation that CGF also has some kind of tree model property.

Definition 3.1. *Let τ be a relational vocabulary. A τ-structure $\mathfrak{A}$ has* tree width k *if $k \in \mathbb{N}$ is minimal with the following property.*

There exists a directed tree $T = (V, E)$ and a function $f : V \to 2^A$ such that

- *for every $v \in V$, $|f(v)| \leq k + 1$,*
- *for every $R \in \tau$ and $\mathbf{a} \in R^{\mathfrak{A}}$, there exists $v \in V$ with $\mathbf{a} \subseteq f(v)$, and*
- *for every $a \in A$, the set $V_a = \{v \in V \mid a \in f(v)\}$ induces a subtree of T.*

Every node v of T induces a substructure $\mathfrak{F}(v) \subseteq \mathfrak{A}$ of cardinality at most $k+1$. Since $f(v)$ may be empty we, admit empty substructures. The tuple $\langle T, (\mathfrak{F}(v))_{v \in T}\rangle$ is called a tree decomposition *of $\mathfrak{A}$.*

Fact 3.2 (Tree Model Property). *Every satisfiable sentence $\psi \in$ CGF of width k has a countable model of tree width at most $k-1$.*

This is a simple corollary of [5], Theorem 4, where the same result is given for an extension of CGF by least fixed point operators.

Fact 3.2 is the starting point for our definition of a *completion tree* for a formula $\psi \in$ CGF. A node v of such a tree no longer stands for a single element of the model as in the modal case, but rather for a substructure $\mathfrak{F}(v)$ of a tree decomposition of a model. To this purpose, we label every node v with a set $\mathbf{C}(v)$ of constants (the elements of the substructure) and a subset of $cl(\psi, \mathbf{C}(v))$, reflecting the formulas that must hold true for these elements.

To deal with *auxiliary elements*—elements helping to form a clique in $G(\mathfrak{A})$ that are not part of this clique themselves—we will use $*$ as a placeholder for an unspecified element in atoms. The following definitions are useful when dealing with these generalised atoms.

Definition 3.3. *Let K denote an infinite set of constants and $* \notin K$. For any set of constants $C \subseteq K$ we set $C^* = C \cup \{*\}$. We use $t_1, t_2, \ldots$ to range over elements of K^*. The relation $\geq^*$ is defined by*

$$Rt_1 \ldots t_n \geq^* Rt'_1 \ldots t'_n \text{ iff for all } i \in \{1 \ldots n\} \text{either } t_i = * \text{ or } t_i = t'_i.$$

For an atom β and a set of formulas Φ we define $\beta \in^* \Phi$ iff there is a $\beta' \in \Phi$ with $\beta \geq^* \beta'$.

For a set of constants $C \subseteq K$ and an atom $\alpha = Rt_1 \ldots t_n$, we define

$$\alpha|^*_C = Rt'_1 \ldots t'_n \quad \text{where} \quad t'_i = \begin{cases} t_i & \text{if } t_i \in C \\ * & \text{otherwise} \end{cases}$$

We use the notation $\mathbf{a}^*$ to indicate that the tuple $\mathbf{a}^*$ contains $*$'s. Obviously, $\geq^*$ is transitive and reflexive, and $\alpha|^*_C \geq^* \alpha$ for all atoms α and sets of constants C.

While these are all syntactic notions, they have a semantic counterpart that clarifies the intuition of $*$ standing for an unspecified element. Let $\mathbf{a}'$ denote the tuple obtained from a tuple $\mathbf{a}^*$ by replacing every occurrence of $*$ in $\mathbf{a}^*$ with a distinct fresh variable, and let $\mathbf{z}$ be precisely the variables used for this replacement. For an atom α, we define

$$\mathfrak{A} \models \alpha(\mathbf{a}^*) \text{ iff } \mathfrak{A} \models \exists \mathbf{z}.\alpha(\mathbf{a}').$$

It is easy to see that

$$\begin{aligned} \alpha(\mathbf{a}) \geq^* \alpha(\mathbf{b}) \quad &\text{implies} \quad \alpha(\mathbf{b}) \models \alpha(\mathbf{a}) \\ \alpha(\mathbf{a}) \in^* \Phi \quad &\text{implies} \quad \Phi \models \alpha(\mathbf{a}) \end{aligned}$$

because, if $\mathbf{a} \geq^* \mathbf{b}$, then $\mathbf{b}$ is obtained from $\mathbf{a}$ by replacing some $*$ with constants, which provide witnesses for the existential quantifier.

Definition 3.4 (Completion Tree, Tableau). *Let $\psi \in$ CGF be a closed formula in* NNF. *A* completion tree $\mathbf{T} = (\mathbf{V}, \mathbf{E}, \mathbf{C}, \Delta, \mathbf{N})$ *for ψ is a vertex labeled tree $(\mathbf{V}, \mathbf{E})$ with the labeling function $\mathbf{C}$ labeling each node $v \in \mathbf{V}$ with a subset of K, Δ labeling each node $v \in \mathbf{V}$ with a subset of $cl(\psi, \mathbf{C}(v)^*)$ such that $*$ occurs only in atoms (without equality) and the function $\mathbf{N} : \mathbf{V} \to \mathbb{N}$ mapping each node to a distinct natural number, with the additional property that, if v is an ancestor of w, then $\mathbf{N}(v) < \mathbf{N}(w)$.*

A constant $c \in K$ is called shared *between two nodes $v_1, v_2 \in \mathbf{V}$, if $c \in \mathbf{C}(v_1) \cap \mathbf{C}(v_2)$, and $c \in \mathbf{C}(w)$ for all nodes w on the (unique, undirected, possibly empty) path connecting v_1 to v_2.*

A node $v \in \mathbf{V}$ is called directly blocked *by a node $w \in \mathbf{V}$, if w is not blocked, $\mathbf{N}(w) < \mathbf{N}(v)$ and there is an injective mapping π from $\mathbf{C}(v)$ into $\mathbf{C}(w)$ such that, for all constants $c \in \mathbf{C}(v)$ that are shared between v and w, $\pi(c) = c$, and $\pi(\Delta(v)) = \Delta(w)|_{\pi(\mathbf{C}(v)^*)}$. Here and throughout this paper we use the convention $\pi(*) = *$ for every function π that verifies a blocking.*

A node is called blocked *if it is directly blocked or if its predecessor is blocked.*

A completion tree $\mathbf{T}$ contains a clash *if there is a node $v \in \mathbf{V}$ such that*

- *for a constant $c \in \mathbf{C}(v)$, $c \neq c \in \Delta(v)$, or*
- *there is an atomic formula α and a tuple of constants $\mathbf{a} \subseteq \mathbf{C}(v)$ such that $\{\alpha(\mathbf{a}), \neg\alpha(\mathbf{a})\} \subseteq \Delta(v)$.*

Otherwise, $\mathbf{T}$ *is called* clash-free. *A completion tree* $\mathbf{T}$ *is called* complete *if none of the* completion rules *given in Figure 1 can be applied to* $\mathbf{T}$. *A complete and clash-free completion tree for ψ is called a* tableau *for ψ.*

To test ψ for satisfiability, the tableau algorithm creates an initial tree with only a single node v_0, $\Delta(v_0) = \{\psi\}$ and $\mathbf{C}(v_0) = \emptyset$. The rules from Figure 1 are succesively applied until either a clash occurs, producing output "ψ `unsatisfiable`*", or the tree is complete, in which case "ψ* `is satisfiable`*" is output.*

R$\wedge$:	if	$\varphi \wedge \vartheta \in \Delta(v)$ and $\{\varphi, \vartheta\} \not\subseteq \Delta(v)$
	then	$\Delta(v) := \Delta(v) \cup \{\varphi, \vartheta\}$
R$\vee$:	if	$\varphi \vee \vartheta \in \Delta(v)$ and $\{\varphi, \vartheta\} \cap \Delta(v) = \emptyset$
	then	$\Delta(v) := \Delta(v) \cup \{\chi\}$ for $\chi \in \{\varphi, \vartheta\}$
R$=$:	if	$a = b \in \Delta(v)$
	then	for all w that share a with $v, \mathbf{C}(w) := (\mathbf{C}(w) \setminus \{a\}) \cup \{b\}$ and $\Delta(w) := \Delta(w)[a \mapsto b]$
R$\forall$:	if	$(\forall \mathbf{yz}.\alpha(\mathbf{a}, \mathbf{y}, \mathbf{z}))\varphi(\mathbf{a}, \mathbf{y})) \in \Delta(v),$ there exists a $\mathbf{b} \subseteq \mathbf{C}(v)$ such that for all atoms $\beta(\mathbf{x}, \mathbf{y}, \mathbf{z}) \in \alpha, \beta(\mathbf{a}, \mathbf{b}, * \cdots *) \in^* \Delta(v),$ and $\varphi(\mathbf{a}, \mathbf{b}) \notin \Delta(v)$
	then	$\Delta(v) := \Delta(v) \cup \{\varphi(\mathbf{a}, \mathbf{b})\}$
R$\exists$:	if	$(\exists \mathbf{yz}.\alpha(\mathbf{a}, \mathbf{y}, \mathbf{z}))\varphi(\mathbf{a}, \mathbf{y})) \in \Delta(v)$ and for every $\mathbf{b}, \mathbf{c} \subseteq \mathbf{C}(v), \{\alpha(\mathbf{a}, \mathbf{b}, \mathbf{c}), \varphi(\mathbf{a}, \mathbf{b})\} \not\subseteq \Delta(v)$ and there is no child w of v with $\{\alpha(\mathbf{a}, \mathbf{b}, \mathbf{c}), \varphi(\mathbf{a}, \mathbf{b})\} \subseteq \Delta(w)$ for some $\mathbf{b}, \mathbf{c} \subseteq \mathbf{C}(w)$ and v is not blocked
	then	let $\mathbf{b}, \mathbf{c}$ be sequences of distinct and fresh constants that match the lengths of $\mathbf{y}, \mathbf{z}$, create a child w of v with $\mathbf{C}(w) := \mathbf{a} \cup \mathbf{b} \cup \mathbf{c}$ and $\Delta(w) := \{\alpha(\mathbf{a}, \mathbf{b}, \mathbf{c}), \varphi(\mathbf{a}, \mathbf{b})\},$ and let $\mathbf{N}(w) = 1 + \max\{\mathbf{N}(v) \ : \ v \in \mathbf{V} \setminus \{w\}\}$
R$\updownarrow$:	if	$\alpha(\mathbf{a}^*) \in \Delta(v), \alpha$ atomic, w is a neighbour of v with $\mathbf{a}^* \cap \mathbf{C}(w) \neq \emptyset,$ and $\alpha(\mathbf{a}^*)\vert^*_{\mathbf{C}(w)} \notin \Delta(w)$
	then	$\Delta(w) := \Delta(w) \cup \{\alpha(\mathbf{a})\vert^*_{\mathbf{C}(w)}\}$
R$\updownarrow\forall$:	if	$\varphi(\mathbf{a}) \in \Delta(v), \varphi(\mathbf{a})$ universal, and y is a neighbour of x with $\mathbf{a} \subseteq \mathbf{C}(w)$ and $\varphi(\mathbf{a}) \notin \Delta(w)$
	then	$\Delta(w) := \Delta(w) \cup \{\varphi(\mathbf{a})\}$

Figure 1: The Completion Rules for CGF

While our notion of tableaux has many similarities to the tableaux appearing in [6], there are two important differences that make the notion of tableaux here more suitable as basis for a tableau algorithm.

We will see that every completion tree generated by the tableau algorithm is finite. Conversely, tableaux in [6], in general, can be infinite.

Also, in [6] every node is labeled with a complete $(\psi, \mathbf{C}(v))$-type, i.e., every formula $\varphi \in \mathrm{cl}(\psi, \mathbf{C}(v))$ is explicitly asserted true of false at v. Conversely, a completion tree contains only assertions about relevant formulas. This implies a lower degree of non-determinism in the algorithm, which is important for an efficient implementation.

Theorem 3.5. *The tableau algorithm is a (non-deterministic) decision procedure for* CGF*-satisfiability.*

Proof: This is an immediate consequence of the following facts established in the subsequent sections.

1. Every sequence of rule applications terminates after a finite number of steps. (*Termination*, Lemma 3.8)
2. If the algorithm constructs a tableau for ψ, then ψ is satisfiable. (*Soundness*, Lemma 3.15)
3. If ψ is satisfiable, then the rules can be applied to generate a tableau for ψ. (*Completeness*, Lemma 3.16) ∎

As a corollary, we get that CGF and hence also the loosely guarded fragment, and the guarded fragment, have the finite model property. For GF this was already known [4], whereas for LGF and CGF this was still an open problem.

Corollary 3.6. *Let $\psi \in$ GF/LGF/CGF. ψ is satisfiable iff ψ is finitely satisfiable.*

Proof: If ψ is satisfiable, the tableau algorithm generates a finite tableau for ψ. The construction in the proof of Lemma 3.15 shows that such a tree induces a finite model. ∎

3.1 Termination

The folloing technical lemma is a simple consequence of the completion rules and the blocking condition.

Lemma 3.7. *Let $\psi \in$ CGF be a closed formula in NNF with $|\psi| = n$, $\mathrm{width}(\psi) = m$, and $\mathbf{T}$ a completion tree generated for ψ by application of the rules in Figure 1. For every node $v \in \mathbf{T}$,*

1. $|\mathbf{C}(v)| \leq m$
2. $|\Delta(v)| \leq n \times (m+1)^m$
3. *Any $\ell > 2^{n \times (m+1)^m}$ distinct nodes in $\mathbf{T}$ contain a blocked node.*

Proof:

1. Nodes are only generated by the R$\exists$-rule and no constants are added to a $\mathbf{C}(v)$ once v has been generated (but some may be removed by application of the R=-rule).

   When triggered by the formula $(\exists \mathbf{yz}.\alpha(\mathbf{a},\mathbf{y},\mathbf{z}))\varphi(\mathbf{a},\mathbf{y})$, the R$\exists$-rule initializes $\mathbf{C}(w)$ such that it contains $\mathbf{a}$ and another constant for every variable in $\mathbf{x}$ and $\mathbf{y}$. Hence,

$$|\mathbf{C}(w)| \leq |\mathbf{a} \cup \mathbf{y} \cup \mathbf{z}| \leq |\mathrm{free}(\alpha)| \leq \mathrm{width}(\psi).$$

2. The set $\Delta(v)$ is a subset of $cl(\psi, \mathbf{C}(v)^*)$, for which $|cl(\psi,\mathbf{C}(v))| \leq n \times (m+1)^m$ holds because there are at most n formulas in $cl(\psi)$, each of which has at most m free variables. There are at most $(|\mathbf{C}(v)|+1)^m$ distinct sequences of length m with constants from $\mathbf{C}(v)^*$.

3. Let $v_1,\dots,v_\ell$ be $\ell > 2^{n\times(m+1)^m}$ distinct nodes. For every v_i, we will construct a mapping $\pi_i : \mathbf{C}(v_i) \to \{1,\dots m\}$ such that, if a constant a is shared between two nodes v_i, v_j, then $\pi_i(a) = \pi_j(a)$.

   Let $u_1,\cdots,u_k$ denote the nodes of a subtree of $\mathbf{T}$ that contains every node v_i and that is rooted at u_1. By induction over the distance to u_1, we define an injective mapping $\nu_i : \mathbf{C}(u_i) \to \{1,\dots,m\}$ for every $i \in \{1,\dots,k\}$ as follows. For ν_1 we pick an arbitrary injective function from $\mathbf{C}(u_1)$ to $\{1,\dots,m\}$. For a node u_i let u_j be the predecessor of u_i in $\mathbf{T}$ and ν_j the corresponding function, which, since u_j has a smaller distance to u_1, has already been defined. For ν_i we choose an arbitrary injective function such that $\nu_i(a) = \nu_j(a)$ for all $a \in \mathbf{C}(u_i) \cap \mathbf{C}(u_j)$.

   All mappings ν_i are injective. For any constant a the set $\mathbf{V}_a := \{v \in \mathbf{V} \mid a \in \mathbf{C}(v)\}$ induces a subtree of $\mathbf{T}$. If $u_i, u_j \in \mathbf{V}_a$ are neighbours, the definition above ensures $\nu_i(a) = \nu_j(a)$. By induction over the length of the connecting path we obtain the same for arbitrary $u_i, u_j \in \mathbf{V}_a$.

   For every node v_i there is a j_i such that $v_i = u_{j_i}$ and we set $\pi_i = \nu_{j_i}$. There are at most $2^{n\times(m+1)^m}$ distinct subsets of $cl(\psi,\{1,\dots,m,*\})$. Hence, there must be two nodes v_i, v_j such that $\pi_i(\Delta(v_i)) = \pi_j(\Delta(v_j))$ and, w.l.o.g., $\mathbf{N}(v_i) < \mathbf{N}(v_j)$. We show that v_j is blocked by v_i via $\pi := \pi_i^{-1} \circ \pi_j$. Note that for π to be well-defined, π_i must be injective. By construction, π preserves shared constants. It remains to be shown that $\pi(\Delta(v_j)) = \Delta(v_i)|_{\pi(\mathbf{C}(v_j))}$. Let $\varphi(\mathbf{a}) \in \Delta(v_j)$. Since $\pi_j(\varphi(\mathbf{a})) \in \pi_j(\Delta(v_j)) = \pi_i(\Delta(v_i))$ there is a $\mathbf{b} \in \mathbf{C}(v_i)$ with $\varphi(\mathbf{b}) \in \Delta(v_i)$ and $\pi_i(\mathbf{b}) = \pi_j(\mathbf{a})$. By definition of π we have $\pi(\mathbf{a}) = \mathbf{b}$ and thus $\pi(\varphi(\mathbf{a})) \in \Delta(v_i)|_{\pi(\mathbf{C}(v_j))}$. Conversely, let $\varphi(\mathbf{b}) \in \Delta(v_i)|_{\pi(\mathbf{C}(v_j))}$. Since $\pi_i(\varphi(\mathbf{b})) \in \pi_i(\Delta(v_i)) = \pi_j(\Delta(v_j))$ there is a $\mathbf{a} \in \mathbf{C}(v_j)$ with $\varphi(\mathbf{a}) \in \Delta(v_j)$ and $\pi_j(\mathbf{a}) = \pi_i(\mathbf{b})$. By construction of π this implies $\pi(\mathbf{a}) = \mathbf{b}$ and hence $\varphi(\mathbf{b}) \in \pi(\Delta(v_j))$.

■

Lemma 3.8 (Termination). *Let $\psi \in$ CGF be a closed formula in NNF. Any sequence of rule application of the tableau algorithm starting from the initial tree terminates.*

Proof: For any completion tree $\mathbf{T}$ generated by the tableau algorithm, we define $\|\cdot\| : \mathbf{V} \mapsto \mathbb{N}^3$ by

$$\begin{aligned}\|v\| := (&|\mathbf{C}(v)|,\ \ n \times (m+1)^m - |\Delta(v)|,\\ &|\{\varphi \in \Delta(v) \mid \varphi \text{ triggers the R}\exists\text{-r. for } v\}|).\end{aligned}$$

The lexicographic order $\prec$ on $\mathbb{N}$ is well-founded, i.e. it has no infinite decreasing chains. Any rule application decreases $\|v\|$ w.r.t. $\prec$ for at least one node v, and never increases $\|v\|$ w.r.t. $\prec$

for an existing node v. However it may create a new node w. Hence, there can only be a finite number of applications of rules to every node in $\mathbf{T}$ and an infinite sequence of rule applications would generate an infinite tree. As a corollary of 3.7, we have that the depths of $\mathbf{T}$ is bounded by $2^{n\times(m+1)^m}+1$, since, on any directed path of that length, there must be a blocked node. Rules are never applied to blocked nodes, so paths with blocked nodes can not grow in length. Hence, $\mathbf{T}$ can only be infinite due to an infinite branching in $\mathbf{T}$. Any successor of a node v is generated by application of the $\mathsf{R}\exists$-rule to v. Each such application generates exactly one successor. Hence, for $\mathbf{T}$ to be inifinite, there must be an infinite number of applications of the $\mathsf{R}\exists$-rule to a node v. As each such application decreases $\|v\|$ we have a contradiction. ■

3.2 Correctness

In order to prove the correctness of the tableau algorithm we have to show that the existence of a tableau for ψ implies satisfiability of ψ. To this purpose, we will construct a model from a tableau. In the following, let $\psi \in \mathrm{CGF}[\tau]$ and let $\mathbf{T} = (\mathbf{V}, \mathbf{E}, \mathbf{C}, \Delta, \mathbf{N})$ be a tableau for ψ. W.l.o.g., we assume, for every node $v \in \mathbf{V}$ and every $a \in \mathbf{C}(v)$, $a = a \in \Delta(v)$. For every blocking situation we fix a mapping π verifying this blocking.

Definition 3.9. We make the blocking relation explicit. For every blocked node v there is a unique blocking node w and we define $\mathbf{B}$ as set of all such pairs (v, w).

Further define $\mathbf{C}(\mathbf{V}) := \bigcup\{\mathbf{C}(v) \ : \ v \in \mathbf{V},\ v \text{ not blocked}\}$. The equivalence relation $\sim$ on $\mathbf{C}(\mathbf{V})$ is the reflexive and transitive closure of the set of all pairs of constants (c, d), where $c \in \mathbf{C}(u)$ and $d \in \mathbf{C}(v)$ for two nodes u and v, $(u, v) \in \mathbf{B}$ and the function π that verifies the blocking maps d to c.

We also use $\sim$ as an operator that maps a constant a to its $\sim$-class $\tilde{a}$. For tuples of constants $\mathbf{a}$, this operation is performed componentwise. We say that $\tilde{\mathbf{a}} \subseteq \mathbf{C}(v)$, if for each $a \in \mathbf{a}$ there is an $a' \in \tilde{a} \cap \mathbf{C}(v)$.

Definition 3.10. Let $v, w \in \mathbf{V}$ and $a \in \mathbf{C}(v)$, $b \in \mathbf{C}(w)$. An *(a, b)-path* in $\mathbf{T}$ is a sequence $(s_1, c_1), \ldots, (s_k, c_k)$ in $\mathbf{V} \times \mathbf{C}(\mathbf{V})$ such that $c_1 = a$, $c_k = b$ and for all $1 \le i < k$ one of the following holds.

1. $(s_i, s_{i+1}) \in \mathbf{E}$ and $c_i = c_{i+1}$
2. $(s_i, s_{i+1}) \in \mathbf{B}$ and $\pi(c_{i+1}) = c_i$
3. 1. and 2. for reversed roles of i and $i + 1$.

That is, an (a, b)-path verifies $a \sim b$. If p is such an (a, b)-path, the projection of p, $\rho(p) = s_1, \ldots, s_k$, is the sequence of nodes encountered along p.

The general idea in the construction of a model from a tableau, is to use $\mathbf{C}(\mathbf{V})/\!\sim$ as the universe and define the relations using the atomic constraints in the nodes. In general, there may be two kinds of problematic situations in a tableau that make this construction impossible, namely *dormant clashes* and *evil cliques*.

Definition 3.11 (Dormant Clash). *Two distinct nodes $v, w \in \mathbf{V}$, two tuples of constants $\mathbf{a}, \mathbf{b}$ and a positive literal β form a* dormant clash *$(v, w, \mathbf{a}, \mathbf{b}, \beta)$ in $\mathbf{T}$, if $\mathbf{a} \in \mathbf{C}(v)$, $\mathbf{b} \in \mathbf{C}(w)$ and it is the case that $\mathbf{a} \neq \mathbf{b}$, but $\mathbf{a} \sim \mathbf{b}$ and either $\beta(\mathbf{a}) \in \Delta(v)$ and $\beta(\mathbf{b}) \notin \Delta(w)$ or $\beta(\mathbf{a}) \notin \Delta(v)$ and $\beta(\mathbf{b}) \in \Delta(w)$.*

Note that for each dormant clash $(v, w, \mathbf{a}, \mathbf{b}, \beta)$, the intersection of the sets $P_i = \{p : p \text{ is an } (a_i, b_i)\text{-path}\}$, $1 \leq i \leq |\mathbf{a}|$, is empty. Any path included in *all* P_i would succesively let the complete atomic information about $\mathbf{a}$ and $\mathbf{b}$ be propagated from v to w using $\mathsf{R}\updownarrow$, either producing a true clash or contradicting the definition of a dormant clash.

Further, there are constants $a_t \in \mathbf{a}$ and $b_t \in \mathbf{b}$, $a_t \neq b_t$ but $a_t \sim b_t$, such that for some $(s_i, c_i), (s_{i+1}, c_{i+1})$ on every (a_t, b_t)-path, either s_i is blocked by s_{i+1} (or vice versa) and the belonging injection π maps c_i to c_{i+1} (c_{i+1} to c_i), or there is a node s blocking both s_i and s_{i+1} such that for the respective injections $\pi_{s_i} : \mathbf{C}(s_i) \to \mathbf{C}(s)$ and $\pi_{s_{i+1}} : \mathbf{C}(s_{i+1}) \to \mathbf{C}(s)$ we have $\pi_{s_i}(c_1) = \pi_{s_{i+1}}(c_{i+1})$. It follows that $\mathbf{B}$ contains (s_i, s_{i+1}) (or (s_{i+1}, s_i)) in the first and both (s, s_i) and (s, s_{i+1}) in the second case.

Definition 3.12 (Evil Clique). *An* evil clique $(\mathbf{a}, \mathbf{b}, \alpha)$ *in* $\mathbf{T}$ *consists of two sequences of constants* $\mathbf{a}$ *and* $\mathbf{b}$ *and a guard* α *occuring in some subformula* $\varphi(\mathbf{x}) = (\forall \mathbf{yz}.\alpha(\mathbf{x}, \mathbf{y}, \mathbf{z}))\eta(\mathbf{x}, \mathbf{y})$ *of* ψ *such that*

- $\mathbf{a}$ *and* $\varphi(\mathbf{a})$ *occur in the* $\mathbf{C}$ *resp. the* Δ*-label of some node in* $\mathbf{V}$*,*
- *for each* $\beta \in \alpha$ *there are a node* w*, some constants* $\mathbf{a}' \sim \mathbf{a}$ *and* $\mathbf{b}' \sim \mathbf{b}$ *such that* $\beta(\mathbf{a}', \mathbf{b}', * \cdots *) \in^* \Delta(w)$*,*
- *there is no node* v*, constants* $\mathbf{a}' \sim \mathbf{a}$ *and* $\mathbf{b}' \sim \mathbf{b}$ *such that* $\varphi(\mathbf{a}') \in \Delta(v)$ *and* $\beta(\mathbf{a}', \mathbf{b}', * \cdots *) \in^* \Delta(v)$ *for all* $\beta \in \alpha$*.* [2]

Evil cliques are also a side-effect of folding back the completion tree into itself via $\mathbf{B}$-edges, but are not required in the sense that all existential subformula of ψ can be satisfied elsewhere. To see this, assume to the contrary that we do not use $\mathbf{B}$-edges and $\mathbf{a}$ is always chosen for $\mathbf{a}'$ and $\mathbf{b}$ for $\mathbf{b}'$ in above definition. Since α is a clique-formula, every pair of constants from $\mathbf{a} \cup \mathbf{b}$ occur in at least one atom, and hence occur together in the $\mathbf{C}$-label of some node. So for any pair of constants c_1, c_2 from $\mathbf{a} \cup \mathbf{b}$ the sets $\mathbf{V}_{c_1}$ and $\mathbf{V}_{c_2}$ are subtrees of $(\mathbf{V}, \mathbf{E})$, and do not require $\mathbf{B}$-edges for their connectedness. It is a well known result in graph theory, that any family of pairwise overlapping trees has a common node—remember that $\mathbf{V}$ with only the $\mathbf{E}$-edges is a tree. Consequently $\mathbf{a} \cup \mathbf{b}$ is a subset of the $\mathbf{C}$-label of this common node.

Therefore, given an evil clique $C = (\mathbf{a}, \mathbf{b}, \alpha)$, we can always find a set of constants $C_C \subseteq \mathbf{a} \cup \mathbf{b}$ and a set of constants $D_C \subseteq \{c : \tilde{c} \in \tilde{\mathbf{a}} \cup \tilde{\mathbf{b}}\} \setminus (\mathbf{a} \cup \mathbf{b})$ such that, if for each $c_C \in C_C$ and $d_C \in D_C$ where $c_C \sim d_C$ we remove a set of $\mathbf{B}$-edges such that no (c_C, d_C)-path is left over, $(\mathbf{a}, \mathbf{b}, \alpha)$ is no longer an evil clique.

We isolate a set of edges as responsible for the two types of problematic situations defined above.

Definition 3.13. Given a tableau $\mathbf{T}$, the set of *critical edges* of $\mathbf{T}$, $S = S(\mathbf{T})$, is a subset of $\mathbf{B}$ defined as follows.

- For each dormant clash $C = (v, w, \mathbf{a}, \mathbf{b}, \beta)$ we choose an index t such that for $a_t \in \mathbf{a}$ and $b_t \in \mathbf{b}$ we have $a_t \neq b_t$. Let S contain the first $\mathbf{B}$-edge from each (a_t, b_t)-path.
- For each evil clique $C = (\mathbf{a}, \mathbf{b}, \alpha)$ we consider the constants C_C and D_C. For each $c_C \in C_C$ and $d_C \in D_C$ where $c_C \sim d_C$ let S again contain the first $\mathbf{B}$-edge from every (c_C, d_C)-path.

[2]By abuse of notation we write $\beta(\mathbf{a}', \mathbf{b}', * \cdots *)$ and $\mathbf{a}' \sim \mathbf{a}$, even though not all elements of $\mathbf{a}'$ need to occur in β and $\mathbf{a}'$ may in general be shorter than $\mathbf{a}$. The same applies to $\mathbf{b}'$.

By making enough (but finitely many) isomorphic copies of all subtrees of the tableau below the root, it is possible to redirect all critical edges into different copies in a manner that gets rid of all (isomorphic copies of) dormant clashes and evil cliques.

Lemma 3.14. *If there is a finite tableau $\mathbf{T}$ for ψ, then there is also a finite tableau $\mathbf{T}'$ for ψ that does not contain critical edges (and hence no dormant clashes or evil cliques).*

In both cases let S_C contain the edges introduced into S by C.

Note that no $\mathbf{B}$-edge can be incident to the root of $\mathbf{T}$, since all other nodes contain at least one constant in their $\mathbf{C}$-label.

Proof: Let λ be the root of $\mathbf{T}$ and let $n = |S|$. We make an enlarged version $\mathbf{T}'$ of $\mathbf{T}$ where all subtrees with roots that are direct $\mathbf{E}$-successors of λ are replaced by 2^n isomorphic copies. More precisely

- $\mathbf{V}' = \{\lambda\} \cup \{v^i \ : \ v \in \mathbf{V} \setminus \{\lambda\}\}$,
- $\mathbf{E}' = \{(\lambda, v^i) \ : \ (\lambda, v) \in \mathbf{E}\} \cup \{(v^i, w^i) \ : \ (v, w) \in \mathbf{E}\}$,
- $\mathbf{C}'(v^i) = \{a^i \ : \ a \in \mathbf{C}(v)\}$, $\mathbf{C}'(\lambda) = \emptyset$,
- $\Delta'(v^i) = \{\varphi(\mathbf{a}^i) \ : \ \varphi(\mathbf{a}) \in \Delta(v)\}$, $\Delta'(\lambda) = \Delta(\lambda)$,
- $\mathbf{N}'(v^i) = 2^n \cdot \mathbf{N}(v) + i$, $\mathbf{N}'(\lambda) = 0$,

for all $0 \leq i < 2^n$. The blocking relation $\mathbf{B}'$ is given by

- $\mathbf{B}' = \bigcup \{(v^i, w^i) \ : \ (v, w) \in \mathbf{B}, \ 0 \leq i < 2^n\}$.

This first step also creates 2^n copies of our critical edges, namely all (r^i, s^i) for which $(r, s) \in S$. We now modify $\mathbf{B}'$ as follows to eliminate all copies of our original critical edges. Let $\{(r_t, s_t) \ : \ 0 \leq t < n\}$ be an enumerated version of S.

Now for all $0 \leq t < n$ and all $0 \leq \ell, j < 2^n$, if the binary representations of ℓ and j differ exactly at the t-th position we delete (r_t^ℓ, s_t^ℓ) and (r_t^j, s_t^j) from $\mathbf{B}'$ and add (r_t^ℓ, s_t^j) and (r_t^j, s_t^ℓ) in their place. Independent of the existence of critical edges in $\mathbf{T}'$, let S' be the set of $\mathbf{B}'$-edges induced by S, i.e. exactly the edges connecting some i-th and j-th copy of $\mathbf{T}$ in $\mathbf{T}'$, $i \neq j$.

This hyper-cube type of structure created by the S'-edges is crucial to the elimination of critical edges. The notation forthwith uses the convention that if X is an object related to $\mathbf{T}$, then X' is the corresponding object related to $\mathbf{T}'$ and vice versa. Also for any object X that was copied in the transition from $\mathbf{T}$ to $\mathbf{T}'$, the indexed version X^i is assumed to be the copy related to T^i.

Claim: $\mathbf{T}'$ is still a complete and clash-free finite completion tree for ψ.

Let T^i be $\mathbf{T}'$ restricted to λ and all i-th copies of the subtrees, i.e. $\mathbf{T}'$ restricted to $\{v^i \ : \ v \in \mathbf{V}\} \cup \{\lambda\}$. Further let $V_a^i = \{v^i \in T^i \ : \ a \in \mathbf{C}(v^i)\}$ and $V_{\tilde{a}}^i = \{v^i \ : \ a' \in \mathbf{C}'(v^i), \text{ex. } p \in P(a, a') \text{ s.t. } \rho(p) \subseteq T^i\}$. When disregarding S', each T^i is isomorphic to $\mathbf{T}$. Further the definition of $\mathbf{B}'$ implies that for any node v that is blocked by a node w in $\mathbf{T}$, all copies v^i are blocked by some node w^j in $\mathbf{T}'$. Consequently $\mathbf{T}'$ is complete. Since each Δ'-label in $\mathbf{T}'$ is identical to some Δ-label in $\mathbf{T}$, $\mathbf{T}'$ is also clash-free.

Note that for $i \neq j$ there are exactly two $\mathbf{B}'$-edges between T^i and T^j iff the binary representations of i and j differ at exactly one position. In all other cases there is no connecting $\mathbf{B}'$-edge.

By construction, if $(s_1^{i_1}, c_1^{i_1}) \cdots (s_k^{i_k}, c_k^{i_k})$ is an (a^i, b^j)-path in $\mathbf{T}'$, then $(s_1, c_1) \cdots (s_k, c_k)$ is an (a, b)-path in $\mathbf{T}$. Consequently, if $(v^i, w^j, \mathbf{a}^i, \mathbf{b}^j, \beta)$ is a dormant clash in $\mathbf{T}'$, then $(v, w, \mathbf{a}, \mathbf{b}, \beta)$ is a dormant clash in $\mathbf{T}$.

We can already note that if (c, d) was a pair of constants giving rise to a set of critical edges S_C, and all (c^i, d^i)-paths were subsequently modified, then there can be no (c^ℓ, d^ℓ)-path using only nodes in T^ℓ. Any such path would have an isomorphic copy in $\mathbf{T}$ leading to some edge being included in S.

Also, in the same circumstances, all (c^i, d^j)-paths contain $S'_{C'}$-edges, even if $i = j$. We just noted, that for any ℓ there never is a local, i.e. restricted to T^ℓ, connection between V_c^ℓ and V_d^ℓ. For any in this sense non-local (c^i, d^j)-path p', the isomorphic copy p in $\mathbf{T}$ is a (c, d)-path. So at least one edge along p belongs to S_C and consequently at least one edge along p' to $S'_{C'}$.

Similarly, if $C' = (\mathbf{a}^i, \mathbf{b}', \alpha)$ is an evil clique in $\mathbf{T}'$, then $(\mathbf{a}, \mathbf{b}, \alpha)$ already was an evil clique in $\mathbf{T}$ and for all $d_{C'}^j \in D'_{C'}$ we have $d_{C'} \in D_C$. We designate the first tuple of elements $\mathbf{a}^i$, because the definition of an evil clique requires all occuring elements to co-exist at some node $v = v^i$. We do not have such an assumption for the second tuple, hence the elements of $\mathbf{b}'$ may live only at different T^js.

Claim: $\mathbf{T}'$ contains no dormant clash.

Assume that $C' = (v^i, w^j, \mathbf{a}^i, \mathbf{b}^j, \beta)$ is a dormant clash in $\mathbf{T}'$ and t is the index for which the (a_t^i, b_t^j)-paths in $\mathbf{T}'$ were modified. We need to show that $\mathbf{a}^i \not\sim \mathbf{b}^j$.

We note that due to the modifications in $\mathbf{B}'$, any (a_t^i, b_t^j)-path has to use S'-edges, even in the case of $i = j$. Also as $\mathbf{C}'(\lambda) = \emptyset$ the root can not occur on any $(\cdot, \cdot)$-path. The construction of $\mathbf{B}'$ further implies that the numbers of $S'_{C'}$-edges on the paths verifying $\mathbf{a}^i \sim \mathbf{b}^j$ are either all even or all odd.

First suppose that all (a_t^i, b_t^j)-paths for C' contain an even number of $S'_{C'}$-edges. Let p be an (a_t^i, b_t^j)-path in $\mathbf{T}'$. Each $S'_{C'}$-edge leads from some $V_{a_t^\ell}^\ell$ to a $V_{\tilde{b}_t^{\ell'}}^{\ell'}$ or vice versa. The seeming asymmetry of taking a for the first, and $\tilde{b}$ for the second set of nodes, is due to the definition of critical edges selecting the *first* $\mathbf{B}$-edge from the (a_t, b_t)-paths for inclusion in S.

We conclude that if the last $S'_{C'}$ edge was taken from a $V_{a_t^\ell}^\ell$ to a $V_{\tilde{b}_t^{\ell'}}^{\ell'}$, then the next $S'_{C'}$-edge along p has to be taken from a $V_{\tilde{b}_t^{\ell''}}^{\ell''}$ to a $V_{a_t^{\ell'''}}^{\ell'''}$ and vice versa. Otherwise there would be e.g. a $(a_t^{\ell'}, b_t^{\ell''})$-path in between. By previous observation this path would necessary contain an $S'_{C'}$-edge, contradicting the choice of two $S'_{C'}$ above.

By a simple parity argument we observe that if p starts in V_a^i it can only end in V_a^j instead of the required $V_{\tilde{b}}^j$. This contradicts the assumption that p is an (a_t^i, b_t^j)-path. Hence $a^i \not\sim b^j$ and C' is not a dormant clash.

Now suppose that the (a_t^i, b_t^j)-paths for C' contain an odd number of $S'_{C'}$-edges. Let p be an (a_t^i, b_t^j)-path containing an odd number of $S'_{C'}$-edges. Then there is at least one actual edge $e' \in S'_{C'}$ that occurs an odd number of times. Suppose that e' is a copy of the edge e_ℓ, i.e. the ℓ-th edge in the enumeration of S. By the construction of $\mathfrak{T}'$ we conclude that bin(i) and bin(j) differ at position ℓ. Consequently any (a_t^i, b_t^j)-path, and indeed any path from T^i to T^j (ignoring connections via λ) has to contain an odd number of edges that are copies of e_ℓ.

We can then find an index $h \neq t$ such that *any* path using e_ℓ does *not* verify $a_h \sim b_h$. Now assume that p' is an (a_h^i, b_h^j)-path in $\mathbf{T}'$. As $a_h^i \in T^i$ and $b_h^j \in T^j$, we know that p' contains an edge that is a copy of e_ℓ. Then the isomorphic copy of p' in $\mathbf{T}$ would be an (a_h, b_h)-path in $\mathbf{T}$, a contradiction.

Finally $a_h^i \not\sim b_h^j$ implies $\mathbf{a} \not\sim \mathbf{b}$, so C' is not a dormant clash.

Claim: $\mathbf{T}'$ contains no evil clique.

This part of the proof has yet to be established in a concise way.

■

Lemma 3.15. *Let $\psi \in \mathrm{CGF}[\tau]$ and let $\mathbf{T}$ be a tableau for ψ. Then ψ is (finitely) satisfiable.*

Proof: According to Lemma 3.14 we assume $\mathbf{T} = (\mathbf{V}, \mathbf{E}, \mathbf{C}, \Delta, \mathbf{N})$ to be a tableau for ψ that does not contain critical edges.

Towards the finite satisfiability we construct a finite structure $\mathfrak{A} = \mathfrak{A}(\mathbf{T})$ with universe $A := \mathbf{C}(\mathbf{V})/{\sim}$. For each relation $R \in \tau$ and each tuple $\mathbf{a} \in A$ of matching arity let $\mathbf{a} \in R^{\mathfrak{A}}$ iff there is a node $v \in \mathbf{V}$ and a tuple of constants $\mathbf{b} \in \mathbf{C}(v)$ such that all $b_i \sim a_i$ and $R\mathbf{b} \in \Delta(v)$. Note that with R$\updownarrow$ and the non-existence of dormant clashes, this is the case iff the same holds true independent of the specific choice of $\mathbf{b}$ or v. Hence $\mathfrak{A}$ is well defined.

Claim: $\mathfrak{A} \models \psi$.

This is implied by the stronger statement that for every closed formula φ using constants from $\mathbf{a}$ that appears in the Δ-label of some unblocked node v of $\mathbf{T}$, $\varphi[\mathbf{a} \mapsto \tilde{\mathbf{a}}]$ holds in $\mathfrak{A}$. Again φ is assumed to be in NNF.

- For equality statements this is immediate. The R=-rule makes sure, that distinct constants occuring at a common node have distinct $\sim$-classes. For inequality statments, assume $a \neq b \in \Delta(v)$, but $a \sim b$. Then we can find an (a, b)-path containg a node $w \neq v$ and a constant $c \in \mathbf{C}(w)$ with $a \sim c \sim b$. Since we have assumed $c = c \in \Delta(v)$, this would imply the existence of the dormant clash $(v, w, ab, cc, c = c)$ in $\mathbf{T}$.
- For an atomic sentence $R\mathbf{a}$, we get $\mathfrak{A} \models R\tilde{\mathbf{a}}$ immediately from the construction of $\mathfrak{A}$. In case of a negated atomic sentence, assume $\varphi(\mathbf{a}) = \neg R\mathbf{a} \in \Delta(v)$ but $\mathfrak{A} \models R\tilde{\mathbf{a}}$. This implies the existence of a (dormant) clash in $\mathbf{T}$.
- For positive Boolean combinations the argument is immediate.
- Let $\varphi(\mathbf{a}) = (\exists \mathbf{y}\mathbf{z}.\alpha(\mathbf{a}, \mathbf{y}, \mathbf{z}))\eta(\mathbf{a}, \mathbf{y})$. If, for some $\mathbf{b}, \mathbf{c} \in \mathbf{C}(v)$, $\alpha(\mathbf{a}, \mathbf{b}, \mathbf{c}), \eta(\mathbf{a}, \mathbf{b}) \in \Delta(v)$, we note that $\mathfrak{A} \models \eta(\tilde{\mathbf{a}}, \tilde{\mathbf{b}})$ and $\mathfrak{A} \models \beta(\tilde{\mathbf{a}}, \tilde{\mathbf{b}}, \tilde{\mathbf{c}})$ for all $\beta \in \alpha$ by induction hypothesis for α and η.

  If there are no $\mathbf{b}, \mathbf{c} \in \mathbf{C}(v)$ with $\alpha(\mathbf{a}, \mathbf{b}, \mathbf{c}), \eta(\mathbf{a}, \mathbf{b}) \in \Delta(v)$, then application of the R$\exists$-Rule yields a successor node w of v with constants $\mathbf{b}, \mathbf{c} \in \mathbf{C}(w)$ such that $\alpha(\mathbf{a}, \mathbf{b}, \mathbf{c}), \eta(\mathbf{a}, \mathbf{b}) \in \Delta(w)$. If w is not blocked, the claim again follows by induction hypothesis for α and η.

  If however w is blocked, consider the node u with $(u, w) \in \mathbf{B}$ and the injection $\pi : \mathbf{C}(w) \to \mathbf{C}(u)$. Then $\alpha(\pi(\mathbf{a}), \pi(\mathbf{b}), \pi(\mathbf{c}))$ and $\eta(\pi(\mathbf{a}), \pi(\mathbf{b}))$ are in the Δ-label of u. Since all pairs of constants (a, a') where $a' = \pi(a)$ are in the same $\sim$-class, it follows by induction that $\mathfrak{A} \models \alpha(\tilde{\mathbf{a}}, \tilde{\mathbf{b}}, \tilde{\mathbf{c}}) \wedge \eta(\tilde{\mathbf{a}}, \tilde{\mathbf{b}})$, and hence $\varphi(\tilde{\mathbf{a}})$ holds in $\mathfrak{A}$.
- Finally let $\varphi(\mathbf{a}) = (\forall \mathbf{y}\mathbf{z}.\alpha(\mathbf{a}, \mathbf{y}, \mathbf{z}))\eta(\mathbf{a}, \mathbf{y})$. Assume that there are tuples $\mathbf{b}, \mathbf{c}$ such that $\mathfrak{A} \models \alpha(\tilde{\mathbf{a}}, \tilde{\mathbf{b}}, \tilde{\mathbf{c}})$. Since there are no evil cliques in $\mathbf{T}$, there is a node w where $\tilde{\mathbf{a}} \cup \tilde{\mathbf{b}} \subseteq \mathbf{C}(w)$,

i.e., there are tuples $\mathbf{a}', \mathbf{b}' \subseteq \mathbf{C}(w)$ with $\mathbf{a}' \sim \mathbf{a}$ and $\mathbf{b}' \sim \mathbf{b}$. Moreover, for all $\beta \in \alpha$, $\beta(\mathbf{a}', \mathbf{b}', * \cdots *) \in^* \Delta(w)$ and $\varphi(\mathbf{a}') \in \Delta(w)$. Hence, the R$\forall$-rule is appplicable for $\varphi(\mathbf{a}')$ at w and must have been applied because $\mathbf{T}$ is complete. This yields $\eta(\mathbf{a}', \mathbf{b}') \in \Delta(v)$, which, by induction yields $\mathfrak{A} \models \eta(\mathbf{a}', \mathbf{b}')$ and hence $\mathfrak{A} \models \eta(\mathbf{a}, \mathbf{b})$. ∎

3.3 Completeness

Lemma 3.16. *Let $\psi \in$ CGF be a closed formula in NNF. If ψ is satisfiable, then there is a sequence of rule applications starting from the initital tree that yields a tableau.*

Proof: Since ψ is satisfiable, there is a model $\mathfrak{A}$ of ψ. We will use $\mathfrak{A}$ to guide the application of the non-deterministic R$\vee$-rule. For this we incremently define a function $g : \bigcup\{\mathbf{C}(v) \mid v \in \mathbf{V}\} \to A$ such that for all $v \in \mathbf{V} \ : \ \mathfrak{A} \models g(\Delta(v))$. We refer to this property by $(*)$.

The set $\Delta(v)$ can contain atomic formulas $\alpha(\mathbf{a}^*)$ where $*$ occurs at some positions of $\mathbf{a}^*$ and is not mapped to an element of A by g. We deal with this as described under Definition 3.3 by setting

$$\mathfrak{A} \models g(\alpha(\mathbf{a})) \ \text{ iff } \ \mathfrak{A} \models \exists \mathbf{z}.g(\alpha(\mathbf{a}')).$$

Claim 1: If for a completion tree $\mathbf{T}$ there exists a function g sucht that $(*)$ holds and a rule is applicable to $\mathbf{T}$, then it can be applied in a way that maintains $(*)$.

We distinguish the different rules.

- If the R$\wedge$-rule is applicable to a node $v \in \mathbf{V}$ with $\varphi \wedge \vartheta \in \Delta(v)$ then, due to $(*)$, $\mathfrak{A} \models g(\varphi \wedge \vartheta)$ and hence $\mathfrak{A} \models \{g(\varphi), g(\vartheta)\}$. Hence, the R$\wedge$-rule can be applied to v without violating $(*)$.
- If the R$\vee$-rule is applicable to a node $v \in \mathbf{V}$ with $\varphi \vee \vartheta \in \Delta(v)$ then, due to $(*)$, $\mathfrak{A} \models g(\varphi \vee \vartheta)$ and hence $\mathfrak{A} \models g(\chi)$ for a $\chi \in \{\varphi, \vartheta\}$. Hence, the R$\vee$-rule can be applied to v without violating $(*)$.
- If the R$=$-rule is applicable to a node $v \in \mathbf{V}$ with $a = b \in \Delta(v)$, then $\mathfrak{A} \models g(b) = g(b)$ implies $g(a) = g(b)$. Hence, for every node w that shares a with v, $g(\Delta(w)) = g(\Delta(w)[a \mapsto b])$ and the rule can be applied without violating $(*)$.
- If the R$\forall$-rule is applicable to a node $v \in \mathbf{V}$ with $(\forall \mathbf{yz}.\alpha(\mathbf{a}, \mathbf{y}, \mathbf{z}))\varphi(\mathbf{a}, \mathbf{y}) \in \Delta(v)$, then there is $\mathbf{b} \subseteq \mathbf{C}(v)$ such that, for all atoms $\beta(\mathbf{x}, \mathbf{y}, \mathbf{z}) \in \alpha$, $\beta(\mathbf{a}, \mathbf{b}, * \cdots *) \in^* \Delta(v)$. Hence, from the definition of $\in^*$, there is a tuple $\mathbf{c} \subseteq \mathbf{C}(v) \cup \{*\}$ such that $\beta(\mathbf{a}, \mathbf{b}, * \cdots *) \geq^* \beta(\mathbf{a}, \mathbf{b}, \mathbf{c})$ and $\beta(\mathbf{a}, \mathbf{b}, \mathbf{c}) \in \Delta(v)$. From $(*)$ we get that $\mathfrak{A} \models \exists \mathbf{z}.\beta(g(\mathbf{a}), g(\mathbf{b}), \mathbf{z})$ and since every z appears in exactly one atom in α, also $\mathfrak{A} \models \exists \mathbf{z}.\alpha(g(\mathbf{a}), g(\mathbf{b}), \mathbf{z})$. Hence, we have

$$\begin{aligned}\{\mathfrak{A} \models \{&\forall \mathbf{y}.(\exists \mathbf{z}.\alpha(g(\mathbf{a}), \mathbf{y}, \mathbf{z}) \to \varphi(g(\mathbf{a}), \mathbf{y})),\\ &\exists \mathbf{z}.\alpha(g(\mathbf{a}), g(\mathbf{b}), \mathbf{z})\}\end{aligned}$$

  which, by Lemma 2.3, implies $\mathfrak{A} \models \varphi(g(\mathbf{a}), g(\mathbf{b}))$ and hence $\varphi(\mathbf{a}, \mathbf{b})$ can be added to $\Delta(v)$ without violating $(*)$.
- If the R$\exists$-rule is applicable to a node $v \in \mathbf{V}$ with $(\exists \mathbf{yz}.\alpha(\mathbf{a}, \mathbf{y}, \mathbf{z}))\varphi(\mathbf{a}, \mathbf{y})$, then this implies

$$\mathfrak{A} \models g((\exists \mathbf{yz}.\alpha(\mathbf{a}, \mathbf{y}, \mathbf{z}))\varphi(\mathbf{a}, \mathbf{y})).$$

Hence, there are sequences $\mathbf{b}', \mathbf{c}' \subseteq A$ of elements such that $\mathfrak{A} \models \{\alpha(g(\mathbf{a}), \mathbf{b}', \mathbf{c}'), \varphi(g(\mathbf{a}), \mathbf{b}')\}$. If we define g such that $g(\mathbf{b}) = \mathbf{b}'$ and $g(\mathbf{c}) = \mathbf{c}'$, then obviously $\mathfrak{A} \models \{g(\alpha(\mathbf{a}, \mathbf{b}, \mathbf{c}), g(\varphi(\mathbf{a}, \mathbf{b}))\}$. Note, that this might involve setting $g(b_1) = g(b_2)$ for some $b_1, b_2 \in \mathbf{b}$. With this construction the resulting extended completion-tree $\mathbf{T}$ and extended function g again satisfy $(*)$.

- If the $\mathsf{R}\updownarrow$-rule is applicable to a node $v \in \mathbf{V}$ with $\alpha(\mathbf{a}^*) \in \Delta(v)$ and a neighbour w with $\mathbf{a}^* \cap \mathbf{C}(w) \neq \emptyset$, then it adds $\alpha(\mathbf{a}^*)|^*_{\mathbf{C}(w)}$ to $\Delta(w)$. From $(*)$ get that $\mathfrak{A} \models \alpha(g(\mathbf{a}^*))$, and since $\alpha(\mathbf{a}^*)|^*_{\mathbf{C}(w)} \geq^* \alpha(\mathbf{a}^*)$, which implies $\mathfrak{A} \models \alpha(g(\mathbf{a}^*))|^*_{\mathbf{C}(w)}$. Hence, adding $\alpha(\mathbf{a}^*)|^*_{\mathbf{C}(w)}$ to $\Delta(w)$ does not violate $(*)$.
- If the $\mathsf{R}\updownarrow\forall$-rule is applicable to a node $v \in \mathbf{V}$ with a universal formula $\varphi(\mathbf{a}) \in \Delta(v)$ and a neighbour w which shares $\mathbf{a}$ with v, $(*)$ yields $\mathfrak{A} \models \varphi(g(\mathbf{a}))$. Hence, adding $\varphi(\mathbf{a})$ to $\Delta(w)$ does not violate $(*)$.

CLAIM 2: A completion-tree $\mathbf{T}$ for which a function g exists such that $(*)$ holds is clash free.

Assume that $\mathbf{T}$ contains a clash, namely, there is a node $v \in \mathbf{V}$ such that either $a \neq a \in \mathbf{V}(v)$— implying $\mathfrak{A} \models g(a) \neq g(a)$—, or that there is a sequence $\mathbf{a} \subseteq \mathbf{C}(v)$, and an atomic formula φ such that $\{\alpha(\mathbf{a}), \neg\alpha(\mathbf{a})\} \subseteq \Delta(v)$. From $(*)$ it would follow that $\mathfrak{A} \models \{\alpha(g(\mathbf{a})), \neg\alpha(g(\mathbf{a}))\}$, also a contradiction.

These claims yield Lemma 3.16 as follows. Let $\mathbf{T}$ be a tableau for ψ. Since $\mathfrak{A} \models \psi$, $(*)$ is satisfied for initial tree together with the empty function g. By Theorem 3.8, any sequence of applications is finite, and from Claim 1 we get that there is a sequence of rule-applications that maintains $(*)$. By Claim 2, this sequence results in a tableau. ■

Lemma 3.16 involves two different kinds of non-determinism, namely, the choice which rule to apply to which constraint (as several rules can be applicable simultaneously), and which disjunct to choose in an application of the $\mathsf{R}\vee$-rule. While the latter choice is *don't-know* non-detemistic, i.e., for a satisfiable formula only certain choices will lead to the discovery of a tableau, the former choice is *don't-care* non-deterministic. This means that arbitrary choices of which rule to apply next will lead to the discovery of a tableau for a satisfiable formula. For an implementation of the tableau algorithm this has the following consequences. Exhaustive search is necessary to deal with all possible expansions of the $\mathsf{R}\vee$-rule, but arbitrary strategies of choosing which rule to apply next and where will lead to a correct implementation, although the efficiency of the implementation will very much depend on a suitable strategy.

3.4 Tree Model Property

The fact that every satisfiable formula of width k has a model of width at most $k-1$ was the starting point for our considerations. Yet, this fact was never relied on to prove the correctness of the tableaux algorithm. Indeed, it is possible to give an alternative proof for Fact 3.2 based on our tableaux algorithm. This requires an alternative construction to the one used in the proof of Lemma 3.15. Note that this proof is also an alternative proof for Lemma 3.15.

Theorem 3.17. *Let $\psi \in$ CGF with $k = \mathrm{width}(\psi)$. ψ is satisfiable iff ψ has a model of width at most $k-1$.*

Proof: If ψ is satisfiable, then the tableaux algorithm generates a tableau for ψ. Using an unraveling construction, we will construct a model for ψ of width at most $k-1$ from $\mathbf{T}$.

Let $\mathbf{V}_u = \{v \in \mathbf{V} \mid v \text{ is not indirectly blocked }\}$ and $\mathsf{Paths}(\mathbf{T}) \subseteq \mathbf{V}_u^+$ inductively defined by [3]

- $[\frac{v_0}{v_0}] \in \mathsf{Paths}(\mathbf{T})$ for the root v_0 of $\mathbf{T}$,
- if $[\frac{v_1}{v_1'} \dots \frac{v_n}{v_n'}] \in \mathsf{Paths}(\mathbf{T})$, w is a successor of v_n and w is not blocked, then $[\frac{v_1}{v_1'} \dots \frac{v_n}{v_n'} \frac{w}{w}] \in \mathsf{Paths}(\mathbf{T})$,
- if $[\frac{v_1}{v_1'} \dots \frac{v_n}{v_n'}] \in \mathsf{Paths}(\mathbf{T})$, w is a successor of v_n blocked by the node $u \in \mathbf{V}$, then $[\frac{v_1}{v_1'} \dots \frac{v_n}{v_n'} \frac{u}{w}] \in \mathsf{Paths}(\mathbf{T})$.

The set $\mathsf{Paths}(\mathbf{T})$ forms a tree, with p' being a successor of p if p' is obtained from p by concatenating one element $\frac{u}{w}$ at the end. We define the auxiliary functions $\mathsf{Tail}, \mathsf{Tail}'$ by setting $\mathsf{Tail}(p) = v_n$ and $\mathsf{Tail}'(p) = v_n'$ for every path $p = [\frac{v_1}{v_1'} \dots \frac{v_n}{v_n'}]$. We further define

$$\mathbf{C}(\mathbf{T}) = \{(a,p) \mid p \in \mathsf{Paths}(\mathbf{T}) \wedge a \in \mathbf{C}(\mathsf{Tail}(p))\}$$

and the relation $\sim$ as the smallest symmetric relation on $\mathbf{C}(\mathbf{T})$ satisfying

- $(a,p) \sim (a,q)$ if $\mathsf{Tail}'(q)$ is an unblocked successor of $\mathsf{Tail}(p)$ and $a \in \mathbf{C}(\mathsf{Tail}(p)) \cap \mathbf{C}(\mathsf{Tail}'(q))$,
- $(a,p) \sim (b,q)$ if $\mathsf{Tail}'(q)$ is a blocked successor of $\mathsf{Tail}(p)$, $a \in \mathbf{C}(\mathsf{Tail}(p)) \cap \mathbf{C}(\mathsf{Tail}'(q))$ and $\pi(a) = b$ for the function π that verifies that $\mathsf{Tail}'(q)$ is blocked by $\mathsf{Tail}(q)$.

With $\approx$ we denote the reflexive, transitive closure of $\sim$. First we need to prove some technicalities for this unraveling.

Claim 1: Let $p \in \mathsf{Paths}(\mathbf{T})$ and $a, b \in \mathbf{C}(\mathsf{Tail}(p))$. Then $(a,p) \approx (b,p)$ iff $a = b$.

Assume the claim does not hold and let $a \neq b$ with $(a,p) \approx (b,p)$. By definition of $\sim$, $(a,p) \not\sim (b,p)$ must hold. Hence, there must be a path $(c_1,p_1) \sim \dots \sim (c_k,p_k)$ such that $a = c_1$, $b = c_k$, and $p = p_1 = p_k$. W.l.o.g., assume we have picked a, b, p such that this path has minimal length k. Such a minimal path must be of length $k = 3$, for if we assume a path of length $k > 3$, there must be $2 \leq i < j \leq k-1$ such that $p_i = p_j$, because the relation $\sim$ is defined along paths in the tree $\mathsf{Paths}(\mathbf{T})$. If $c_i = c_j$ then we can shorten the path between position i and j and obtain a shorter path. If $c_i \neq c_j$, then the path $(c_i,p_i) \sim \dots \sim (c_j,p_j)$ is also a shorter path with the same properties. Hence, a minial path must be of the form $(a,p) \sim (c,q) \sim (b,p)$. If $\mathsf{Tail}'(q)$ is not blocked, by the definition of $\sim$, $a = c = b$ must hold. Hence, since $a \neq b$, $\mathsf{Tail}'(q)$ must be blocked by $\mathsf{Tail}(q)$. From the definition of $\sim$ we have $a, b \in \mathbf{C}(\mathsf{Tail}'(q))$ and $\pi(a) = c = \pi(b)$ for the function π verifying that $\mathsf{Tail}'(q)$ is blocked by $\mathsf{Tail}(q)$. Since π must be injective, this is a contradiction.

Since the set $\mathsf{Paths}(\mathbf{T})$ is a tree, and as a consequence of Claim 1 we get the following:

Claim 2: Let $p, p' \in \mathsf{Paths}(\mathbf{T})$ with $p = [\frac{v_1}{v_1'} \dots \frac{v_n}{v_n'}]$, $p' = [\frac{v_1}{v_1'} \dots \frac{v_n}{v_n'} \frac{w}{w'}]$. If, for $a \in \mathbf{C}(v_n), b \in \mathbf{C}(w)$, $(a,p) \approx (b,p')$ then $(a,p) \sim (b,p')$.

[3]This complicated for of unraveling, where we record both blocked an blocking node is necessary because there might be a situation where two successors v_1, v_2 of a node are blocked by the same node w.

If $(a,p) \approx (b,p')$ then there must be a path $(c_1,p_1) \sim \cdots \sim (c_k,p_k)$ such that $a = c_1$, $b = c_k$, $p = p_1$, and $p' = p_k$. Since $\sim$ is only defined along paths in the tree $\mathsf{Paths}(\mathbf{T})$, there must be a step from p to p' (or, dually, from p' to p) in this path, more precisely, there must be an $i \in \{1, \dots k-1\}$ such that $p_i = p$ and $p_{i+1} = p'$ holds. Hence, we have the situation

$$(a,p) \approx (c_i,p) \sim (c_{i+1},p') \approx (b,p')$$

Now Claim 1 implies $a = c_i$ and $b = c_{i+1}$ and hence $(a,p) \sim (b,p')$.

Using Claim 2, we can show that the blocking condition and the $\mathsf{R}\updownarrow$- and $\mathsf{R}\updownarrow\forall$-rule work as desired:

Claim 3: Let $p,q \in \mathsf{Paths}(\mathbf{T})$, $\mathbf{a} \subseteq \mathbf{C}(\mathsf{Tail}(p)), \mathbf{b} \subseteq \mathbf{C}(\mathsf{Tail}(q)))$ and $(\mathbf{a},p) \approx (\mathbf{b},q)$.

- For every atomic formula β, $\beta(\mathbf{a}, * \cdots *) \in^* \Delta(\mathsf{Tail}(p))$ iff $\beta(\mathbf{b}, * \cdots *) \in^* \Delta(\mathsf{Tail}(q))$.
- For every universal formula φ, $\varphi(\mathbf{a}) \in \Delta(\mathsf{Tail}(p))$ iff $\varphi(\mathbf{b}) \in \Delta(\mathsf{Tail}(q))$.

Equivalence classes of $\approx$ induce subtrees of the tree $\mathsf{Paths}(\mathbf{T})$, hence, if $(\mathbf{a},p) \approx (\mathbf{b},q)$, then there must be a path $(\mathbf{c}_1,p_1) \approx \cdots \approx (\mathbf{c}_k,p_k)$ with $p_1 = p, p_k = q$, $\mathbf{a} = \mathbf{c}_1$, $\mathbf{b} = \mathbf{c}_m$, and p_i is a neighbour of p_{i+1} in the tree $\mathsf{Paths}(\mathbf{T})$. From Claim 2, we get that any two neighbours $p_i, p_i + 1$ in $\mathsf{Paths}(\mathbf{T})$, $(\mathbf{c}_i,p_i) \approx (\mathbf{c}_{i+1},p_{i+1})$ implies $(\mathbf{c}_i,p_i) \sim (\mathbf{c}_{i+1},p_{i+1})$.

W.o.l.g., assume p_{i+1} is a successor of p_i in the tree $\mathsf{Paths}(\mathbf{T})$ and set $v = \mathsf{Tail}(p_i)$ and $w = \mathsf{Tail}(p_{i+1})$. There are two possibilities:

- if $\mathsf{Tail}'(p_{i+1})$ is not blocked, then $\mathsf{Tail}(p_{i+1}) = \mathsf{Tail}'(p_{i+1})$ and by the definition of $\sim$, $\mathsf{Tail}(p_{i+1})$ is a successor of $\mathsf{Tail}(p_i)$ in $\mathbf{T}$ and $\mathbf{c}_i = \mathbf{c}_{i+1}$. Due to the $\mathsf{R}\updownarrow$-rule, $\beta(\mathbf{c}_i, * \cdots *) \in^* \Delta(\mathsf{Tail}(p_i))$ iff $\beta(\mathbf{c}_{i+1}, * \cdots *) \in^* \Delta(\mathsf{Tail}(p_{i+1}))$. Due to the $\mathsf{R}\updownarrow\forall$-rule, $\varphi(\mathbf{c}_i) \in \Delta(\mathsf{Tail}(p_i))$ iff $\varphi(\mathbf{c}_{i+1}) \in \Delta(\mathsf{Tail}'(p_{i+1})) = \Delta(\mathsf{Tail}(p_{i+1}))$.
- if $\mathsf{Tail}'(p_{i+1})$ is blocked by $\mathsf{Tail}(p_{i+1})$ and $\mathsf{Tail}'(p_{i+1})$ is a successor of $\mathsf{Tail}(p_i)$ in $\mathbf{T}$. Then, by definition of $\sim$ we have $\mathbf{c}_i \subseteq \mathbf{C}(\mathsf{Tail}(p_i)) \cap \mathbf{C}(\mathsf{Tail}'(p_{i+1}))$ and due to the $\mathsf{R}\updownarrow$- and $\mathsf{R}\updownarrow\forall$-rule, for any atomic or universal formula φ, $\varphi(\mathbf{c}_i) \in \Delta(\mathsf{Tail}(p_i))$ iff $\varphi(\mathbf{c}_i) \in \Delta(\mathsf{Tail}'(p_{i+1}))$. Furthermore, for the function π verifying that $\mathsf{Tail}'(p_{i+1})$ is blocked by $\mathsf{Tail}(p_{i+1})$, we have that $\pi(\Delta(\mathsf{Tail}'(p_{i+1}))) = \Delta(\mathsf{Tail}(p_{i+1}))|_{\pi(\mathbf{C}(\mathsf{Tail}'(p_{i+1})))}$ and hence for every formula φ, $\varphi(\mathbf{c}_i) \in \Delta(\mathsf{Tail}(p))$, $\varphi(\mathbf{c}_i) \in \Delta(\mathsf{Tail}'(p_{i+1}))$ iff $\pi(\varphi(\mathbf{c}_i)) = \varphi(\mathbf{c}_{i+1}) \in \Delta(\mathsf{Tail}(p_{i+1}))$.

Due to Claim 3, we can now define a structure $\mathfrak{A}$ over the universe $A = \mathbf{C}(\mathbf{T})/{\approx}$ by setting, for a relation $R \in \tau$ of arity m, $([a_1,p_1]_\approx, \dots, [a_m,p_m]_\approx) \in R^{\mathfrak{A}}$ iff there is a path $p \in \mathsf{Paths}(\mathbf{T})$ and constants $c_1, \dots c_m$ such that $(c_i,p) \in [a_i,p_i]_\approx$ and $Rc_1 \dots c_m \in \Delta(\mathsf{Tail}(p))$.

To simplify things we define the following examples of "abuses of notation". Let $\mathbf{a} = a_1, \dots, a_n$ be a sequence of constants, $\mathbf{p} = p_1, \dots, p_n$ a sequence of paths of matching length, and q a single path. We define

$$[\mathbf{a},\mathbf{p}]_\approx = ([a_1,p_1]_\approx, \dots, [a_n,p_n]_\approx)$$
$$[\mathbf{a},q]_\approx = ([a_1,q]_\approx, \dots, [a_n,q]_\approx)$$

It remains to show that this construction yields $\mathfrak{A} \models \psi$. This is a consequence of the following claim that can be shown by induction over the structure of the formula φ.

Claim 4: For every path $p \in \mathsf{Paths}(\mathbf{T})$ and $\mathbf{a} \subseteq \mathbf{C}(\mathsf{Tail}(p))$, if $\varphi(\mathbf{a}) \in \Delta(\mathsf{Tail}(p))$, then $\mathfrak{A} \models \varphi([\mathbf{a}, p]_\approx)$.

We show this claim by induction on the structure of formulas φ.

- If $\varphi(\mathbf{a}) = Ra_1 \dots a_m \in \Delta(\mathsf{Tail}(p))$, then the claim holds immediately by construction of $\mathfrak{A}$.
- Assume $\varphi(\mathbf{a}) = \neg R\mathbf{a} \in \Delta(\mathsf{Tail}(p))$, but $[\mathbf{a}, p]_\approx \in R^{\mathfrak{A}}$. Then, by the definition of $\mathfrak{A}$, there must be a path p' and constants $\mathbf{c}$ such that $(\mathbf{a}, p) \approx (\mathbf{c}, p')$ and $R\mathbf{c} \in \Delta(\mathsf{Tail}(p'))$. From Claim 3 we have that $(\mathbf{a}, p) \approx (\mathbf{c}, p')$ implies $R\mathbf{a} \in \Delta(\mathsf{Tail}(p))$ and hence $\mathbf{T}$ would contain the clash $\{R\mathbf{a}, \neg R\mathbf{a}\} \subseteq \Delta(\mathsf{Tail}(p))$.
- Assume $\varphi(\mathbf{a}) = a \neq b \in \Delta(\mathsf{Tail}(p))$ but $[a, p]_\approx = [b, p]_\approx$. From Claim 1 we get that this implies $a = b$ and hence $\mathbf{T}$ contains the clash $a \neq a \in \Delta(\mathsf{Tail}(p))$.
- For Boolean combinations the claim is immediate due to the $\mathsf{R}\wedge$- and $\mathsf{R}\vee$-rule.
- Let $\varphi(\mathbf{a}) = (\forall \mathbf{yz}.\alpha(\mathbf{a}, \mathbf{y}, \mathbf{z}))\chi(\mathbf{a}, \mathbf{y}) \in \Delta(\mathsf{Tail}(p))$ and $\mathbf{b}, \mathbf{p}, \mathbf{c}, \mathbf{q}$ such that

  $$\mathfrak{A} \models \alpha([\mathbf{a}, p]_\approx, [\mathbf{b}, \mathbf{p}]_\approx, [\mathbf{c}, \mathbf{q}]_\approx). \tag{1}$$

  Every $y_i \in \mathbf{y}$ coexists with every other variable $y_j \in \mathbf{y}$ in at least one conjunct $\beta^{(y_i, y_j)} \in \alpha(\mathbf{a}, \mathbf{y}, \mathbf{z})$ and with every element $a_\ell \in \mathbf{a}$ in at least one conjunct $\beta^{(y_i, a_\ell)} \in \alpha(\mathbf{a}, \mathbf{y}, \mathbf{z})$. Since (1), for every two elemts $[b_i, p_i]_\approx, [b_j, p_j]_\approx \in [\mathbf{b}, \mathbf{p}]_\approx$ there is a path $q^{(i,j)}$ and constants $d^{(i,j)}, e^{(i,j)}$ such that $(b_i, p_i) \approx (c^{(i,j)}, q^{(i,j)})$ and $(b_j, p_j) \approx (d^{(i,j)}, q^{(i,j)})$. Similarly, for every element $[b_i, p_i]_\approx \in [\mathbf{b}, \mathbf{p}]_\approx$ and every element (a_ℓ, p) there exists a path $r^{(i,\ell)}$ and constants $f^{(i,j)}, g^{(i,j)}$ such that $(b_i, p_i) \approx (f^{(i,\ell)}, r^{(i,\ell)})$ and $(a_\ell, p) \approx (g^{(i,\ell)}, r^{(i,\ell)})$. Equivalence classes of $\approx$ induce subtrees of $\mathsf{Paths}(\mathbf{T})$. Every subtree induced by $[b_i, p_i]_\approx$ overlaps with the subtree induced by $[b_j, p_j]_\approx$ at $q^{(i,j)}$ and with the subtree induced by $[a_\ell, p]_\approx$ at $r^{(i,\ell)}$. It is a well-known result in graph theory that this implies the existence of a single path s which lies on all of the induced subtrees. Thus, there must be tuples $\mathbf{a}'$, $\mathbf{b}'$ such that

  $$(\mathbf{a}, p) \approx (\mathbf{a}', s) \text{ and } (\mathbf{b}, \mathbf{p}) \approx (\mathbf{b}', s). \tag{2}$$

  For every $\beta(\mathbf{x}, \mathbf{y}, \mathbf{z}) \in \alpha(\mathbf{x}, \mathbf{y}, \mathbf{z})$, Claim 3 implies $\beta(\mathbf{a}', \mathbf{b}', * \cdots *) \in^* \Delta(\mathsf{Tail}(s))$ as follows: from (1,2) we get $\mathfrak{A} \models \beta([\mathbf{a}', s]_\approx, [\mathbf{b}', s]_\approx, [\mathbf{c}, \mathbf{q}]_\approx)$. Since β is an atom, this implies the existence of a path t and tuples $\mathbf{a}'', \mathbf{b}'', \mathbf{c}'$ with

  $$(\mathbf{a}', s) \approx (\mathbf{a}'', t) \text{ and } (\mathbf{b}', s) \approx (\mathbf{b}'', t) \text{ and } (\mathbf{c}, \mathbf{q}) \approx (\mathbf{c}', t) \text{ and } \beta(\mathbf{a}'', \mathbf{b}'', \mathbf{c}') \in \Delta(\mathsf{Tail}(t)) \tag{3}$$

  Since $\beta(\mathbf{a}'', \mathbf{b}'', * \cdots *) \geq^* \beta(\mathbf{a}'', \mathbf{b}'', \mathbf{c}')$, Claim 3 yields $\beta(\mathbf{a}', \mathbf{b}', * \cdots *) \in^* \Delta(\mathsf{Tail}(s))$.

  Since this is true for every atom β and, also due to Claim 3$(\forall \mathbf{yz}.\alpha(\mathbf{a}', \mathbf{y}, \mathbf{z}))\chi(\mathbf{a}', \mathbf{y}) \in \Delta(\mathsf{Tail}(s))$, the completeness of $\mathbf{T}$ yields $\chi(\mathbf{a}', \mathbf{b}') \in \Delta(\mathsf{Tail}(s))$. By induction, this implies $\mathfrak{A} \models \chi([\mathbf{a}', s]_\approx, [\mathbf{b}', s]_\approx)$. Together with (2) this implies $\mathfrak{A} \models \chi([\mathbf{a}, p]_\approx, [\mathbf{b}, \mathbf{p}]_\approx)$ and hence $\mathfrak{A} \models \varphi([\mathbf{a}, p]_\approx)$.
- If $\varphi(\mathbf{a}) = (\exists \mathbf{yz}.\alpha(\mathbf{a}, \mathbf{y}, \mathbf{z}))\chi(\mathbf{a}, \mathbf{y}) \in \Delta(\mathsf{Tail}(p))$, there are two possibilities.
  - there are $\mathbf{b}, \mathbf{c} \subseteq \mathbf{C}(\mathsf{Tail}(p))$ such that $\{\alpha(\mathbf{a}, \mathbf{b}, \mathbf{c})\} \subseteq \Delta(\mathsf{Tail}(p))$ and $\chi(\mathbf{a}, \mathbf{b}) \in \Delta(\mathbf{a}, \mathbf{b})$. Then, by induction, we have

    $$\mathfrak{A} \models \{\alpha([\mathbf{a}, p]_\approx, [\mathbf{b}, p]_\approx, [\mathbf{c}, p]_\approx), \chi([\mathbf{a}, p]_\approx, [\mathbf{b}, p]_\approx)\}$$

    and hence $\mathfrak{A} \models \varphi([\mathbf{a}, p]_\approx)$.

– there are no such $\mathbf{b}, \mathbf{c} \subseteq \mathbf{C}(\mathsf{Tail}(p))$, then there is a successor w of $\mathsf{Tail}(p)$ and a sequence of constants $\mathbf{b}, \mathbf{c} \subseteq \mathbf{C}(w)$ with $\{\alpha(\mathbf{a}, \mathbf{b}, \mathbf{c}), \chi(\mathbf{a}, \mathbf{b})\} \subseteq \Delta(w)$. The node w can be blocked or not.

  * If w is not blocked, then $p' = [p, \frac{w}{w}] \in \mathsf{Paths}(\mathbf{T})$ and, by induction,

$$\mathfrak{A} \models \{\alpha([\mathbf{a}, p']_\approx, [\mathbf{b}, p']_\approx, [\mathbf{c}, p']_\approx), \chi([\mathbf{a}, p']_\approx, [\mathbf{b}, p']_\approx)\}$$

    From the definition of $\approx$ we have, $(\mathbf{a}, p') \approx (\mathbf{a}, p)$ and hence $\mathfrak{A} \models \varphi([\mathbf{a}, p]_\approx)$.

  * If w is blocked by a node u (with function π) then $p' = [p, \frac{u}{w}] \in \mathsf{Paths}(\mathbf{T})$. From the blocking condition, we have that u is unblocked and $\pi\{\alpha(\mathbf{a}, \mathbf{b}, \mathbf{c}), \chi(\mathbf{a}, \mathbf{b})\}) \subseteq \Delta(u)$. Hence, by induction,

$$\mathfrak{A} \models \{\alpha([\pi(\mathbf{a}), p']_\approx, [\pi(\mathbf{b}), p']_\approx, [\pi(\mathbf{c}), p']_\approx), \chi([\pi(\mathbf{a}), p']_\approx, [\pi(\mathbf{b}), p']_\approx)\}.$$

    By the definition of $\approx$ we have that $(\mathbf{a}, p) \approx (\pi(\mathbf{a}), p')$ and hence, $\mathfrak{A} \models \varphi([\mathbf{a}, p]_\approx)$.

As a special instance of Claim 4 we get that $\mathfrak{A} \models \psi$. Due to Lemma 3.7, for every node $v \in \mathbf{V}$, $|\mathbf{C}(v)| \leq \text{width}(\psi)$ and hence $\mathfrak{A}$ has width at most $\text{width}(\psi) - 1$. Note, that we have also given an alternative proof for Lemma 3.15 ∎

4 Conclusion

We have developed a tableau algorithm for CGF, which we hope can serve as basis for an efficient implementation of a decision procedure for CGF. This hope is justified by the fact that some of the most efficient implementations of modal or description logic reasoners are based on tableaux calculi similar to the one for CGF presented in this paper. As a corollary from the constructions used to prove the correctness of the tableaux algorithm, we show that CGF, and hence LGF and GF, have the finite model property. We also give a new proof of the fact that every satisfiable GF/LGF/CGF sentence of width k has a model of tree width at most $k - 1$.

References

[1] H. Andréka, J. van Benthem, and I. Németi. Modal languages and bounded fragments of predicate logic. *Journal of Philosophical Logic*, 27:217–274, 1998.

[2] F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. *Information and Computation*, 134(1):1–58, 1997.

[3] H. Ganzinger and H. de Nivelle. A superposition decision procedure for the guarded fragment with equality. In *Proc. 14th IEEE Symp. on Logic in Computer Science*, 1999.

[4] E. Grädel. On the restraining power of guards. *Journal of Symbolic Logic*. To appear.

[5] E. Grädel. Decision procedures for guarded logics. In *Automated Deduction - CADE16. Proceedings of 16th International Conference on Automated Deduction, Trento, 1999*, volume 1632 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 1999.

[6] E. Grädel and I. Walukiewicz. Guarded fixed point logic. In *Proc. 14th IEEE Symp. on Logic in Computer Science*, 1999.

[7] J. Y. Halpern and Y. Moses. A guide to completeness and complexity for model logics of knowledge and belief. *Artificial Intelligence*, 54(3):319–379, April 1992.

[8] C. Hirsch and S. Tobies. A tableaux algorithm for the clique guarded fragment. LTCS-Report 00-03, LuFG Theoretical Computer Science, RWTH Aachen, Germany, 2000. See http://www-lti.informatik.rwth-aachen.de/Forschung/Reports.html.

[9] I. Horrocks, P. F. Patel-Schneider, and R. Sebastiani. An analysis of empirical testing for modal decision procedures. *Logic Journal of the IGPL*, 8(3):293–323, 2000.

[10] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In H. Ganzinger, D. McAllester, and A .Voronkov, editors, *Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99)*, number 1705 in Lecture Notes in Artificial Intelligence, pages 161–180. Springer-Verlag, September 1999.

[11] R. Ladner. The computational complexity of provability in systems of propositional modal logic. *SIAM Journal on Computing*, 6:467–480, 1977.

[12] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48:1–26, 1991.

[13] J. van Benthem. Dynamic bits and pieces. ILLC research report, University of Amsterdam, 1997.

[14] C. Weidenbach. SPASS—version 0.49. *J. of Automated Reasoning*, 18(2):247–252, April 1997.