A Tableau Algorithm for the Clique Guarded Fragment

Extended Abstract

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

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

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 GF must be of the form

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

where α is a positive atomic formula, the *guard*, that contains all free variables of φ. 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) [16] 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 was open until recently. In [10] Hodkinson shows that LGF and a further variant, the *packed fragment* (PF) [14], have the finite model property. Indeed the packed fragment turns out to be a syntactic variant of the clique guarded fragment.

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 [13, 7, 2, 12]. Some of the fastest implementations of modal satisfiability procedures are based on tableau calculi [11]. 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 in 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 GF has the finite model property. While this result is not new, we feel that our proof is more

[1]There are resolution based decision procedures for GF and LGF [3] that are readily implemented using the saturation theorem prover SPASS [17]. It is unclear if this approach can be extended to CGF.

elementary as it does not require some more advanced model-theoretic constructions along [8]. We conjecture that the same method can be extended to include CGF and LGF. Also, we obtain an alternative proof for the fact that CGF has a generalised tree model property, i.e., every satisfiable CGF formula of width k has a model of tree width at most $k-1$ [5].

Due to the limited space for this extended abstract, we refer to [9] for most of the proofs.

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*

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

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 τ-structure and ρ 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*

$$\begin{aligned}\mathfrak{A}, \rho \models \forall y.\varphi(\mathbf{x}, y) \textit{ iff for all } a \in A \textit{ such that}\\ \rho(\mathbf{x}) \cup \{a\} \textit{ forms a clique in } G(\mathfrak{A})\\ \textit{it is the case that } \mathfrak{A}, \rho[x \mapsto a] \models \varphi \ ,\end{aligned}$$

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 a* clique-formula *for a set $\mathbf{x} \subseteq \mathrm{free}(\alpha)$ 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 $\mathrm{free}(\alpha) \setminus \mathbf{x}$ 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.*
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 $\mathrm{free}(\varphi) \subseteq \mathbf{x} \cup \mathbf{y}$,*

   $$\begin{aligned}&\textit{then} && \exists \mathbf{yz}.(\alpha(\mathbf{x},\mathbf{y},\mathbf{z}) \wedge \varphi(\mathbf{x},\mathbf{y}))\\ &\textit{and} && \forall \mathbf{yz}.(\alpha(\mathbf{x},\mathbf{y},\mathbf{z}) \rightarrow \varphi(\mathbf{x},\mathbf{y}))\end{aligned}$$

   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 formula manipulation.

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

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

Some further transformations can be used to show that a packed quantification can be simulated by a syntactic clique guarded formula yielding the following result.

Lemma 2.4. PF *and* CGF *are equally expressive.*

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

Lemma 2.5. *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 $clique(\mathbf{x}, y, \mathbf{z})$ of clique-formulas for $\mathbf{x}, y$ such that $\mathbf{a}, b$ form a clique in $G(\mathfrak{A})$ iff $\mathfrak{A} \models \exists \mathbf{z}.clique(\mathbf{a}, b, \mathbf{z})$. 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. If we fix a finite relational vocabulary, this transformation is polynomial in the width of the formula.

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 2.6 (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*

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

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

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

3 A Tableau Algorithm for CGF

For various modal and description logics, decidability can be shown by means of tableau 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 [13, 15, 7, 12]. 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 [11].

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., 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 tableau 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 $V_a = \{v \in V : a \in f(v)\}$ 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 $\mathfrak{A}$.*

A logic $\mathcal{L}$ has the generalised tree model property *if there exists a computable function t, assigning to every sentence $\psi \in \mathcal{L}$ a natural number $t(\psi)$ such that, if ψ is satisfiable, then ψ has a model of tree width at most $t(\psi)$.*

Fact 3.2 (Tree Model Property for CGF**).** *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 μCGF, that is CGF extended by a least fixed point operator.

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*

$Rt_1 \ldots t_n \geq^* Rt'_1 \ldots t'_n$ iff for all $i \in \{1 \ldots n\}$ either $t_i = *$ 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 $\beta = Rt_1 \ldots t_n$, we define

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

We use the notation $\mathbf{a}^*$ to indicate that the tuple $\mathbf{a}^*$ may contain $*$'s. Obviously, $\geq^*$ is transitive and reflexive, and $\beta|^*_C \geq^* \beta$ for all atoms β 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 in this replacement. For an atom β, we define

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

It is easy to see that

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

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 (Compl. 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[2] *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.*

[2]The definition of blocking is recursive. This does not cause any problems because the status of a node v only depends on its label and the status of nodes w with $\mathbf{N}(w) < \mathbf{N}(v)$. The recursion terminates at the root node, where the $\mathbf{N}$-value is minimal.

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 β and a tuple of constants $\mathbf{a} \subseteq \mathbf{C}(v)$ such that $\{\beta(\mathbf{a}), \neg\beta(\mathbf{a})\} \subseteq \Delta(v)$.*

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 successively applied until either a clash occurs, producing output "ψ `unsatisfiable`*", or the tree is complete, in which case "ψ* `satisfiable`*" is output.*

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 $\varphi \in \text{cl}(\psi, \mathbf{C}(v))$ 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 3.7)

R∧ : if $\varphi \wedge \vartheta \in \Delta(v)$ and $\{\varphi, \vartheta\} \not\subseteq \Delta(v)$
then $\Delta(v) := \Delta(v) \cup \{\varphi, \vartheta\}$

R∨ : if $\varphi \vee \vartheta \in \Delta(v)$ and $\{\varphi, \vartheta\} \cap \Delta(v) = \emptyset$
then $\Delta(v) := \Delta(v) \cup \{\chi\}$ for $\chi \in \{\varphi, \vartheta\}$ (chosen non-deterministically)

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∀ : 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∃ : 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↕ : if $\beta(\mathbf{a}^*) \in \Delta(v), \beta$ atomic, w is a neighbour of v with $\mathbf{a}^* \cap \mathbf{C}(w) \neq \emptyset$, and $\beta(\mathbf{a}^*)|^*_{\mathbf{C}(w)} \notin \Delta(w)$
then $\Delta(w) := \Delta(w) \cup \{\beta(\mathbf{a})|^*_{\mathbf{C}(w)}\}$

R↕∀ : 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

2. If ψ is satisfiable, then the rules can be applied to generate a tableau for ψ. (*Completeness*, Lemma 3.8)

3. If the algorithm constructs a tableau for ψ, then ψ is satisfiable (*Soundness*). We give two alternative proofs for soundness. One for GF (Lemma 3.14) and one for the full CGF (Lemma 3.16). ■

We give two proofs for the soundness of the tableau algorithm because as corollaries of the constructions used in the different proofs, we obtain alternative proofs for the finite model property of GF (Corollary 3.15) and of the generalised tree model property of CGF (Corollary 3.17).

3.1 Termination

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

Lemma 3.6. *Let $\psi \in$ CGF be a closed formula in NNF with $|\psi| = n$, 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.*

Lemma 3.7 (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

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

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 Lemma 3.6, 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. The R$\exists$-rule is 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 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 R$\exists$-rule to a node v. As each such application decreases $\|v\|$ we have a contradiction. ∎

3.2 Completeness

Lemma 3.8. *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 initial 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 below 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 such that $(*)$ holds and a rule is applicable to $\mathbf{T}$, then it can be applied in a way that maintains $(*)$.

- For the R$\wedge$- and the R$\vee$-rule this is obvious.

- 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 $\mathbf{c}^* \subseteq \mathbf{C}(v)^*$ 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 R$\updownarrow$-rule is applicable to a node $v \in \mathbf{V}$ with $\beta(\mathbf{a}^*) \in \Delta(v)$ and a neighbour w with $\mathbf{a}^* \cap \mathbf{C}(w) \neq \emptyset$, then it adds $\beta(\mathbf{a}^*)|^*_{\mathbf{C}(w)}$ to $\Delta(w)$. From $(*)$ we get that $\mathfrak{A} \models \beta(g(\mathbf{a}^*))$, and since $\beta(\mathbf{a}^*)|^*_{\mathbf{C}(w)} \geq^* \beta(\mathbf{a}^*)$, this implies $\mathfrak{A} \models \beta(g(\mathbf{a}^*))|^*_{\mathbf{C}(w)}$. Hence, adding $\beta(\mathbf{a}^*)|^*_{\mathbf{C}(w)}$ to $\Delta(w)$ does not violate $(*)$.

- If the 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 $\{\beta(\mathbf{a}), \neg\beta(\mathbf{a})\} \subseteq \Delta(v)$. From $(*)$ it would follow that $\mathfrak{A} \models \{\beta(g(\mathbf{a})), \neg\beta(g(\mathbf{a}))\}$, also a contradiction.

These claims yield Lemma 3.8 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 Lemma 3.7, 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 3.8 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 R$\vee$-rule. While the latter choice is *don't-know* non-deterministic, 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 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 strongly depend on a sophisticated strategy.

3.3 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 an, indeed finite, model from a tableau. The following applies only to GF; the generalization to CGF is yet work in progress and can at this point only be conjectured. An alternative correctness proof applicable to the CGF case is given in a later section in Lemma 3.16, providing a further proof for the generalised tree model property but omitting the finite model property.

In the following, let $\psi \in \mathrm{GF}[\tau]$ 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 node u blocking v and we define $\mathbf{B}$ as set of all such pairs (u, v). 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 $1 \leq i < k$ 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 $a \sim b$.

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 problematic situations in a tableau that make this construction impossible, so called *dormant clashes*.

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 successively 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. 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.

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.

Lemma 3.13. *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 dormant clashes.*

The construction used to show this Lemma is omitted in this paper and can be found in [9].

Lemma 3.14. *Let $\psi \in \mathrm{GF}[\tau]$ and let $\mathbf{T}$ be a tableau for ψ. Then ψ is (finitely) satisfiable.*

Proof: According to Lemma 3.13 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 $\mathsf{R}\updownarrow$ 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 $\mathsf{R}{=}$-rule makes sure, that distinct constants occuring at a common node have distinct $\sim$-classes. For inequality statements, assume $a \neq b \in \Delta(v)$, but $a \sim b$. Then we can find an (a,b)-path containing 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 $\varphi(\mathbf{a}) = (\exists \mathbf{y}.\beta(\mathbf{a},\mathbf{y}))\eta(\mathbf{a},\mathbf{y})$. If, for some $\mathbf{b} \in \mathbf{C}(v)$, $\beta(\mathbf{a},\mathbf{b}), \eta(\mathbf{a},\mathbf{b}) \in \Delta(v)$, we note that $\mathfrak{A} \models \eta(\tilde{\mathbf{a}},\tilde{\mathbf{b}})$ and $\mathfrak{A} \models \beta(\tilde{\mathbf{a}},\tilde{\mathbf{b}})$ by induction hypothesis for β and η.

  If there is no $\mathbf{b} \in \mathbf{C}(v)$ with $\beta(\mathbf{a},\mathbf{b}), \eta(\mathbf{a},\mathbf{b}) \in \Delta(v)$, then application of the R$\exists$-Rule yields a successor node w of v with constants $\mathbf{b} \in \mathbf{C}(w)$ such that $\beta(\mathbf{a},\mathbf{b}), \eta(\mathbf{a},\mathbf{b}) \in \Delta(w)$. If w is not blocked, the claim again follows by induction hypothesis for β 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 $\beta(\pi(\mathbf{a}),\pi(\mathbf{b}))$ 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 $\mathfrak{A} \models \beta(\tilde{\mathbf{a}},\tilde{\mathbf{b}}) \wedge \eta(\tilde{\mathbf{a}},\tilde{\mathbf{b}})$, and hence $\varphi(\tilde{\mathbf{a}})$ holds in $\mathfrak{A}$.

- Finally let $\varphi(\mathbf{a}) = (\forall \mathbf{y}.\beta(\mathbf{a},\mathbf{y}))\eta(\mathbf{a},\mathbf{y})$. Assume that there is a tuple $\mathbf{b}$ such that $\mathfrak{A} \models \beta(\tilde{\mathbf{a}},\tilde{\mathbf{b}})$. Then 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, $\beta(\mathbf{a}',\mathbf{b}') \in \Delta(w)$ and $\varphi(\mathbf{a}') \in \Delta(w)$. Hence, the R$\forall$-rule is applicable for $\varphi(\mathbf{a}')$ at w and must have been applied because $\mathbf{T}$ is complete. This gives us $\eta(\mathbf{a}',\mathbf{b}') \in \Delta(v)$, which, by induction, yields $\mathfrak{A} \models \eta(\mathbf{a}',\mathbf{b}')$ and hence $\mathfrak{A} \models \eta(\mathbf{a},\mathbf{b})$. ∎

Unfortunately, for CGF it is not sufficient to get rid of the dormant clashes, but one has to deal also with *evil cliques*. These are cliques in the Gaifman graph of the constructed model that are not explicitly represented in the tableau and are only caused by folding back from blocked to blocking nodes. These cliques might interfere with universally quantified subformulas of the input formula ψ in a way that leads to the obtained structure not being a model for ψ even though it was constructed from a tableau for ψ. Although we believe that it is possible to construct a finite model from a tableau using a similar construction to the one used to establish the result for GF, the proof remains part of future work.

Hence, we only obtain the finite model property of the GF as a corollary from this.

Corollary 3.15. GF *has the finite model property.*

Proof: If $\psi \in \mathrm{GF}[\tau]$ is satisfiable, then, by Lemma 3.8, it has a finite tableau. As shown in the proof of Lemma 3.14, such a tableau induces a finite model for ψ ∎

3.4 Correctness, alternative version

Since the construction used in the proof of Lemma 3.14 so far could only be shown to be valid for GF, we still need to show soundness of the algorithm for CGF. In this section, we will give an alternative proof for the soundness of the algorithm, which is valid also for CGF. From the construction employed in the proof we obtain and alternative proof of Fact 3.2 as a corollary.

Lemma 3.16. *Let* $\psi \in \mathrm{CGF}[\tau]$ *with* $k = \mathrm{width}(\psi)$ *and let* $\mathbf{T}$ *be a tableau for* ψ *generated by the tableau algorithm. Then* ψ *is satisfiable and has a model of tree width at most* $k-1$.

Due to limited space, we refer to [9] for the full proof and give only a sketch here.

Proof: Let $\mathbf{T} = (\mathbf{V},\mathbf{E},\mathbf{C},\Delta,\mathbf{N})$ a tableau for ψ. Using an unraveling construction, we will construct a model for ψ of width at most $k-1$ from $\mathbf{T}$. We define

$$\mathbf{V}_u = \{v \in \mathbf{V} \ : \ v \text{ is not indirectly blocked } \}$$

and $\mathsf{Paths}(\mathbf{T}) \subseteq \mathbf{V}_u^+$ (the set of non-empty strings over $\mathbf{V}_u$) inductively defined by[3]

[3]This complicated form of unraveling, where we record both blocked and 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.

- $[\frac{v_0}{v_0}] \in \mathsf{Paths}(\mathbf{T})$ 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

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

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 $a = b$.

Claim 1 is shown by contradiction. For $a \neq b$ with $(a,p) \approx (b,p)$ we take a shortest $\sim$-path from (a,p) to (b,p), which must be of the form $(a,p) \sim (c,q) \sim (b,p)$. This implies that $\mathsf{Tail}'(q)$ must be blocked by $\mathsf{Tail}(q)$ (because otherwise $a = b = c$ would hold), which implies that the function π verifying this blocking cannot be injective. This is impossible.

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 $(a,p) \sim (b,p')$.

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 $\varphi(\mathbf{b}) \in \Delta(\mathsf{Tail}(q))$.

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 $Rc_1 \dots c_m \in \Delta(\mathsf{Tail}(p))$.

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

As a special instance of Claim 4 we get that $\mathfrak{A} \models \psi$. Due to Lemma 3.6, for every node $v \in \mathbf{V}$, $|\mathbf{C}(v)| \leq \text{width}(\psi)$ and hence $\mathfrak{A}$ has tree width at most $\text{width}(\psi) - 1$. ∎

Corollary 3.17. CGF, *and hence also* PF/LGF/GF *have the generalised tree model property.*

Proof: Let $\psi \in \text{CGF}[\tau]$ be satisfiable. Then, from Lemma 3.8 we get that there is a tableau $\mathbf{T}$ for ψ. By Lemma 3.16, $\mathbf{T}$ induces a model for ψ of tree width at most $\text{width}(\psi) - 1$. Note that we have never relied on Fact 3.2 to obtain any of the results in this paper and hence have indeed given an alternative proof for the generalised tree model property of CGF. ∎

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 tableau calculi similar to the one for CGF presented in this paper. As a corollary from the constructions used to prove the correctness of the tableau algorithm, we give an, in our opinion, simpler proof for the finite modal property of GF. An extension of our approach to CGF is part of future work. We also give a new proof of the fact that GF/LGF/CGF have the generalised tree model property.

Acknowledgements

We would like to thank Andrei Voronkov for helpful discussions. The second author is supported by the DFG, Project No. GR 1324/3–1.

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*, pages 295–303, 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 H. Ganzinger, editor,*Proceedings of 16th International Conference on Automated Deduction*, volume 1632 of *Lecture Notes in Artificial Intelligence*, pages 31–51. Springer-Verlag, 1999.

[6] E. Grädel and I. Walukiewicz. Guarded fixed point logic. In *Proc. 14th IEEE Symp. on Logic in Computer Science*, pages 45–54, 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] B. Herwig. Extending partial isomorphisms on finite structures. *Combinatorica*, 15:365–371, 1995.

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

[10] I. Hodkinson Loosely guarded fragment of first-order logic has the finite model property. Submitted, 2000. Online available from `http://www.doc.ic.ac.uk/~imh/index.html`.

[11] 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.

[12] 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.

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

[14] M. Marx. Tolerance Logic. In *Journal of Logic, Language and Computation.* To appear. Online availble from `http://turing.wins.uva.nl/~marx/papers.html`.

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

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

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