1

A Tableau Algorithm for the Clique Guarded Fragment

Colin Hirsch and Stephan Tobies

ABSTRACT. We describe a "modal style" tableau algorithm that decides satisfiability for the clique guarded fragment. As a corollary of constructions used to prove the correctness of the algorithm, we obtain a new proof for the generalised tree model property of the clique guarded fragment.

1 Introduction

The Guarded Fragment of first-order logic, introduced by Andréka, van Benthem, and Németi (1998), 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 (Andréka et al. 1998, Grädel 1999b).

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 an 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

Advances in Modal Logic, Volume 3
F. Wolter, H. Wansing, M. de Rijke, and M. Zakharyaschev, eds.

the above sense. Most importantly, one can obtain the *loosely guarded fragment* (LGF) (van Benthem 1997) and the *clique guarded fragment* (CGF) (Grädel 1999a), for which decidability, invariance under clique guarded bisimulation, and some other properties have been shown in (Grädel 1999a). Suffice to say that both LGF and CGF properly extend GF, and, with respect to sentences, CGF properly extends LGF, as shown in (Grädel 1999b).

GF, LGF, and CGF are decidable and known to be 2-ExpTime complete, which is shown by Grädel (1999a, 1999b) using game and automata-based approaches. While these approaches yield (worst-case) optimal 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. By contrast many decidability results for modal or description logics are based on tableau algorithms (see Ladner 1977, Halpern and Moses 1992, Donini et al. 1997, or Horrocks et al. 1999 for examples) and some of the fastest implementations of modal satisfiability procedures are based on tableau calculi (Horrocks et al. 2000). 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 from tableau algorithms for modal logics in order 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]

Recall the conjecture by Vardi that the tree model property is the main reason for the decidability of many modal style logics (Vardi 1997). As pointed out in (Grädel 1999b), the generalised tree model property explains the similarly robust decidability of guarded logics, and can be seen as a strong indication that guarded logics are a generalisation of modal logics that retain the essence of modal logics. This becomes even more evident when regarding the respective fixed-point extensions (Grädel 1999a). The generalised tree model property of CGF is also essential for our tableau algorithm. Indeed, as a corollary of the constructions used to show the soundness of our algorithm, we obtain an alternative proof for the fact that CGF has the generalised tree model property.

[1] There are resolution based decision procedures for GF and LGF (Ganzinger and de Nivelle 1999) that are readily implemented using the saturation theorem prover SPASS (Weidenbach 1997). While CGF-SAT can be easily reduced to LGF-SAT, we believe that dedicated CGF algorithms are more efficient, c.f. before Definition 3.3.

2 Preliminaries

For the definitions of GF and LGF we refer the reader to (Grädel 1999a). 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

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

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 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

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

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 $\mathbf{x} \subseteq \text{free}(\alpha)$ if α is a (possibly empty if $\mathbf{x}$ contains only one variable) conjunction of atoms (excluding equality statements) such that each two distinct elements from $\mathbf{x}$ coexist in at least one atom, each atom contains at least an element from $\mathbf{x}$, and each element from $\text{free}(\alpha) \setminus \mathbf{x}$ occurs exactly once in α. 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 $\text{free}(\varphi) \subseteq \mathbf{x} \cup \mathbf{y}$,

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

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. A formula of the form $\forall \mathbf{yz}.(\alpha(\mathbf{x}, \mathbf{y}, \mathbf{z}) \rightarrow \varphi(\mathbf{x}, \mathbf{y}))$ is called *universally quantified*.

The following Lemma can be shown by elementary formula manipulations that exploit that every $z \in \mathbf{z}$ occurs exactly once in α.

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

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

The use of the name CGF for both 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. If we fix a finite relational vocabulary, this transformation is polynomial in the number of variables of the formula, or, more precisely, the maximal number of free variables of all subformulas. $\dashv$

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

Definition 2.5 (NNF, Closure, Width) In the following, all formulas are assumed 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 sentence $\psi \in \mathrm{CGF}$ 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 $\psi \in \mathrm{CGF}$ 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 (Ladner 1977, Halpern and Moses 1992, Horrocks et al. 1999). 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 of 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 or description logics satisfiability algorithms use tableau calculi (Horrocks et al. 2000).

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. All these notions will also be encountered in our tableau algorithm for CGF, an indication for the "modal nature" of CGF, as it is amenable to the same

techniques used successfully for modal logics.

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 $k + 1$. 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 Prop. 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 (Grädel 1999a), 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 the 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 the auxiliary constant symbol $*$ as a placeholder for unspecified elements in atoms. The intention is to keep the number of constants at each node as small as possible. The $*$ will be used for the extra elements occuring in clique formulas that are not part of the clique itself.

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 $\beta = Rt_1 \ldots t_n$, we define

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

We use the notation $\mathbf{a}^*$ to indicate that the tuple $\mathbf{a}^*$ may contain $*$'s. Obviously, $\geq^*$ is transitive and reflexive, and $\beta \circledast C \geq^* \beta$ 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

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

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.

We further write $\Phi|_C$ to denote the subset of Φ containing all formulas that only use constants in C.

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 $cl(\psi, \mathbf{C}(v)^*)$ where all formulas $\beta(\mathbf{x}, *, \ldots, *) \in \Delta(v)$ using $*$ are atoms (excluding equality statements), 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) shortest 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.

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 $\mathbf{C}(v_0) = \{a_0\}$ for an arbitrary constant a_0. The rules from Figure 1 are applied until either a clash occurs, producing output "ψ `unsatisfiable`", or the tree is complete, in which case "ψ `satisfiable`" is output.

The set $\mathbf{C}(v_0)$ is initialised with a non-empty set of constants to make sure that empty structures are excluded.

While our notion of tableaux has many similarities to the tableaux appearing in (Grädel and Walukiewicz 1999), there are two important differences that make the version used 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 (Grädel and Walukiewicz 1999), in general, can be infinite.

Also, in (Grädel and Walukiewicz 1999) 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.

[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.

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 $\chi \in \{\varphi, \vartheta\}$ (chosen non-deterministically)
R= :	if	$a = b \in \Delta(v), a \neq b$
	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	$\beta(\mathbf{a}^*) \in \Delta(v), \beta$ atomic, not an equality, w is a neighbour of v with $\mathbf{a}^* \cap \mathbf{C}(w) \neq \emptyset$, and $\beta(\mathbf{a}^*) \Rsh \mathbf{C}(w) \notin \Delta(w)$
	then	$\Delta(w) := \Delta(w) \cup \{\beta(\mathbf{a}^*) \Rsh \mathbf{C}(w)\}$
R$\updownarrow\forall$:	if	$\varphi(\mathbf{a}) \in \Delta(v), \varphi(\mathbf{a})$ is univ. quantified, and w is a neighbour of v 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

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)
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*, Lemma 3.11). $\dashv$

3.1 Termination

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

Lemma 3.6 *Let $\psi \in$ CGF be a sentence in NNF with $|\psi| = n$,* width(ψ) $= m$*, 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: Nodes are only generated when intialising the tree (with a singleq constant) and 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,

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

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 $\mathbf{C}(v)^*$.

Let $v_1, \ldots, v_\ell$ be $\ell > 2^{n \times (m+1)^m}$ distinct nodes. For every v_i, we will construct an injective mapping $\pi_i : \mathbf{C}(v_i) \to \{1, \ldots m\}$ 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 $\nu_i : \mathbf{C}(u_i) \to \{1, \ldots, m\}$ for every $i \in \{1, \ldots, k\}$ as follows. For ν_1 we pick an arbitrary injective function from $\mathbf{C}(u_1)$ to $\{1, \ldots, m\}$. For a node u_i let u_j be the predecessor of u_i in $\mathbf{T}$ and ν_j the corresponding function, which has already been defined because u_j has a smaller distance to u_1 than u_i. 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 shortest 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 $cl(\psi, \{1, \ldots, m, *\})$. 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., $\mathbf{N}(v_i) < \mathbf{N}(v_j)$. This implies 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. Since $\pi_i(\Delta(v_i)) = \pi_j(\Delta(v_j))$, $\pi(\Delta(v_j)) = \Delta(v_i)|_{\pi(\mathbf{C}(v_j))}$ holds. $\dashv$

Lemma 3.7 (Termination) *Let $\psi \in$ CGF be a sentence in NNF. Any sequence of rule applications 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

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

The lexicographic order $\prec$ on $\mathbb{N}^3$ 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 new successors, one at a time. Since $\prec$ is well-founded, there can only be a finite number of applications of rules to every node in $\mathbf{T}$ and hence a finite number of successors and an infinite sequence of rule applications would generate a tree of infinite depth.

Yet, as a corollary of Lemma 3.6, we have that the depth of $\mathbf{T}$ is bounded by $2^{n\times(m+1)^m}$. For assume that there is a path of length $> 2^{n\times(m+1)^m}$ in $\mathbf{T}$ with deepest node v. By the time v has been created (by an application of the $\mathsf{R}\exists$-rule to its predecessor u), the path from the root of $\mathbf{T}$ to u contains at least $2^{n\times(m+1)^m}$ nodes, and hence a blocked node. This implies that u is blocked too, and the $\mathsf{R}\exists$-rule cannot be applied to create v. $\dashv$

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 $\mathsf{R}\vee$-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 $\mathbf{a}^*$. The constant $*$ is not mapped to an element of A by g. We deal with this as described just after 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 3.9 *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 $v \in \mathbf{V}$ with $a = b \in \Delta(v)$, then, since $\mathfrak{A} \models g(a) = g(b)$, $g(a) = g(b)$ must hold. 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 $v \in \mathbf{V}$ with $(\forall \mathbf{yz}.\alpha(\mathbf{a}, \mathbf{y}, \mathbf{z}))\varphi(\mathbf{a}, \mathbf{y}) \in \Delta(v)$ and $\mathbf{b} \subseteq \mathbf{C}(v)$ with $\beta(\mathbf{a}, \mathbf{b}, * \cdots *) \in^* \Delta(v)$ for all atoms $\beta(\mathbf{x}, \mathbf{y}, \mathbf{z}) \in \alpha$, then, 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 $z \in \mathbf{z}$ appears exactly once in α, also $\mathfrak{A} \models \exists \mathbf{z}.\alpha(g(\mathbf{a}), g(\mathbf{b}), \mathbf{z})$. Hence, we have

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

  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 $v \in \mathbf{V}$ 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 $v \in \mathbf{V}$ with $\beta(\mathbf{a}^*) \in \Delta(v)$ and a neighbour w with $\mathbf{a}^* \cap \mathbf{C}(w) \neq \emptyset$, then it adds $\beta(\mathbf{a}^*) \circledast \mathbf{C}(w)$ to $\Delta(w)$. From (§) we get that $\mathfrak{A} \models \beta(g(\mathbf{a}^*))$, and since $\beta(\mathbf{b}^*) := \beta(\mathbf{a}^*) \circledast \mathbf{C}(w) \geq^* \beta(\mathbf{a}^*)$, this implies $\mathfrak{A} \models \beta(g(\mathbf{b}^*))$. Hence, adding $\beta(\mathbf{a}^*) \circledast \mathbf{C}(w) = \beta(\mathbf{b}^*)$ to $\Delta(w)$ does not violate (§).
- If the R$\updownarrow\forall$-rule is applicable to a node $v \in \mathbf{V}$ with a universally quantified 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 3.10 *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 β such that $\{\beta(\mathbf{a}), \neg\beta(\mathbf{a})\} \subseteq \Delta(v)$. From (§), $\mathfrak{A} \models \{\beta(g(\mathbf{a})), \neg\beta(g(\mathbf{a}))\}$ would follow, 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 the initial tree together with the function g mapping a_0 to an arbitrary element of the universe of $\mathfrak{A}$. By Lemma 3.7, any sequence of applications is finite, and from Claim 3.9 we get that there is a sequence of rule-applications that maintains (§). By Claim 3.10, this sequence results in a tableau. This completes the proof of Lemma 3.8. ⊣

Lemma 3.8 involves two different kinds of non-determinism, namely, the choice which rule to apply to which constraint (as several rules might be applicable simultaneously), and which disjunct to choose in an application of the R∨-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∨-rule, but arbitrary strategies of choosing which rule to apply next, and where to apply it, 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 a model from a tableau. From the construction employed in the proof we obtain an alternative proof of Fact 3.2.

Lemma 3.11 *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 $k-1$.*

Proof: Let $\mathbf{T} = (\mathbf{V}, \mathbf{E}, \mathbf{C}, \Delta, \mathbf{N})$ a tableau for ψ. For every direct blocking situation we fix a mapping π verifying this blocking. Using an unraveling construction, we will construct a model $\mathfrak{A}$ for ψ of width at most $k-1$ from $\mathbf{T}$. First, we "unravel" blocking situations in $\mathbf{T}$ by

successively replacing every blocked node with a copy of the subtree of $\mathbf{T}$ rooted at the blocking node. Formally, this is achieved by the following path construction. We define

$$\mathbf{V}_u = \{v \in \mathbf{V} \ : \ v \text{ is not blocked or directly blocked } \}.$$

Since from now on we only deal with nodes from $\mathbf{V}_u$, every blocking is direct and we will no longer explicitly mention this fact.

The set $\mathsf{Paths}(\mathbf{T})$ is 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 $p = [\frac{v_1}{v'_1} \dots \frac{v_n}{v'_n}]$.

Intuitively, for every node v of $\mathbf{T}$, the paths $p \in \mathsf{Paths}(\mathbf{T})$ with $v = \mathsf{Tail}(p)$ stand for distinct copies of v created by the unraveling. The universe of $\mathfrak{A}$ consists of (classes of) constants labeling nodes in $\mathbf{T}$ paired with the paths at whose Tail they appear to distinguish constants occuring at different copies of a node of $\mathbf{T}$. Formally, we define

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

Constants appearing at consecutive nodes of $\mathbf{T}$ stand for the same element and the same holds for constants related by a mapping π verifying a block. Hence, to obtain the universe of $\mathfrak{A}$, we factorise $\mathbf{C}(\mathbf{T})$ as follows. Let $\sim$ be the smallest symmetric relation on $\mathbf{C}(\mathbf{T})$ satisfying

- $(a, p) \sim (a, q)$ if q is a successor of p in $\mathsf{Paths}(\mathbf{T})$, $\mathsf{Tail}'(q)$ 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 q is a successor of p in $\mathsf{Paths}(\mathbf{T})$, $\mathsf{Tail}'(q)$ 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 $\sim$ and with $[a, p]_\approx$ the class of (a, p), i.e., the set $\{(b, q) \in \mathbf{C}(\mathbf{T}) \mid (b, q) \approx (a, p)\}$. Since

[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 directly blocked by the same node w.

$(a,p) \sim (b,q)$ iff p, q are neighbours in $\mathsf{Paths}(\mathbf{T})$, for every (a,p), the set

$$\mathsf{Paths}([a,p]_{\approx}) := \{q \mid \exists b.(b,q) \in [a,p]_{\approx}\}$$

is a subtree of $\mathsf{Paths}(\mathbf{T})$.

The classes of $\mathbf{C}(\mathbf{T})/\approx$ will be the elements of the universe of $\mathfrak{A}$. First we need to prove some technicalities for this construction.

Claim 3.12 *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 $(c_1,p_1) \sim \cdots \sim (c_k,p_k)$ 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 $(c_i,p_i) \sim \cdots \sim (c_j,p_j)$ is also a shorter path with the same properties. Hence, a minimal 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 3.12, we get the following.

Claim 3.13 *Let $p, q \in \mathsf{Paths}(\mathbf{T})$ with $p = [\frac{v_1}{v'_1} \ldots \frac{v_n}{v'_n}]$, $q = [\frac{v_1}{v'_1} \ldots \frac{v_n}{v'_n} \frac{w}{w'}]$. If, for $a \in \mathbf{C}(v_n), b \in \mathbf{C}(w)$, $(a,p) \approx (b,q)$ then $(a,p) \sim (b,q)$.*

If $(a,p) \approx (b,q)$ 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 $q = p_k$. Since $\sim$ is only defined along paths in the tree $\mathsf{Paths}(\mathbf{T})$, there must be a step from p to q (or, dually, from q to p) in this path, more precisely, there must be an $i \in \{1, \ldots k-1\}$ such that $p_i = p$ and $p_{i+1} = q$ holds. Hence, we have the situation

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

Claim 3.12 implies $a = c_i$ and $b = c_{i+1}$ and hence $(a,p) \sim (b,q)$.

Using Claim 3.13, we can show that the blocking condition and the $\mathsf{R}\updownarrow$- and $\mathsf{R}\updownarrow\forall$-rule work as desired.

Claim 3.14 *Let* $p, q \in \mathsf{Paths}(\mathbf{T})$, $\mathbf{a} \subseteq \mathbf{C}(\mathsf{Tail}(p)), \mathbf{b} \subseteq \mathbf{C}(\mathsf{Tail}(q))$, $\mathbf{a}, \mathbf{b}$ *non-empty tuples, and* $(\mathbf{a}, p) \approx (\mathbf{b}, q)$.

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

Since both propositions are symmetric, we only need to prove one direction. If $(\mathbf{a}, p) \approx (\mathbf{b}, q)$ with $\mathbf{a} = a_1 a_2 \dots a_m$ and $\mathbf{b} = b_1 b_2 \dots b_m$, then

$$\{p, q\} \subseteq \bigcap_{i=1}^{m} \mathsf{Paths}([a_i, p]_\approx)$$

and, as an intersection of subtrees of $\mathsf{Paths}(\mathbf{T})$, $\bigcap_{i=1}^{m} \mathsf{Paths}([a_i, p]_\approx)$ is itself a subtree of $\mathsf{Paths}(\mathbf{T})$. Hence, in $\mathsf{Paths}(\mathbf{T})$ there is a path $p_1, \dots, p_k$ for which there exist tuples of constants $\mathbf{c}_1, \dots, \mathbf{c}_k$ with $(\mathbf{c}_1, p_1) \approx \cdots \approx (\mathbf{c}_k, p_k)$, $p = p_1$, $q = p_k$, $\mathbf{a} = \mathbf{c}_1$, and $\mathbf{b} = \mathbf{c}_k$. Since $\mathbf{a}, \mathbf{b}$ are non-empty, so are the $\mathbf{c}_i$. From Claim 3.13, we get that for 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 $(\mathbf{c}_i, p_i) \sim (\mathbf{c}_{i+1}, p_{i+1})$.

By two similar inductions on i with $1 \leq i \leq k$ we show that if $\beta(\mathbf{a}, * \cdots *) \in^* \Delta(\mathsf{Tail}(p))$ then $\beta(\mathbf{c}_i, * \cdots *) \in^* \Delta(\mathsf{Tail}(p_i))$ and if $\varphi(\mathbf{a}) \in \Delta(\mathsf{Tail}(p))$ then $\varphi(\mathbf{c}_i) \in \Delta(\mathsf{Tail}(\mathbf{c}_i))$.

For $i = 1$ in both cases nothing has to be shown. Now assume that the we have shown these properties up to i. W.l.o.g., assume p_{i+1} is a successor of p_i in the tree $\mathsf{Paths}(\mathbf{T})$. The other case is handled dually. There are two possibilities:

- $\mathsf{Tail}'(p_{i+1})$ 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 $\mathbf{c}_i = \mathbf{c}_{i+1}$ holds.
  If $\beta(\mathbf{a}, * \cdots *) \in^* \Delta(\mathsf{Tail}(p))$ then $\beta(\mathbf{c}_i, * \cdots *) \in^* \Delta(\mathsf{Tail}(p_i))$ holds by induction and due to the R↕-rule, this implies $\beta(\mathbf{c}_{i+1}, * \cdots *) \in^* \Delta(\mathsf{Tail}(p_{i+1}))$. The R↕-rule is applicable because, for the the non-empty tuple $\mathbf{c}_i$, $\mathbf{c}_i = \mathbf{c}_{i+1} \subseteq \mathbf{C}(\mathsf{Tail}(p_{i+1}))$ holds.
  If $\varphi(\mathbf{a}) \in \Delta(\mathsf{Tail}(p))$ then by induction $\varphi(\mathbf{c}_i) \in \Delta(\mathsf{Tail}(\mathbf{p}_i))$ and due to the R↕∀-rule this implies $\varphi(\mathbf{c}_{i+1}) \in \Delta(\mathsf{Tail}(p_{i+1}))$.
- $\mathsf{Tail}'(p_{i+1})$ is blocked by $\mathsf{Tail}(p_{i+1})$ (with function π) and $\mathsf{Tail}'(p_{i+1})$ is a successor of $\mathsf{Tail}(p_i)$ in $\mathbf{T}$. Then, by definition of $\sim$ we have $\mathbf{c}_{i+1} = \pi(\mathbf{c}_i)$ and $\mathbf{c}_i \subseteq \mathbf{C}(\mathsf{Tail}(p_i)) \cap \mathbf{C}(\mathsf{Tail}'(p_{i+1}))$.
  If $\beta(\mathbf{a}, * \cdots *) \in^* \Delta(\mathsf{Tail}(p))$ then $\beta(\mathbf{c}_i, * \cdots *) \in^* \Delta(\mathsf{Tail}(p_i))$ holds by induction and due to the R↕-rule this implies $\beta(\mathbf{c}_i, * \cdots *) \in^* \Delta(\mathsf{Tail}'(p_{i+1}))$. The R↕-rule is applicable because, for the non-empty tuple $\mathbf{c}_i$, $\mathbf{c}_i \subseteq \mathbf{C}(\mathsf{Tail}'(p_{i+1}))$ holds. The node $\mathsf{Tail}(p_{i+1})$

blocks $\mathsf{Tail}'(p_{i+1})$, which implies

$$\pi(\beta(\mathbf{c}_i, * \cdots *)) = \beta(\mathbf{c}_{i+1}, * \cdots *) \in^* \Delta(\mathsf{Tail}(p_{i+1})).$$

If $\varphi(\mathbf{a}) \in \Delta(\mathsf{Tail}(p))$ then by induction $\varphi(\mathbf{c}_i) \in \Delta(\mathsf{Tail}(p_i))$ and due to the $\mathsf{R}\updownarrow\forall$-rule this implies $\varphi(\mathbf{c}_i) \in \Delta(\mathsf{Tail}'(p_{i+1}))$. Since $\mathsf{Tail}(p_{i+1})$ blocks $\mathsf{Tail}'(p_{i+1})$, $\pi(\varphi(\mathbf{c}_i)) = \varphi(\mathbf{c}_{i+1}) \in \Delta(\mathsf{Tail}(p_{i+1}))$ holds.

We now define the structure $\mathfrak{A}$ over the universe $A = \mathbf{C}(\mathbf{T})/\!\approx$. For a relation $R \in \tau$ of arity m, $R^{\mathfrak{A}}$ is defined to be the set of tuples $([a_1, p_1]_\approx, \ldots, [a_m, p_m]_\approx)$ for which there exists a path $p \in \mathsf{Paths}(\mathbf{T})$ and constants $c_1, \ldots c_m$ such that $(c_i, p) \approx (a_i, p_i)$ for all $1 \leq i \leq m$, and $Rc_1 \ldots c_m \in \Delta(\mathsf{Tail}(p))$.

It remains to show that this construction yields $\mathfrak{A} \models \psi$. This is a consequence of the following claim.

Claim 3.15 *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 φ. If $\varphi(\mathbf{a}) = Ra_1 \ldots a_m \in \Delta(\mathsf{Tail}(p))$, 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 3.14 we have that $(\mathbf{a}, p) \approx (\mathbf{c}, p')$ implies $R\mathbf{a} \in^* \Delta(\mathsf{Tail}(p))$ and, since $\mathbf{a}$ contains no occurrence of $*$, $R\mathbf{a} \in \Delta(\mathsf{Tail}(p))$. Hence $\mathbf{T}$ contains the clash $\{R\mathbf{a}, \neg R\mathbf{a}\} \subseteq \Delta(\mathsf{Tail}(p))$, a contradiction to the fact that $\mathbf{T}$ is clash-free. Thus, $[\mathbf{a}, p]_\approx \notin R^{\mathfrak{A}}$.

Assume $\varphi(\mathbf{a}) = a \neq b \in \Delta(\mathsf{Tail}(p))$ but $[a, p]_\approx = [b, p]_\approx$. From Claim 3.12 we get that this implies $a = b$ and hence $\mathbf{T}$ contains the clash $a \neq a \in \Delta(\mathsf{Tail}(p))$. Again, this is a contradiction to the fact that $\mathbf{T}$ is clash-free and $[a, p]_\approx \neq [b, p]_\approx$ must hold.

For positive 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 $\mathbf{b}, \mathbf{p}, \mathbf{c}, \mathbf{q}$ arbitrarily chosen with

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

We need to show that also $\mathfrak{A} \models \chi([\mathbf{a}, p]_\approx, [\mathbf{b}, \mathbf{p}]_\approx)$ holds. In order to bring completeness of $\mathbf{T}$ and the $\mathsf{R}\forall$-rule into play, we show that information about the fact that (1) holds is present at a *single* node in $\mathbf{T}$ where it triggers the $\mathsf{R}\forall$-rule. We rely on the fact that universal quantifiers must be guarded.

Every $y_i \in \mathbf{y}$ coexists with every other variable $y_j \in \mathbf{y}$ in at least one atom $\beta^{(i,j)} \in \alpha(\mathbf{a}, \mathbf{y}, \mathbf{z})$ and with every element $a_\ell \in \mathbf{a}$ in at least

one atom $\gamma^{(i,\ell)} \in \alpha(\mathbf{a},\mathbf{y},\mathbf{z})$. For any two distinct variables y_i, y_j, $\mathfrak{A} \models \beta^{(i,j)}([\mathbf{a},p]_\approx, [\mathbf{b},\mathbf{p}]_\approx, [\mathbf{c},\mathbf{q}]_\approx)$ holds and this can only be the case if 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 $f^{(i,\ell)}, g^{(i,\ell)}$ such that $(b_i,p_i) \approx (f^{(i,\ell)}, r^{(i,\ell)})$ and $(a_\ell,p) \approx (g^{(i,\ell)}, r^{(i,\ell)})$. For every i and ℓ, $\mathsf{Paths}([b_i,p_i]_\approx)$ and $\mathsf{Paths}([a_\ell,p]_\approx)$ are subtrees of $\mathsf{Paths}(\mathbf{T})$.

The tree $\mathsf{Paths}([b_i,p_i]_\approx)$ overlaps with the tree $\mathsf{Paths}([b_j,p_j]_\approx)$ at $q^{(i,j)}$ and with the tree $\mathsf{Paths}([a_\ell,p]_\approx)$ at $r^{(i,\ell)}$. From this it follows (Golumbic 1980, Proposition 4.7) that there exists a common path

$$s \in \bigcap_i \mathsf{Paths}([b_i,p_i]_\approx) \cap \bigcap_\ell \mathsf{Paths}([a_\ell,p]_\approx).$$

Thus, there are tuples $\mathbf{a}'$, $\mathbf{b}'$ such that

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

We now show that the preconditions of the R$\forall$-rule are satisfied at $\mathsf{Tail}(s)$ for the formula $(\forall \mathbf{yz}.\alpha(\mathbf{a}',\mathbf{y},\mathbf{z}))\chi(\mathbf{a}',\mathbf{y})$ and the tuple $\mathbf{b}'$. First, due to Claim 3.14, $(\forall \mathbf{yz}.\alpha(\mathbf{a}',\mathbf{y},\mathbf{z}))\chi(\mathbf{a}',\mathbf{y}) \in \Delta(\mathsf{Tail}(s))$ holds because $(\mathbf{a},p) \approx (\mathbf{a}',s)$ and $(\forall \mathbf{yz}.\alpha(\mathbf{a},\mathbf{y},\mathbf{z}))\chi(\mathbf{a},\mathbf{y}) \in \Delta(\mathsf{Tail}(p))$.

For every $\beta(\mathbf{x},\mathbf{y},\mathbf{z}) \in \alpha(\mathbf{x},\mathbf{y},\mathbf{z})$, $\beta(\mathbf{a}',\mathbf{b}',*\cdots*) \in^* \Delta(\mathsf{Tail}(s))$ holds 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

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

It holds that $\beta(\mathbf{a}'',\mathbf{b}'',*\cdots*) \geq^* \beta(\mathbf{a}'',\mathbf{b}'',\mathbf{c}')$ and since $\beta(\mathbf{a}'',\mathbf{b}'',\mathbf{c}') \in \Delta(\mathsf{Tail}(t))$, $\beta(\mathbf{a}'',\mathbf{b}'',*\cdots*) \in^* \Delta(\mathsf{Tail}(t))$. Thus, by Claim 3.14 it holds that $\beta(\mathbf{a}',\mathbf{b}',*\cdots*) \in^* \Delta(\mathsf{Tail}(s))$.

Since this is true for every atom β, the preconditions of the R$\forall$-rule are satisfied and the completeness of $\mathbf{T}$ yields $\chi(\mathbf{a}',\mathbf{b}') \in \Delta(\mathsf{Tail}(s))$. By induction, $\mathfrak{A} \models \chi([\mathbf{a}',s]_\approx, [\mathbf{b}',s]_\approx)$ holds and together with (2) this implies $\mathfrak{A} \models \chi([\mathbf{a},p]_\approx, [\mathbf{b},\mathbf{p}]_\approx)$. Since $\mathbf{a},\mathbf{p},\mathbf{c},\mathbf{q}$ have been chose arbitrarily, $\mathfrak{A} \models \varphi([\mathbf{a},p]_\approx)$ holds.

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 $\mathbf{b},\mathbf{c} \subseteq \mathbf{C}(\mathsf{Tail}(p))$ with $\{\alpha(\mathbf{a},\mathbf{b},\mathbf{c}), \chi(\mathbf{a},\mathbf{b})\} \subseteq \Delta(\mathsf{Tail}(p))$. 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 $\mathbf{b}, \mathbf{c} \subseteq \mathbf{C}(w)$ 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
  $$\mathfrak{A} \models \{ \begin{array}{l} \alpha([\pi(\mathbf{a}), p']_\approx, [\pi(\mathbf{b}), p']_\approx, [\pi(\mathbf{c}), p']_\approx), \\ \chi([\pi(\mathbf{a}), p']_\approx, [\pi(\mathbf{b}), p']_\approx) \end{array} \},$$
  and by 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 3.15 we get that $\mathfrak{A} \models \psi$. From Lemma 3.6, we get that, for every node $v \in \mathbf{V}$, $|\mathbf{C}(v)| \leq \mathrm{width}(\psi)$ and hence the tree $\mathsf{Paths}(\mathbf{T})$ together with the function $f : \mathsf{Paths}(\mathbf{T}) \to \mathbf{C}(\mathbf{T})/\!\approx$ with $f(p) = \mathbf{C}(\mathsf{Tail}(p))/\!\approx$ provides a tree decomposition of $\mathfrak{A}$ of width $\leq \mathrm{width}(\psi) - 1$. This completes the proof of Lemma 3.11. ⊣

Corollary 3.16 CGF*, and hence also* LGF *and* GF *have the generalised tree model property.*

Proof: Let $\psi \in \mathrm{CGF}[\tau]$ be satisfiable. Then, from Lemma 3.8 we get that there is a tableau $\mathbf{T}$ for ψ. By Lemma 3.11, $\mathbf{T}$ induces a model for ψ of tree width at most $\mathrm{width}(\psi) - 1$. 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. For LGF and GF, observe that the embedding of these logics into CGF may increase the width of the sentence but not by more than a recursive amount. ⊣

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 we obtain

a new proof of the fact that GF/LGF/CGF have the generalised tree model property.

Acknowledgements

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

References

Andréka, H., J. van Benthem, and I. Németi. 1998. Modal Languages and Bounded Fragments of Predicate Logic. *Journal of Philosophical Logic* 27:217–274.

Donini, F. M., M. Lenzerini, D. Nardi, and W. Nutt. 1997. The Complexity of Concept Languages. *Information and Computation* 134(1):1–58.

Ganzinger, H., and H. de Nivelle. 1999. A Superposition Decision Procedure for the Guarded Fragment with Equality. In *Proc. 14th IEEE Symp. on Logic in Computer Science*, 295–303.

Golumbic, M. 1980. *Algorithmic Graph Theory and Perfect Graphs.* New York, NY: Academic Press.

Grädel, E. 1999a. Decision procedures for guarded logics. In *Automated Deduction - CADE16. Proceedings of 16th International Conference on Automated Deduction, Trento, 1999.* Lecture Notes in Artificial Intelligence, No. 1632. Springer-Verlag.

Grädel, E. 1999b. On the restraining power of guards. *Journal of Symbolic Logic* 64(4):1719–1742.

Grädel, E., and I. Walukiewicz. 1999. Guarded Fixed Point Logic. In *Proc. 14th IEEE Symp. on Logic in Computer Science*, 45–54.

Halpern, J. Y., and Y. Moses. 1992. A Guide to Completeness and Complexity for Model Logics of Knowledge and Belief. *Artificial Intelligence* 54(3):319–379.

Horrocks, I., P. F. Patel-Schneider, and R. Sebastiani. 2000. An Analysis of Empirical Testing for Modal Decision Procedures. *Logic Journal of the IGPL* 8(3):293–323.

Horrocks, I., U. Sattler, and S. Tobies. 1999. Practical Reasoning for Expressive Description Logics. In *Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99)*, ed. H. Ganzinger, D. McAllester, and A .Voronkov, 161–180. Lecture Notes in Artificial Intelligence, No. 1705. Springer-Verlag, September.

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

van Benthem, J. 1997. Dynamic bits and pieces. ILLC Research Report LP-97-01. University of Amsterdam.

Vardi, M. 1997. Why is modal logic so robustly decidable? In *Descriptive Complexity and Finite Models*, ed. N. Immerman and P. Kolaitis. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 31, 149–184. AMS.

Weidenbach, C. 1997. SPASS—Version 0.49. *J. of Automated Reasoning* 18(2):247–252.

Colin Hirsch
Mathematische Grundlagen der Informatik, RWTH Aachen
Ahornstr. 55
52064 Aachen
Germany
E-mail: hirsch@cs.rwth-aachen.de
URL: http://www-mgi.informatik.rwth-aachen.de/~hirsch/

Stephan Tobies
LuFG Theoretical Computer Science, RWTH Aachen
Ahronstr. 55
52064 Aachen
Germany
E-mail: tobies@cs.rwth-aachen.de
URL: http://www-lti.informatik.rwth-aachen.de/~tobies/