A Tableau Calculus with Automaton-Labelled Formulae for Regular Grammar Logics

Rajeev Goré[1] and Linh Anh Nguyen[2]

[1] The Australian National University and NICTA
Canberra ACT 0200, Australia
Rajeev.Gore@anu.edu.au
[2] Institute of Informatics, University of Warsaw
ul. Banacha 2, 02-097 Warsaw, Poland
nguyen@mimuw.edu.pl

Abstract. We present a sound and complete tableau calculus for the class of regular grammar logics. Our tableau rules use a special feature called automaton-labelled formulae, which are similar to formulae of automaton propositional dynamic logic. Our calculus is cut-free and has the analytic superformula property so it gives a decision procedure. We show that the known EXPTIME upper bound for regular grammar logics can be obtained using our tableau calculus. We also give an effective Craig interpolation lemma for regular grammar logics using our calculus.

1 Introduction

Multimodal logics (and their description logic cousins) are useful in many areas of computer science: for example, multimodal logics are used in knowledge representation by interpreting $[i]\varphi$ as "agent i knows/believes that φ is true" [7, 15, 1]. Grammar logics are normal multimodal logics characterised by "inclusion" axioms like $[t_1] \dots [t_h]\varphi \supset [s_1] \dots [s_k]\varphi$, where $[t_i]$ and $[s_j]$ are modalities indexed by members t_i and s_j from some fixed set of indices. Thus $[1][2]\varphi \rightarrow [1]\varphi$ captures "if agent one knows that agent two knows φ, then agent one knows φ".

Inclusion axioms correspond in a strict sense to grammar rules of the form $t_1 t_2 \dots t_h \rightarrow s_1 s_2 \dots s_k$ when the index set is treated as a set of atomic words and juxtaposition is treated as word composition. Various refinements ask whether the corresponding grammar is left or right linear, or whether the language generated by the corresponding grammar is regular, context-free etc.

Grammar logics were introduced by Fariñas del Cerro and Penttonen in [8] and have been studied widely [3, 4, 20, 11, 5]. Baldoni *et al.* [3] gave a prefixed tableau calculus for grammar logics and used it to show that the general satisfiability problem of right linear grammar logics is decidable and the general satisfiability problem of context-free grammar logics is undecidable. But the techniques of Baldoni *et al.* cannot be easily extended to regular grammar logics.

While trying to understand why the decidability proof by Baldoni *et al.* [3, 2] cannot be naturally extended to left linear grammars, Demri [4] observed that although right linear grammars generate the same class of languages as

left linear grammars, this correspondence is not useful at the level of regular grammar logics. By using a transformation into the satisfiability problem for propositional dynamic logic (PDL), Demri was able to prove that the general satisfiability problem of regular grammar logics is EXPTIME-complete and that the general satisfiability problem of linear grammar logics is undecidable. In [5], Demri and de Nivelle gave a translation of the satisfiability problem for grammar logics with converse into the two-variable guarded fragment GF^2 of first-order logic, and showed that the general satisfiability problem for regular grammar logics with converse is in EXPTIME. The relationship between grammar logics and description logics was considered, among others, in [11, 20].

Thus, various methods have been required to obtain complexity results and decision procedures for regular grammar logics. We show that it is possible to give a (non-prefixed) tableau calculus which is a decision procedure for the whole class of regular grammar logics, and which also gives an estimate of the complexity of these logics. Efficient tableaux for propositional multimodal (description) logics are highly competitive with translation methods, so it is not at all obvious that the translation into GF^2 from [5] is the best method for deciding these logics.

The naive way to encode inclusion axioms in a non-prefixed tableau calculus is to add a rule like ($[t]$) shown below at left. But such rules cannot lead to a *general* decision procedure because there are well-known examples like transitivity $[t]\varphi \supset [t][t]\varphi$, whose analogous rule is shown below at right, which immediately cause an infinite branch by adding $[t][t]\varphi$, and then $[t][t][t]\varphi$, and so on:

$$([t])\ \frac{X; [t]\varphi}{X; [t]\varphi; [s_1][s_2]\ldots[s_k]\varphi} \qquad (4t)\ \frac{X; [t]\varphi}{X; [t]\varphi; [t][t]\varphi}$$

Our calculus uses a special feature called automaton-labelled formulae, which are similar to formulae of APDL [10]. Informally, whenever a formula $[t]\varphi$ is true at a tableau node w, we add an automaton labelled formula that tracks the modal transitions from w. If a sequence of transitions leads to a tableau node u, and this sequence corresponds to a word $s_1 s_2 \ldots s_k$ recognised by the automaton labelled formula, then we add the formula φ to u. This captures the effect of the rule ($[t]$) above left in a tractable manner since the influence of $[t]\varphi$ being true at w can be computed directly from the content of the automaton labelled formulae in node u. Our tableau calculus is sound, complete, cut-free and has the analytic superformula property, so it is a decision procedure. As usual for tableau calculi, it allows efficient implementation and good complexity estimation.

In Section 2, we define regular grammar logics and automaton-labelled formulae. In Section 3, we present our tableau calculus for regular grammar logics, and prove it sound. In Section 4, we prove it complete. In Section 5, we prove that the general satisfiability problem of regular grammar logics is in EXPTIME by using our tableau rules in a systematic way. In Section 6, we use our calculus to prove effective Craig interpolation for regular grammar logics. Further work and concluding remarks are in Section 7. The Appendix contains an example.

Acknowledgements: We are grateful to Pietro Abate, Stéphane Demri, Marcus Kracht and an anonymous reviewer for their helpful comments and pointers.

2 Preliminaries

2.1 Definitions for Multimodal Logics

Our modal language is built from two disjoint sets: $\mathcal{MOD}$ is a finite set of modal indices and $\mathcal{PROP}$ is a set of primitive propositions. We use p and q for elements of $\mathcal{PROP}$ and use t and s for elements of $\mathcal{MOD}$. Formulae of our primitive language are recursively defined using the BNF grammar below:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \supset \varphi \mid [t]\varphi \mid \langle t\rangle\varphi$$

A *Kripke frame* is a tuple $\langle W, \tau, \{R_t \mid t \in \mathcal{MOD}\}\rangle$, where W is a nonempty set of possible worlds, $\tau \in W$ is the current world, and each R_t is a binary relation on W, called the accessibility relation for $[t]$ and $\langle t\rangle$. If $R_t(w, u)$ holds then we say that the world u is accessible from the world w via R_t.

A *Kripke model* is a tuple $\langle W, \tau, \{R_t \mid t \in \mathcal{MOD}\}, h\rangle$, where $\langle W, \tau, \{R_t \mid t \in \mathcal{MOD}\}\rangle$ is a Kripke frame and h is a function mapping worlds to sets of primitive propositions. For $w \in W$, the set of primitive propositions "true" at w is $h(w)$.

A *model graph* is a tuple $\langle W, \tau, \{R_t \mid t \in \mathcal{MOD}\}, H\rangle$, where $\langle W, \tau, \{R_t \mid t \in \mathcal{MOD}\}\rangle$ is a Kripke frame and H is a function mapping worlds to formula sets. We sometimes treat model graphs as models with H restricted to $\mathcal{PROP}$.

Given a Kripke model $M = \langle W, \tau, \{R_t \mid t \in \mathcal{MOD}\}, h\rangle$ and a world $w \in W$, the *satisfaction relation* $\models$ is defined as usual for the classical connectives with two extra clauses for the modalities as below:

$$\begin{array}{lll} M, w \models [t]\varphi & \text{iff} & \forall v \in W. R_t(w, v) \text{ implies } M, v \models \varphi \\ M, w \models \langle t\rangle\varphi & \text{iff} & \exists v \in W. R_t(w, v) \text{ and } M, v \models \varphi. \end{array}$$

We say that φ *is satisfied at* w *in* M if $M, w \models \varphi$. We say that φ *is satisfied in* M and call M a *model of* φ if $M, \tau \models \varphi$.

If we consider only Kripke models, with no restrictions on R_t, we obtain a normal multimodal logic with a standard Hilbert-style axiomatisation K_n.

Note: We now assume that formulae are in *negation normal form*, where $\supset$ is translated away and $\neg$ occurs only directly before primitive propositions. Every formula φ has a logically equivalent formula φ' which is in negation normal form.

2.2 Regular Grammar Logics

Recall that a *finite automaton* A is a tuple $\langle \Sigma, Q, I, \delta, F\rangle$, where Σ is the alphabet (for our case, $\Sigma = \mathcal{MOD}$), Q is a finite set of states, $I \subseteq Q$ is the set of initial states, $\delta \subseteq Q \times \Sigma \times Q$ is the transition relation, and $F \subseteq Q$ is the set of accepting states. A *run* of A on a word $s_1 \ldots s_k$ is a finite sequence of states $q_0, q_1, \ldots, q_k$ such that $q_0 \in I$ and $\delta(q_{i-1}, s_i, q_i)$ holds for every $1 \leq i \leq k$. It is an *accepting run* if $q_k \in F$. We say that A *accepts* word w if there exists an accepting run of A on w. The set of all words accepted/recognised by A is denoted by $\mathcal{L}(A)$.

Given two binary relations R_1 and R_2 over W, their relational composition $R_1 \circ R_2 = \{(x, y) \mid \exists y \in W. R_1(x, y) \;\&\; R_2(y, z)\}$ is also a binary relation over W.

A *grammar logic* is a multimodal logic extending K_n with "inclusion axioms" of the form $[t_1]\dots[t_h]\varphi \supset [s_1]\dots[s_k]\varphi$, where $\{t_1,\dots t_h, s_1,\dots s_k\} \subseteq \mathcal{MOD}$. Each inclusion axiom corresponds to the restriction $R_{s_1}\circ\dots\circ R_{s_k} \subseteq R_{t_1}\circ\dots\circ R_{t_h}$ on accessibility relations where the corresponding side stands for the identity relation if $k = 0$ or $h = 0$. For a grammar logic L, the *L-frame restrictions* are the set of all such corresponding restrictions. A Kripke model is an *L-model* if its frame satisfies all L-frame restrictions. A formula φ is *L-satisfiable* if there exists an L-model satisfying it. A formula φ is *L-valid* if it is satisfied in all L-models.

An inclusion axiom $[t_1]\dots[t_h]\varphi \supset [s_1]\dots[s_k]\varphi$ can also be seen as the grammar rule $t_1\dots t_h \to s_1\dots s_k$ where the corresponding side stands for the empty word if $k = 0$ or $h = 0$. Thus the inclusion axioms of a grammar logic L capture a grammar $\mathcal{G}(L)$. Here we do not distinguish terminal symbols and nonterminal symbols. $\mathcal{G}(L)$ is *context-free* if its rules are of the form $t \to s_1\dots s_k$, and is *regular* if it is context-free and for every $t \in \mathcal{MOD}$ there exists a finite automaton A_t that recognises the words derivable from t using $\mathcal{G}(L)$.

A *regular grammar logic* L is a grammar logic whose inclusion axioms correspond to grammar rules that collectively capture a regular grammar $\mathcal{G}(L)$. A regular language is traditionally specified either by a regular expression or by a left/right linear grammar or by a finite automaton. The first two forms can be transformed in PTIME to an equivalent finite automaton that is at most polynomially larger. But there is no syntactic way to specify the class of regular (context-free) grammars, and checking whether a context-free grammar generates a regular language is undecidable (see, e.g., [14]). Hence, we cannot compute these automata if we are given an arbitrary regular grammar logic. We therefore assume that for each $t \in \mathcal{MOD}$ we are given an automaton A_t recognising the words derivable from t using $\mathcal{G}(L)$. These are the *automata specifying L*.

Lemma 1. *Let L be a regular grammar logic and let $\{A_t \mid t \in \mathcal{MOD}\}$ be the automata specifying L. Then the following conditions are equivalent:*

(i) *the word $s_1\dots s_k$ is accepted by A_t*
(ii) *the formula $[t]\varphi \supset [s_1]\dots[s_k]\varphi$ is L-valid*
(iii) *the inclusion $R_{s_1}\circ\dots\circ R_{s_k} \subseteq R_t$ is a consequence of the L-frame restrictions.*

Proof. The equivalence $(ii) \Leftrightarrow (iii)$ is well-known from correspondence theory [19]. The implication $(i) \Rightarrow (ii)$ follows by induction on the length of the derivation of $s_1\dots s_k$ from t by the grammar $\mathcal{G}(L)$, using substitution, the K-axiom $[t](\varphi \supset \psi) \supset ([t]\varphi \supset [t]\psi)$ and the modal necessitation rule $\varphi/[t]\varphi$. The implication $(iii) \Rightarrow (i)$ follows by induction on the length of the derivation of $R_{s_1}\circ\dots\circ R_{s_k} \subseteq R_t$ from the L-frame restrictions. See also [3, 4] for details.

Example 1. Let $\mathcal{MOD} = \{1,\dots,m\}$ for a fixed m. Consider the grammar logic with the inclusion axioms $[i]\varphi \supset [j][i]\varphi$ for any $i,j \in \mathcal{MOD}$ and $[i]\varphi \supset [j]\varphi$ if $i > j$. This is a regular grammar logic because the set of words derivable from i using the corresponding grammar is $\{1,\dots,m\}^*.\{1,\dots,i\}$. This set is recognised by the automaton $A_i = \langle \mathcal{MOD}, \{p,q\}, \{p\}, \delta_i, \{q\}\rangle$ with $\delta_i = \{(p,j,p) \mid 1 \le j \le m\} \cup \{(p,j,q) \mid 1 \le j \le i\}$. Note that the corresponding grammar is not "linear" in that at most one symbol in the right hand side of a rule can be nonterminal.

2.3 Automaton-Labelled Formulae

If A is a finite automaton, Q is a subset of the states of A, and φ is a formula in the primitive language then $(A, Q) : \varphi$ is an *automaton-labelled formula*.

Fix a regular grammar logic L and let $\{A_t = \langle \mathcal{MOD}, Q_t, I_t, \delta_t, F_t\rangle \mid t \in \mathcal{MOD}\}$ be the automata specifying L. Let $\delta_t(Q, s) = \{q' \mid \exists q \in Q.(q, s, q') \in \delta_t\}$ be the states which can be reached from Q via an s-transition using A_t. The intuitions of automaton labelled formulae are as follows:

Tagging: A formula of the form $[t]\varphi$ in a world u is represented by $(A_t, I_t) : \varphi$.
Tracking: If $(A_t, Q) : \varphi$ occurs at u and $R(u, v)$ holds then we add the formula $(A_t, \delta_t(Q, s)) : \varphi$ to v. In particular, if $(A_t, I_t) : \varphi$ appears in world u and $R_s(u, v)$ holds then we add $(A_t, \delta_t(I_t, s)) : \varphi$ to the world v.
Acceptance: If $(A_t, Q) : \varphi$ occurs at u and Q contains an accepting state of A_t, then we add φ to u.

The formal semantics of automaton-labelled formulae are defined as follows. Let ε be the empty word and define $\widetilde{\delta_t}(Q, \varepsilon) = Q$ and $\widetilde{\delta_t}(Q, s_1 \dots s_k) = \widetilde{\delta_t}(\delta_t(Q, s_1), s_2 \dots s_k)$. If M is a Kripke model, w is a world of M, and $A_t = \langle \mathcal{MOD}, Q_t, I_t, \delta_t, F_t\rangle$ is an automaton, then $M, w \models (A_t, Q) : \varphi$ iff there exist worlds $w_0, \dots, w_k = w$ (of M) and indices $s_1, \dots, s_k \in \mathcal{MOD}$ such that $M, w_0 \models [t]\varphi$, $R_{s_i}(w_{i-1}, w_i)$ holds for $1 \leq i \leq k$, and $\widetilde{\delta_t}(I_t, s_1 \dots s_k) = Q$. Pictorially: $M, w \models (A_t, Q) : \varphi$ iff

$$w_0 \xrightarrow{s_1} w_1 \xrightarrow{s_2} \cdots \qquad\qquad w_{k-1} \xrightarrow{\quad s_k \quad} w_k = w$$

$$[t]\varphi \qquad \widetilde{\delta}(I_t, s_1) \qquad \cdots \qquad \widetilde{\delta}(I_t, s_1 \dots s_{k-1}) \qquad \widetilde{\delta}(I_t, s_1 \dots s_k) = Q$$

We can see the soundness of these definitions by the following inter-derivable sequence of validities of multimodal tense logic which use the residuation properties of $\langle s\rangle^{-1}$ and $[s]$ shown at right where $\langle s\rangle^{-1}$ is the converse of $\langle s\rangle$:

$$\dfrac{\dfrac{\dfrac{[t]\varphi \supset [s_1][s_2]\dots[s_k]\varphi}{\langle s_1\rangle^{-1}[t]\varphi \supset [s_2]\dots[s_k]\varphi}}{\dots}}{\langle s_k\rangle^{-1}\dots\langle s_1\rangle^{-1}[t]\varphi \supset \varphi} \qquad\qquad \dfrac{\varphi \supset [s]\psi}{\langle s\rangle^{-1}\varphi \supset \psi}$$

That is, if we want to ensure that $[t]\varphi \supset [s_1][s_2]\dots[s_k]\varphi$ is valid, then it suffices to ensure that $\langle s_k\rangle^{-1}\dots\langle s_1\rangle^{-1}[t]\varphi \supset \varphi$ is valid instead. But converse modalities are not part of our official language so we use the occurrence of $[t]\varphi$ at w_0 to start an automaton A_t which tracks the following constraint: φ must be true at any world w reachable from w_0 by the path/word $s_1 \dots s_k$.

Our automaton-labelled formulae are similar to formulae of automaton propositional dynamic logic (APDL) [10]. A formula involving automata in APDL is of the form $[A]\varphi$, where A is a finite automata with *one* initial state

and *one* accepting state. An automaton labelled formula like our $(A_t, Q) : \varphi$ with $Q = \{q_1, q_2, \ldots, q_k\}$ can be simulated by the APDL formula $[B_1]\varphi \vee \ldots \vee [B_k]\varphi$ where each B_i is the automaton A_t restricted to start at the initial state q_i. Thus our formulation uses a more compact representation in which APDL formulae that differ only in their initial state are grouped together. Moreover, we do not treat different "states" of an automaton as different automata. Our compact representation not only saves memory but also increases efficiency of deduction.

From now on, by a *formula* we mean either a formula in the primitive language (as defined in Section 2.1) or an automaton-labelled formula.

2.4 Definitions for Tableau Calculi

As in our previous works on tableau calculi [9, 17], our tableau formulation trace their roots to Hintikka via [18]. A *tableau rule* σ consists of a numerator N above the line and a (finite) list of denominators D_1, D_2, ..., D_k (below the line) separated by vertical bars. The numerator is a finite formula set, and so is each denominator. As we shall see later, each rule is read downwards as "if the numerator is L-satisfiable, then so is one of the denominators". The numerator of each tableau rule contains one or more distinguished formulae called the *principal formulae*. A *tableau calculus* $\mathcal{C}L$ for a logic L is a finite set of tableau rules.

A $\mathcal{C}L$-tableau for X is a tree with root X whose nodes carry finite formula sets obtained from their parent nodes by instantiating a tableau rule with the proviso that if a child s carries a set Z and Z has already appeared on the branch from the root to s then s is an *end node*.

Let Δ be a set of tableau rules. We say that Y is *obtainable from X by applications of rules from Δ* if there exists a tableau for X which uses only rules from Δ and has a node that carries Y. A branch in a tableau is *closed* if its end node carries only $\bot$. A tableau is *closed* if every one of its branches is closed. A tableau is *open* if it is not closed. A finite formula set X in the primitive language is said to be *$\mathcal{C}L$-consistent* if every $\mathcal{C}L$-tableau for X is open. If there is a closed $\mathcal{C}L$-tableau for X then we say that X is *$\mathcal{C}L$-inconsistent*.

A tableau calculus $\mathcal{C}L$ is *sound* if for all finite formula sets X in the primitive language, X is L-satisfiable implies X is $\mathcal{C}L$-consistent. It is *complete* if for all finite formula sets X in the primitive language, X is $\mathcal{C}L$-consistent implies X is L-satisfiable. Let σ be a rule of $\mathcal{C}L$. We say that σ is sound w.r.t. L if for every instance σ' of σ, if the numerator of σ' is L-satisfiable then so is one of the denominators of σ'. Any $\mathcal{C}L$ containing only rules sound w.r.t. L is sound.

3 A Tableau Calculus for Regular Grammar Logics

Fix a regular grammar logic L and let $\{A_t = \langle \mathcal{MOD}, Q_t, I_t, \delta_t, F_t\rangle \mid t \in \mathcal{MOD}\}$ be the automata specifying L. Recall that formulae are in negation normal form. We use X for a formula set, and semicolon to separate elements of a formula set. We have deliberately used "s.t." for "such that" in the set notation of the

$$(\bot)\ \frac{X;p;\neg p}{\bot} \qquad (\wedge)\ \frac{X;\varphi\wedge\psi}{X;\varphi;\psi} \qquad (\vee)\ \frac{X;\varphi\vee\psi}{X;\varphi \mid X;\psi}$$

$$(\mathsf{label})\ \frac{X;[t]\varphi}{X;(A_t,I_t){:}\varphi} \qquad (\mathsf{add})\ \frac{X;(A_t,Q){:}\varphi}{X;(A_t,Q){:}\varphi;\varphi}\ \text{if } Q\cap F_t\neq\emptyset$$

$$(\mathsf{trans})\ \frac{X;\langle t\rangle\varphi}{\{(A_s,\delta_s(Q,t)){:}\psi\ s.t.\ (A_s,Q){:}\psi\in X\};\varphi}$$

Fig. 1. Tableau Rules

denominator of the (trans)-rule because we use colon in automaton-labelled formulae and the alternative | indicates a branching rule! The tableau calculus $\mathcal{CL}$ is given in Figure 1. The first five rules are *static* rules, and the last rule is a *transitional* rule. An example is in the appendix.

A tableau calculus $\mathcal{CL}$ has the *analytic superformula* property iff to every finite set X we can assign a finite set $X^*_{\mathcal{CL}}$ which contains all formulae that may appear in any tableau for X. We write $Sf(\varphi)$ for the set of all subformulae of φ, and $Sf(X)$ for the set $\bigcup_{\varphi\in X} Sf(\varphi)\cup\{\bot\}$. Our calculus has the analytic superformula property, with $X^*_{\mathcal{CL}} = Sf(X)\cup\{(A_t,Q){:}\varphi\ s.t.\ [t]\varphi\in Sf(X) \text{ and } Q\subseteq Q_t\}$.

Lemma 2. *The tableau calculus $\mathcal{CL}$ is sound.*

Proof. We show that $\mathcal{CL}$ contains only rules sound w.r.t. L as follows. Suppose that the numerator of the considered rule is satisfied at a world w in a model $M = \langle W,\tau,\{R_t \mid t\in\mathcal{MOD}\},h\rangle$. We have to show that at least one of the denominators of the rule is also satisfiable. For the static rules, we show that some denominator is satisfied at w itself. For the transitional rule (trans), we show that its denominator is satisfied at some world reachable from w via R_t.

$(\bot),(\wedge),(\vee)$**:** These cases are obvious.
(label)**:** If $M,w\models X;[t]\varphi$ then $M,w\models(A_t,I_t){:}\varphi$ by definition.
(add)**:** Suppose that $M,w\models X;(A_t,Q){:}\varphi$ and $Q\cap F_t\neq\emptyset$. By definition, there exist worlds $w_0,\ldots,w_{k-1},w_k=w$ and indices $s_1,\ldots,s_k\in\mathcal{MOD}$ such that $M,w_0\models[t]\varphi$, and $R_{s_i}(w_{i-1},w_i)$ holds for $1\leq i\leq k$, and $\widetilde{\delta_t}(I_t,s_1\ldots s_k)=Q$. Since $Q\cap F_t\neq\emptyset$, the word $s_1\ldots s_k$ is accepted by A_t. By Lemma 1, it follows that $M,w_0\models[s_1]\ldots[s_k]\varphi$. Since $w=w_k$ and $R_{s_i}(w_{i-1},w_i)$ holds for $1\leq i\leq k$, we must have $M,w\models\varphi$.
(trans)**:** Suppose that $M,w\models X;\langle t\rangle\varphi$. Then there exists some u such that $R_t(w,u)$ holds and $M,u\models\varphi$. For each $(A_s,Q){:}\psi\in X$, we have $M,w\models(A_s,Q){:}\psi$, and by the semantics of automaton-labelled formulae, it follows that $M,u\models(A_s,\delta_s(Q,t)){:}\psi$. Hence, the denominator is satisfied at u.

4 Completeness

We prove completeness of our calculus via model graphs following [18, 9, 16, 17] by giving an algorithm that accepts a finite $\mathcal{CL}$-consistent formula set X in the primitive language and constructs an L-model graph (defined in Section 4.2) for X that satisfies every one of its formulae at the appropriate world.

4.1 Saturation

In the rules $(\wedge)$, $(\vee)$, (label) the principal formula does not occur in the denominators. For any of these rules δ, let δ' denote the rule obtained from δ by adding the principal formula to each of the denominators. Let $\mathcal{SCL}$ denote the set of static rules of $\mathcal{CL}$ with $(\wedge)$, $(\vee)$, (label) replaced by $(\wedge')$, $(\vee')$, (label'). For every rule of $\mathcal{SCL}$, except $(\bot)$, the numerator is included in each of the denominators.

For a finite $\mathcal{CL}$-consistent formula set X, a formula set Y is called a *$\mathcal{CL}$-saturation of X* if Y is a maximal $\mathcal{CL}$-consistent set obtainable from X by applications of the rules of $\mathcal{SCL}$. A set X is *closed w.r.t. a tableau rule* if applying that rule to X gives back X as one of the denominators.

Lemma 3. *Let X be a finite $\mathcal{CL}$-consistent formula set and Y a $\mathcal{CL}$-saturation of X. Then $X \subseteq Y \subseteq X^*_{\mathcal{CL}}$ and Y is closed w.r.t. the rules of $\mathcal{SCL}$. Furthermore, there is an effective procedure that, given a finite $\mathcal{CL}$-consistent formula set X, constructs some $\mathcal{CL}$-saturation of X.*

Proof. It is clear that $X \subseteq Y \subseteq X^*_{\mathcal{CL}}$. Observe that if a rule of $\mathcal{SCL}$ is applicable to Y, then one of the corresponding instances of the denominators is $\mathcal{CL}$-consistent. Since Y is a $\mathcal{CL}$-saturation, Y is closed w.r.t. the rules of $\mathcal{SCL}$.

We construct a $\mathcal{CL}$-saturation of X as follows: let $Y = X$; while there is some rule δ of $\mathcal{SCL}$ applicable to Y such that one of its corresponding denominator instance Z is $\mathcal{CL}$-consistent and strictly contains Y, set $Y = Z$. At each iteration, $Y \subset Z \subseteq X^*_{\mathcal{CL}}$. Hence the above process always terminates. It is clear that the resulting set Y is a $\mathcal{CL}$-saturation of X.

4.2 Proving Completeness via Model Graphs

A model graph is an *L-model graph* if its frame is an L-frame. An L-model graph $\langle W, \tau, \{R_t \mid t \in \mathcal{MOD}\}, H\rangle$ is *saturated* if every $w \in W$ satisfies:

And: if $\varphi \wedge \psi \in H(w)$ then $\{\varphi, \psi\} \subseteq H(w)$;
Or: if $\varphi \vee \psi \in H(w)$ then $\varphi \in H(w)$ or $\psi \in H(w)$;
Box: if $[t]\varphi \in H(w)$ and $R_t(w, u)$ holds then $\varphi \in H(u)$;
Dia: if $\langle t\rangle\varphi \in H(w)$ then there exists a $u \in W$ such that $R_t(w, u)$ and $\varphi \in H(u)$.

A saturated model graph is *consistent* if no world contains $\bot$, and no world contains $\{p, \neg p\}$. Our model graphs merely denote a data structure, while Rautenberg's model graphs are required to be saturated and consistent.

Lemma 4. *If $M = \langle W, \tau, \{R_t \mid t \in \mathcal{MOD}\}, H\rangle$ is a consistent saturated L-model graph, then M satisfies all formulae of $H(\tau)$ which are in the primitive language.*

Proof. By proving $\varphi \in H(w)$ implies $M, w \models \varphi$ by induction on the length of φ.

Given a finite $\mathcal{CL}$-consistent set X in the primitive language, we construct a consistent saturated L-model graph $M = \langle W, \tau, \{R_t \mid t \in \mathcal{MOD}\}, H\rangle$ such that $X \subseteq H(\tau)$, thereby giving an L-model for X.

4.3 Constructing Model Graphs

In the following algorithm, the worlds of the constructed model graph are marked either as *unresolved* or as *resolved*.

Algorithm 1
Input: a finite $\mathcal{C}L$-consistent set X of formulae in the primitive language.
Output: an L-model graph $M = \langle W, \tau, \{R_t \mid t \in \mathcal{MOD}\}, H\rangle$ satisfying X.

1. Let $W = \{\tau\}$, $H(\tau)$ be a $\mathcal{C}L$-saturation of X, and $R'_t = \emptyset$ for all $t \in \mathcal{MOD}$. Mark τ as unresolved.
2. While there are unresolved worlds, take one, say w, and do the following:
   (a) For every formula $\langle t\rangle\varphi$ in $H(w)$:
      i. Let $Y = \{(A_s, \delta_s(Q,t)){:}\psi \text{ s.t. } (A_s, Q){:}\psi \in H(w)\} \cup \{\varphi\}$ be the result of applying rule (trans) to $H(w)$, and let Z be a $\mathcal{C}L$-saturation of Y.
      ii. If $\exists u \in W$ on the path from the root to w with $H(u) = Z$, then add the pair (w, u) to R'_t. Otherwise, add a new world w_φ with content Z to W, mark it as unresolved, and add the pair (w, w_φ) to R'_t.
   (b) Mark w as resolved.
3. Let R_t be the least extension of R'_t for $t \in \mathcal{MOD}$ such that $\langle W, \tau, \{R_t \mid t \in \mathcal{MOD}\}\rangle$ is an L-frame.

This algorithm always terminates: eventually, for every w, either w contains no $\langle t\rangle$-formulae, or there exists an ancestor with $H(u) = Z$ at Step 2(a)ii because all $\mathcal{C}L$-saturated sets are drawn from the finite and fixed set $X^*_{\mathcal{C}L}$.

Lemma 5. *Suppose $R_t(w,u)$ holds via Step 3. Then there exist $w_0, \ldots, w_k$ in M with $w_0 = w$, $w_k = u$, and indices $s_1, \ldots, s_k \in \mathcal{MOD}$ such that $R'_{s_i}(w_{i-1}, w_i)$ holds for $1 \le i \le k$, and $R_{s_1} \circ \ldots \circ R_{s_k} \subseteq R_t$ follows from the L-frame restrictions.*

Proof. By induction on number of inferences in deriving $R_t(w,u)$ when extending R'_s to R_s for $s \in \mathcal{MOD}$, with L-frame restrictions of the form $R_{t_1} \circ \ldots \circ R_{t_h} \subseteq R_s$.

4.4 Completeness Proof

Lemma 6. *Let X be a finite $\mathcal{C}L$-consistent set of formulae in the primitive language and $M = \langle W, \tau, \{R_t \mid t \in \mathcal{MOD}\}, H\rangle$ be the model graph for X constructed by Algorithm 1. Then M is a consistent saturated L-model graph satisfying X.*

Proof. It is clear that M is an L-model graph and for any $w \in W$, the set $H(w)$ is $\mathcal{C}L$-consistent. We want to show that M is a saturated model graph. It suffices to show that, for every $w, u \in W$, if $[t]\varphi \in H(w)$ and $R_t(w,u)$ holds then $\varphi \in H(u)$. Suppose that $[t]\varphi \in H(w)$ and $R_t(w,u)$ holds. By Lemma 5, there exist worlds $w_0, \ldots, w_k$ with $w_0 = w$, $w_k = u$ and indices $s_1, \ldots, s_k \in \mathcal{MOD}$ such that $R'_{s_i}(w_{i-1}, w_i)$ holds for $1 \le i \le k$ and $R_{s_1} \circ \ldots \circ R_{s_k} \subseteq R_t$ is a consequence of the L-frame restrictions. Since $H(w)$ is a $\mathcal{C}L$-saturation, we have that $(A_t, I_t){:}\,\varphi \in H(w)$. By Step 2a of Algorithm 1, $(A_t, \widetilde{\delta}_t(I_t, s_1 \ldots s_i)){:}\varphi \in H(w_i)$ for $1 \le i \le k$. Thus $(A_t, \widetilde{\delta}_t(I_t, s_1 \ldots s_k)){:}\varphi \in H(u)$. Since $R_{s_1} \circ \ldots \circ R_{s_k} \subseteq R_t$ is a consequence of the L-frame restrictions, by Lemma 1, the word $s_1 \ldots s_k$ is accepted by A_t. Hence $\widetilde{\delta}_t(I_t, s_1 \ldots s_k) \cap F_t \neq \emptyset$. It follows that $\varphi \in H(u)$, since $H(u)$ is a $\mathcal{C}L$-saturation.

The following theorem follows from Lemmas 2 and 6.

Theorem 1. *The tableau calculus $\mathcal{C}L$ is sound and complete.*

5 Complexity

The satisfiability problem of a logic L is to check the L-satisfiability of an input formula φ. The general satisfiability problem of a class $\mathcal{C}$ of logics is to check L-satisfiability of an input formula φ in an input logic $L \in \mathcal{C}$.

Demri [4] proved that the general satisfiability problem of regular grammar logics is EXPTIME-complete by a transformation into satisfiability for PDL. We now obtain the upper bound EXPTIME using our tableaux calculus.

We need a rule $(\cup)$ to coalesce $(A_t, Q)\colon\varphi$ and $(A_t, Q')\colon\varphi$ into $(A_t, Q \cup Q')\colon\varphi$

$$(\cup)\ \frac{X\ ;\ (A_t, Q)\colon\varphi\ ;\ (A_t, Q')\colon\varphi}{X\ ;\ (A_t, Q \cup Q')\colon\varphi}$$

Observe that $X; (A_t, Q)\colon\varphi; (A_t, Q')\colon\varphi$ is $\mathcal{C}L$-consistent iff $X; (A_t, Q \cup Q')\colon\varphi$ is $\mathcal{C}L$-consistent. This follows from the facts that $\delta_t(Q, s) \cup \delta_t(Q', s) = \delta_t(Q \cup Q', s)$ and $((Q \cap F_t \neq \emptyset) \vee (Q' \cap F_t \neq \emptyset)) \equiv ((Q \cup Q') \cap F_t \neq \emptyset)$. Thus, rule $(\cup)$ can be added to $\mathcal{C}L$ as a *static* rule, and used whenever possible without affecting soundness and completeness. Let $\mathcal{C}L\mathrm{u}$ be $\mathcal{C}L$ plus $(\cup)$.

Allowing $(\cup)$ requires a change in the semantics of automaton-labelled formulae to: if M is a Kripke model, w is a world of M, and $A_t = \langle \mathcal{MOD}, Q_t, I_t, \delta_t, F_t \rangle$ is an automaton, then $M, w \models (A_t, Q) : \varphi$ iff for every $q \in Q$ there exist worlds $w_0, \ldots, w_{k-1}, w_k = w$ (of M) and indices $s_1, \ldots, s_k \in \mathcal{MOD}$ such that $M, w_0 \models [t]\varphi$, and $R_{s_i}(w_{i-1}, w_i)$ holds for $1 \leq i \leq k$, and $q \in \widetilde{\delta_t}(I_t, s_1 \ldots s_k) \subseteq Q$.

Let L be a regular logic and X a finite formula set in the primitive language. Let n be the sum of the sizes of the formulae in X and the sizes of the automata specifying L. To check whether X is L-satisfiable we can search for a closed $\mathcal{C}L\mathrm{u}$-tableau for X, or equivalently, examine an and-or tree for X constructed by using Algorithm 1 to apply our $\mathcal{C}L\mathrm{u}$-tableau rules in a systematic way. In such a tree, and-branching is caused by all possible applications of rule (trans), while or-branching is caused by an application of rule $(\vee)$. The other $\mathcal{C}L\mathrm{u}$-tableau rules, including $(\cup)$, are applied locally for each node whenever possible.

There are at most $O(n)$ unlabelled subformulae of X, and at most $2^{O(n)}$ different labels. By using the rule $(\cup)$ whenever possible, each subformula of X occurs in a node with at most two labels, so a node contains at most $2n$ i.e. $O(n)$ labelled formulae. Hence there are at most $(2^{O(n)})^{O(n)} = 2^{O(n^2)}$ different nodes. Without the rule $(\cup)$, there are at most $2^{2^{O(n)}}$ different nodes, which breaks EXPTIME worst-case complexity, so the $(\cup)$ rule is absolutely essential. But it is not necessary for the satisfiability problem of a fixed logic.

Algorithm 1 terminates in general because it checks for repeated ancestors: this check is built into the definition of an end-node, and also in Step 2(a)ii. Thus the same node can appear on multiple branches. In the worst case, Algorithm 1 therefore requires $2^{2^{O(n^2)}}$ time. We therefore refine it into Algorithm 2 below:

Algorithm 2
Input: a finite set X of formulae in the primitive language.
Output: a finite graph $G = (V, E)$

1. Let $G =< V, E >=< \{X\}, \emptyset >$, and mark X as unresolved.
2. While V contains unresolved nodes, take one, say n, and do:
   (a) If $(\cup)$ is applicable to n then apply it to obtain denominator d_1
   (b) Else if any static rule of $\mathcal{C}L$ is applicable to n then apply it to obtain denominator(s) d_1 (and possibly d_2)
   (c) Else, for every formula $\langle t\rangle\varphi_i$ in n, let $d_i = \{(A_s, \delta_s(Q,t)){:}\psi_i \ s.t.\ (A_s, Q){:}\ \psi_i \in n\} \cup \{\varphi_i\}$ be the denominator obtained by applying (trans) to n
   (d) Mark n as resolved (n is an or/and node if the applied rule is/isn't $(\vee)$)
   (e) For every denominator $d = d_1, \cdots, d_k$:
      i. If some proxy $c \in V$ has $c = d$, then add the edge (n, c) to E
      ii. Else add d to V, add (n, d) to E, and mark d as unresolved.

Algorithm 2 builds an and-or graph G monotonically by "caching" previously seen nodes (but not their open/closed status). The graph G contains a node d for *every* applicable static rule denominator, not just their $\mathcal{C}L$-saturation as in Algorithm 1. Each node appears only once because repetitions are represented by "cross-tree" edges to their first occurrence, so G has at most $2^{O(n^2)}$ nodes.

We now make passes of the and-or graph G, marking nodes as *false* in a monotonic way. In the first pass we mark the node containing $\bot$, if it exists, since *false* captures inconsistency. In each subsequent pass we mark any unmarked or-node with two *false*-marked children, and mark any unmarked and-node with at least one *false*-marked child. We stop making passes when some pass marks no node. Otherwise, we must terminate after $2^{O(n^2)}$ passes since the root must then be marked with *false*. Note that once a node is marked with *false* this mark is never erased. Finally, mark all non-*false* nodes with *true* giving graph G_f.

Lemma 7. *If node $n \in G_f$ is marked* false *then n is $\mathcal{C}L$u-inconsistent.*

Proof. By induction on the number of passes needed to mark n with *false*.

Lemma 8. *If a node $n \in G_f$ is marked* true *then it is $\mathcal{C}L$u-consistent.*

Proof. An easy proof is to take the sub-graph G_n generated by n; replace each sequence of *true*-marked static rule denominators by one *true*-marked node containing their union, which represents their $\mathcal{C}L$u-saturation; and turn the resulting sub-graph into an L-frame by appropriately extending it as in Step 3 of Algorithm 1. For each node x, putting $p \in h(x)$ iff $p \in x$ gives an L-model for n since: all eventualities in a *true*-marked node are fulfilled by its children, and these are guaranteed to be marked *true*; and each *true*-marked or-node has at least one *true*-marked child. By the completeness of $\mathcal{C}L$u, every $\mathcal{C}L$u-tableau for n must be open. A slightly trickier proof converts G_n into an and-or *tree* by mimicking the rule applications from G_n but unwinding edges to non-ancestor-proxies by making a copy of the proxy. This reproduces all $\mathcal{C}L$u-tableaux for n constructible by Algorithm 1 (*sic*) and each one is open by its construction.

Algorithm 2 and the creation of G_f runs in time $(2^{O(n^2)})^2 = 2^{2.O(n^2)}$ and so the general satisfiability problem of regular grammar logics is in EXPTIME.

$$(I\bot)\ Z \stackrel{\top}{\frown} Z'; p; \neg p \qquad (I\bot)\ Z; p \stackrel{p}{\frown} Z'; \neg p$$

$$(I\wedge)\ \frac{Z \stackrel{\zeta}{\frown} Z'; \varphi \wedge \psi}{Z \stackrel{\zeta}{\frown} Z'; \varphi; \psi} \qquad (I\vee)\ \frac{Z \stackrel{\zeta \wedge \xi}{\frown} Z'; \varphi \vee \psi}{Z \stackrel{\zeta}{\frown} Z'; \varphi \mid Z \stackrel{\xi}{\frown} Z'; \psi}$$

$$(I\mathsf{label})\ \frac{Z \stackrel{\zeta}{\frown} Z'; [t]\varphi}{Z \stackrel{\zeta}{\frown} Z'; (A_t, I_t){:}\varphi} \qquad (I\mathsf{add})\ \frac{Z \stackrel{\zeta}{\frown} Z'; (A_t, Q){:}\varphi}{Z \stackrel{\zeta}{\frown} Z'; (A_t, Q){:}\varphi; \varphi}\ \text{if } Q \cap F_t \neq \emptyset$$

$$(I\mathsf{trans})\ \frac{Z \stackrel{[t]\zeta}{\frown} Z'; \langle t\rangle\varphi}{\{(A_s, \delta_s(Q,t)){:}\psi \text{ s.t. } (A_s, Q){:}\psi \in Z\} \stackrel{\zeta}{\frown} \{(A_s, \delta_s(Q,t)){:}\psi \text{ s.t. } (A_s, Q){:}\psi \in Z'\}; \varphi}$$

Fig. 2. Rules of the Calculus for Interpolation

6 Effective Interpolation

We say that ζ is an *interpolation formula in L for the formula $\varphi \supset \psi$* if all primitive propositions of ζ are common to φ and ψ, and $\varphi \supset \zeta$ and $\zeta \supset \psi$ are both L-valid. The Craig interpolation lemma for L states that if $\varphi \supset \psi$ is L-valid, then there exists an interpolation formula in L for $\varphi \supset \psi$. This lemma is effective if the proof of the lemma actually constructs the interpolation formula.

Assume our language contains $\top$ with the usual semantics. We prove effective Craig interpolation for all regular grammar logics using the method of [16].

Our tableau calculi are refutation calculi, so we use an indirect formulation of interpolation. Given two sets X and Y of formulae, and using $\overline{\zeta}$ to denote the negation normal form of $\neg\zeta$, we say that ζ is an interpolation formula w.r.t. $\mathcal{C}L$ for the pair $\langle X, Y\rangle$, and also that $X \stackrel{\zeta}{\frown} Y$ *is $\mathcal{C}L$-valid*, if: all primitive propositions of ζ are common to X and Y, the formula ζ does not contain automaton-labelled formulae, and the sets $X; \overline{\zeta}$ and $\zeta; Y$ are both $\mathcal{C}L$-inconsistent. Since $\mathcal{C}L$ is sound and complete, it follows that if $\varphi \stackrel{\zeta}{\frown} \overline{\psi}$ is $\mathcal{C}L$-valid, then $\varphi \supset \zeta$ and $\zeta \supset \psi$ are both L-valid, and hence that ζ is an interpolation formula in L for $\varphi \supset \psi$.

We now show that for any finite formula sets X and Y, if $X; Y$ is $\mathcal{C}L$-inconsistent, then there exists an interpolation formula w.r.t. $\mathcal{C}L$ for the pair $\langle X, Y\rangle$. It follows that the Craig interpolation lemma holds for L.

Observe that $Y \stackrel{\overline{\varphi}}{\frown} X$ is $\mathcal{C}L$-valid iff $X \stackrel{\varphi}{\frown} Y$ is $\mathcal{C}L$-valid. We call $Y \stackrel{\overline{\varphi}}{\frown} X$ the *reverse form* of $X \stackrel{\varphi}{\frown} Y$.

The rule $(I\delta)$ below left is an *interpolation rule* if the inference step below right is an instance of the tableau rule with name (δ):

$$(I\delta)\ \frac{N \stackrel{\varphi}{\frown} N'}{D_1 \stackrel{\varphi_1}{\frown} D_1' \mid \ldots \mid D_k \stackrel{\varphi_k}{\frown} D_k'} \qquad \frac{N; N'}{D_1; D_1' \mid \ldots \mid D_k; D_k'}$$

Provided that (δ) is a $\mathcal{C}L$-tableau rule, the interpolation rule $(I\delta)$ is *$\mathcal{C}L$-sound* if $\mathcal{C}L$-validity of all $D_1 \stackrel{\varphi_1}{\frown} D_1', \ldots, D_k \stackrel{\varphi_k}{\frown} D_k'$ implies $\mathcal{C}L$-validity of $N \stackrel{\varphi}{\frown} N'$.

Figure 2 contains the interpolation rules obtained from the tableau rules for regular grammar logics. Each tableau rule of $\mathcal{CL}$ except $(I\bot)$ has one corresponding interpolation rule. Rule $(\bot)$ has an interpolation rule for each of its two principal formulae but these rules have no denominator because it is not necessary. Rule $(\cup)$ has no interpolation rule since it is just an optimisation rule.

Lemma 9. *The above interpolation rules are $\mathcal{CL}$-sound.*

Proof. We consider $(I\mathsf{trans})$ only, the others are similar. Let $X = \{(A_s, \delta_s(Q,t)){:}\,\psi \ s.t.\ (A_s, Q){:}\,\psi \in Z\}$ and $Y = \{(A_s, \delta_s(Q,t)){:}\,\psi \ s.t.\ (A_s, Q){:}\,\psi \in Z'\}$. Suppose that $X \overset{\zeta}{\frown} Y; \varphi$ is $\mathcal{CL}$-valid. Thus, both $X; \overline{\zeta}$ and $\zeta; Y; \varphi$ are $\mathcal{CL}$-inconsistent, and have closed $\mathcal{CL}$-tableaux. We show that $Z \overset{[t]\zeta}{\frown} Z'; \langle t\rangle\varphi$ is $\mathcal{CL}$-valid by giving closed $\mathcal{CL}$-tableaux for both $Z; \langle t\rangle\overline{\zeta}$ and $[t]\zeta; Z'; \langle t\rangle\varphi$:

$$\dfrac{\dfrac{Z; \langle t\rangle\overline{\zeta}}{X; \overline{\zeta}}\,(\mathsf{trans})}{\bot}\,(\text{assumption}) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[t]\zeta; Z'; \langle t\rangle\varphi}{(A_t, I_t){:}\,\zeta; Z'; \langle t\rangle\varphi}\,(\mathsf{label})}{(A_t, \delta_t(I_t,t)){:}\,\zeta; Y; \varphi}\,(\mathsf{trans})}{(A_t, \delta_t(I_t,t)){:}\,\zeta; \zeta; Y; \varphi}\,(\mathsf{add})}{\zeta; Y; \varphi}\,(\text{wk})}{\bot}\,(\text{assumption})$$

Applying (add) above right is justified because $\delta_t(I_t, t) \cap F_t \neq \emptyset$ since A_t accepts word t. Also, the rule (wk) of weakening is obviously admissible.

These rules build the numerator's interpolant from those of the denominators. Using Lemma 9, and the technique of [16, Lemmas 13 and 14] we obtain:

Theorem 2. *Regular grammar logics enjoy effective Craig interpolation.*

7 Further Work and Conclusions

Our main contribution is a tableau calculus that forms a decision procedure for the whole class of regular grammar logics. Our automaton-labelled formulae are similar to formulae of APDL [10], but with a more compact representation using sets of states instead of single states. We have shown that automaton-labelled formulae work well with the traditional techniques of proving soundness and completeness. Our calculus gives a simple estimate of the upper complexity bound of regular grammar logics, and can be used to obtain effective Craig interpolation for these logics. We have since found that Craig interpolation for regular grammar logics follows from [13, Corollary B4.1] and [12].

The prefixed tableaux of Baldoni *et al.* give a decision procedure only for right linear logics. A prefixed calculus that simulates our calculus would be less efficient because it would repeatedly search the current branch for computation, not just for loops as in our case. Moreover, it is well-known that loop checking can be done efficiently using, e.g., a hash table. Finally, the transformation of

Demri and de Nivelle into GF^2 is based on states, but not sets of states, which reduces efficiency. Also their resulting formula sets are much larger because they keep a copy of the formulae defining an automaton A_t for each formula $[t]\varphi$, whereas we can keep only t and Q for (A_t, Q) in $(A_t, Q){:}\varphi$. Similar observations have been stated for formulae of APDL.

By propagating *false* "on the fly", we believe we can prove global caching sound for checking satisfiability in multimodal **K** with global assumptions i.e. "checking $\mathcal{ALC}$-satisfiability of a concept w.r.t. a TBox with general axioms" [6].

References

1. F. Baader and U. Sattler. An overview of tableau algorithms for description logics. *Studia Logica*, 69:5–40, 2001.
2. M. Baldoni. *Normal Multimodal Logics: Automatic Deduction and Logic Programming Extension*. PhD thesis, Dip. di Inf., Univ. degli Studi di Torino, Italy, 1998.
3. M. Baldoni, L. Giordano, and A. Martelli. A tableau for multimodal logics and some (un)decidability results. In *TABLEAUX'1998, LNCS 1397:44-59*, 1998.
4. S. Demri. The complexity of regularity in grammar logics and related modal logics. *Journal of Logic and Computation*, 11(6):933–960, 2001 (see also the long version).
5. S. Demri and H. de Nivelle. Deciding regular grammar logics with converse through first-order logic. *Journal of Logic, Language and Information*, 2005. To appear.
6. F Donini and F Massacci. EXPTIME tableaux for $\mathcal{ALC}$. *Artificial Intelligence*, 124:87–138, 2000.
7. R Fagin, J Y Halpern, Y Moses, and M Y Vardi. *Reasoning About Knowledge*. MIT Press, 1995.
8. L. Fariñas del Cerro and M. Penttonen. Grammar logics. *Logique et Analyse*, 121-122:123–134, 1988.
9. R. Goré. Tableau methods for modal and temporal logics. In D'Agostino et al, editor, *Handbook of Tableau Methods*, pages 297–396. Kluwer, 1999.
10. D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. MIT Press, 2000.
11. I. Horrocks and U. Sattler. Decidability of SHIQ with complex role inclusion axioms. *Artificial Intelligence*, 160(1-2):79–104, 2004.
12. M. Kracht. Reducing modal consequence relations. *JLC*, 11(6):879–907, 2001.
13. M. Marx and Y. Venema. *Multi-dimensional Modal Logic*. Kluwer, 1997.
14. A. Mateescu and A. Salomaa. Formal languages: an introduction and a synopsis. In *Handbook of Formal Languages - Volume 1*, pages 1–40. Springer, 1997.
15. J.-J.Ch. Meyer and W. van der Hoek. *Epistemic Logic for Computer Science and Artificial Intelligence*. Cambridge University Press, 1995.
16. L.A. Nguyen. Analytic tableau systems and interpolation for the modal logics KB, KDB, K5, KD5. *Studia Logica*, 69(1):41–57, 2001.
17. L.A. Nguyen. Analytic tableau systems for propositional bimodal logics of knowledge and belief. In *TABLEAUX 2002, LNAI 2381:206-220*. Springer, 2002.
18. W. Rautenberg. Modal tableau calculi and interpolation. *JPL*, 12:403–423, 1983.
19. J. van Benthem. Correspondence theory. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic, Vol II*, pages 167–247. Reidel, Dordrecht, 1984.
20. M. Wessel. Obstacles on the way to qualitative spatial reasoning with description logics: Some undecidability results. In *Description Logics 2001*.

Appendix: an example

Example 2. Let $\mathcal{MOD} = \{0, 1, 2\}$. Consider the grammar logic L with the inclusion axioms $[0]\varphi \supset \varphi$ and $[i]\varphi \supset [j][k]\varphi$ if $i = (j + k) \bmod 3$. This is a regular grammar logic because the corresponding grammar is regular. We have $A_i = \langle \mathcal{MOD}, \mathcal{MOD}, \{0\}, \delta, \{i\}\rangle$ for $i \in \mathcal{MOD}$, where $\delta = \{(j, k, l) \mid j, k, l \in \{0, 1, 2\} \text{ and } l = (j + k) \bmod 3\}$.

We give a closed $\mathcal{C}L$-tableau for $X = \{\langle 0\rangle p, [0](\neg p \vee \langle 1\rangle q), [1](\neg q \vee \langle 2\rangle r), [0]\neg r\}$, in which principal formulae of nodes are underlined. The arrows stand for rule applications and are annotated with the rule name. The labels R_i for $i \in \{0, 1, 2\}$ to the right of the arrows marked with (trans)-rule applications stand for the label on the associated edges in the underlying model being explored by the tableau.