A tableau calculus for temporal description logic: the expanding domain case.

Holger Sturm and Frank Wolter
Institut für Informatik, Universität Leipzig
Augustus-Platz 10-11, 04109 Leipzig, Germany

October 17, 2000

Abstract. In this paper we present a tableau calculus for a temporal extension of the description logic $\mathcal{ALC}$, called $\mathcal{TL}_{\mathcal{ALC}}$. This logic is based on the temporal language with 'Until' interpreted over the natural numbers with expanding $\mathcal{ALC}$-domains. The tableau calculus forms an elaborate combination of Wolper's tableau calculus for propositional linear temporal logic, the standard tableau-algorithm for $\mathcal{ALC}$, and the method of quasi-models as it has been introduced by Wolter and Zakharyaschev. Based on those three ingredients the paper provides a new method of how tableau-based decision procedures can be constructed for many-dimensional logics which lack the finite model property. The method can be applied to deal with other temporalized formalisms as well.

1 Introduction

In many application domains of logic in Computer Science and Artificial Intelligence it is no longer enough to describe the static aspect of the world. In particular, there is a need to formalize its temporal evolution as well. Consequently, over the last years we have observed a growing interest in so-called temporalizations of logical formalisms. We mention [14, 15] for a general approach, [13, 34] for extensions of epistemic logics by means of temporal operators to model, among others, multi-agent systems, [33] which supplies a logic of space with temporal operators in order to describe the evolution of regions in time, [1, 12] for work on temporal databases, and [20, 23, 24, 28] which investigate temporal logic based on (fragments of) first-order logic. Particularly many researchers are concerned with temporalizations of description logics. Recall that description logics like $\mathcal{ALC}$ were designed to represent and reason about static and time-independent domains.[1] They have been applied, for example, in the framework of information systems [11], databases [7, 9, 10], and software engineering [35]. Temporalizations of this static formalism have been suggested and investigated in [2, 4, 5, 6, 25, 26, 32], consult also [3] for a survey.

While on the syntactic side a temporalization adds temporal operators to the original vocabulary, on the semantic side it adds a new, temporal, dimension to the domain interpreting the original formalism. So, in a sense, it is rather easy to temporalize a given language – just add temporal operators and a flow of time interpreting them. However, it is known that often the addition of a new dimension to a simple and well understood formalism drastically

[1]The reader can consult the special issue on Description Logic in the Journal of Logic an Computation (9), 1999, for an overview of recent work in the field.

increases its computational complexity, see [15, 19, 20, 23, 24, 27, 31]. In fact, apparently very small differences in the design of temporalizations can have unexpected drastic consequences when the computational complexity or implementability of the resulting system is taken into account. While, on the one hand, considerable progress has been made as concerns the determination of the decidability and complexity of temporalized systems, witness [19, 24, 25, 32], on the other hand, only very few "implementable" algorithms — i.e., tableaux or resolution algorithms — have been suggested. Laudable exceptions are the work of [2, 5, 25, 34].

In this paper we follow the latter line of research and develop a tableau-algorithm for the extension of $\mathcal{ALC}$ introduced in [32] interpreted, however, not in models with constant domain but in models with increasing domains.[2] That is to say, we extend the language of $\mathcal{ALC}$ by means of the temporal operator $\mathcal{U}$ ('Until') which can be applied to concepts C, D and formulas (alias axioms) φ, ψ to form the new concept $C\mathcal{U}D$ and the new formula $\varphi\mathcal{U}\psi$. The language is interpreted in models based on the natural numbers $\mathbb{N}$ in which every n comes equipped with an $\mathcal{ALC}$-model describing the domain at moment n. It is assumed that the domain of the $\mathcal{ALC}$-model at moment n is included in the domains at every moment following n. The idea behind this formalism is that $\mathcal{ALC}$ represents the knowledge about the states of a given process while the temporal component describes the behaviour of the process in time, i.e., the sequence of states. We note that it is proved in [32] that the satisfiability problem for the language we consider here is decidable; so, of course, our primary concern in this paper is *not* decidability but the construction of a tableau-based and (in principle) implementable decision procedure.

Similar to the approaches in [25, 34] our tableau-algorithm is based on both the standard tableau for $\mathcal{ALC}$ (see [8, 21]) and the standard tableau for propositional temporal logic developed in [29], see also [17]. In contrast to those tableaux, however, in our case we cannot simply take the "join" of the two known tableaux but because of the interaction between $\mathcal{ALC}$ and 'Until' we have to merge the two algorithms in a rather subtle manner. This difference is reflected by the fact that the logic we consider does not have the bounded domain property — the $\mathcal{ALC}$-domains required to satisfy certain formulas grow arbitrary large. A consequence of this is another unusual feature of our tableau: complete tableaux here do not directly induce satisfying models but only so-called quasimodels (in which the possibly infinite $\mathcal{ALC}$-domains are encoded by constraint systems). Quasimodels have been first introduced in [30, 32] to prove the decidability of certain temporalized (or modal) description logics. They have been applied as well to investigate first-order temporal logics [20]. In contrast to the quasimodels there, in this paper the quasiworlds which build up the quasimodels are not defined semantically but syntactically as constraint systems which are closed under certain constraint rules.

We should like to note that the method of temporalizing tableaux as it is developed in the present paper does not only apply to $\mathcal{ALC}$ but to various other formalisms like the epistemic polymodal logics (for n agents) $\mathbf{K}_n$, $\mathbf{S4}_n$, and $\mathbf{S5}_n$ (see [18]), or more expressive description languages than $\mathcal{ALC}$ like $\mathcal{ALC}$ with inverse roles, see [22].

The paper has the following structure: In Section 2 we describe the syntax and semantics of $\mathcal{TL}_{\mathcal{ALC}}$, and show that $\mathcal{TL}_{\mathcal{ALC}}$ lacks the bounded domain property. Constraint systems and related notions will be defined in Section 3. After having introduced the notion of a

[2]It is easy to see that models with varying domains can be simulated by means of models with expanding domains (see e.g. [32]), so our tableau applies to the varying domain case as well. The constant domain case is more difficult and requires the addition of a new tableau rule and more involved completeness proofs. It will be investigated in a forthcoming paper.

quasimodel, we prove in Section 4 that for $\mathcal{TL}_{\mathcal{ALC}}$-formulas to be satisfiable (in the usual sense) and having a quasimodel turn out to be equivalent things. Section 5 is reserved for describing the tableau rules. The main notions with respect to tableaux will then be introduced in Section 6. In Section 7 we prove the calculus to be correct, whereas completeness and decidability form the topic of Section 8. In order to illustrate how the tableau works, we give a concrete example in Section 9. In Section 10 we compare our results with work done by others. The paper closes with some suggestions concerning future research in Section 11.

Acknowledgements. We would like to thank two anonymous referees as well as Carsten Lutz and Michael Zakharyaschev for comments on earlier versions of this paper. Work on this paper was supported by the DFG project Wo582/3-1.

2 Basic Definitions

The language of the logic $\mathcal{TL}_{\mathcal{ALC}}$ is based on three sets,

- a set of concept names $C_0, C_1, \ldots$;
- a set of role names $R_0, R_1, \ldots$;
- and a set of object names $a_0, a_1, \ldots$.

The set of concepts is defined inductively as follows:

- the concept names C_i and the logical constant $\top$ are concepts,
- if C, D are concepts and R is a role name, then $C \sqcap D$, $\neg C$, $\exists R.C$, $\bigcirc C$, and $C\mathcal{U}D$ are concepts.

The set of formulas is then defined as follows:

- if C, D are concepts and a is an object name, then $C = D$ and $a : C$ are (atomic) formulas,
- if φ and ψ are formulas, so are $\neg\varphi$, $\varphi \wedge \psi$, $\bigcirc\varphi$ and $\varphi\mathcal{U}\psi$.

Unlike the languages that were considered in [30, 32], our language does not contain role instantiations of the form aRb. It is rather straightforward, however, to find an extension of the tableau calculus to be developed in this paper which covers languages with role instantiations as well. Since the more general case would just add some more notation and rules, without giving new insights, we leave this as an exercise to the reader.

The following syntactic definitions will be needed throughout the paper: Let Φ be a set of formulas. $Sb(\Phi)$ designates the set of subformulas of formulas in Φ. By $ob(\Phi)$ we mean the set of all object names that occur in Φ, and by $con(\Phi)$ the set of all concepts in Φ. Moreover, let $\#$ be some unary operator, say $\neg$ or $\bigcirc$. Then $\#(\Phi)$ is defined as the union of the two sets Φ and $\{\#\varphi \,|\, \varphi \in \Phi\}$. The closure of Φ is defined as $\bigcirc(\neg Sb(\Phi \cup \{\top\}))$. In a similar way we define the closure of a set of concepts. The fragment generated by Φ, in symbols $Fg(\Phi)$, is defined as the union of the following three sets: $ob(\Phi)$, $\bigcirc(\neg con(\Phi \cup \{\top\}))$ and $\bigcirc(\neg Sb(\Phi \cup \{\top\}))$. We denote by $\sharp_c(\Phi)$ and $\sharp_f(\Phi)$ the cardinalities of $\bigcirc(\neg con(\Phi \cup \{\top\}))$ and $\bigcirc(\neg Sb(\Phi \cup \{\top\}))$, respectively.

Definition 2.1 *A $\mathcal{TL}_{\mathcal{ALC}}$-model is a triple $\mathfrak{M} = (\mathbb{N}, <, I)$, where $(\mathbb{N}, <)$ is the set of natural numbers equipped with the usual strict order and I is a function associating with each $n \in \mathbb{N}$ some $\mathcal{ALC}$-model*

$$I(n) = (\Delta^n, R_0^{I,n}, \dots, C_0^{I,n}, \dots, a_0^{I,n}, \dots),$$

in which $R_i^{I,n}$ are binary relations on Δ^n, $C_i^{I,n}$ are subsets of Δ^n, and $a_i^{I,n} \in \Delta^{I,n}$ such that $a_i^{I,n} = a_i^{I,m}$, for every $n, m \in \mathbb{N}$, and $\Delta^n \subseteq \Delta^m$, for every $n, m \in \mathbb{N}$ with $n < m$.

The extension of a concept C in a model $\mathfrak{M}$ at a moment n, in symbols $C^{I,n}$, is defined as usual:

$$\top^{I,n} = \Delta^n;$$

$$(C \sqcap D)^{I,n} = C^{I,n} \cap D^{I,n};$$

$$(\neg C)^{I,n} = \Delta^n \setminus C^{I,n};$$

$$(\exists R_i.C)^{I,n} = \{d \in \Delta^n \mid \exists d' \in C^{I,n} : dR_i^{I,n}d'\};$$

$$(C\mathcal{U}D)^{I,n} = \{d \in \Delta^n \mid \exists m \geq n(d \in D^{I,m} \,\&\, \forall k(n \leq k < m \Rightarrow d \in C^{I,k}))\};$$

$$(\bigcirc C)^{I,n} = \{d \in \Delta^n \mid d \in C^{I,n+1}\}.$$

Observe that the operator $\mathcal{U}$ is here interpreted by means of the relation $\leq$.

The satisfaction-relation $\models$ between pairs $(\mathfrak{M}, n)$, $n \in \mathbb{N}$, and formulas φ is defined in the standard manner as well:

$\mathfrak{M}, n \models \top$;

$\mathfrak{M}, n \models C = D$ iff $C^{I,n} = D^{I,n}$;

$\mathfrak{M}, n \models a : C$ iff $a^{I,n} \in C^{I,n}$;

$\mathfrak{M}, n \models \neg\varphi$ iff $\mathfrak{M}, n \not\models \varphi$;

$\mathfrak{M}, n \models \varphi \wedge \psi$ iff $\mathfrak{M}, n \models \varphi$ and $\mathfrak{M}, n \models \psi$;

$\mathfrak{M}, n \models \varphi\mathcal{U}\psi$ iff
there is some $m \geq n$ with $\mathfrak{M}, m \models \psi$ such that for all k with $n \leq k < m$, $\mathfrak{M}, k \models \varphi$;

$\mathfrak{M}, n \models \bigcirc\varphi$ iff $\mathfrak{M}, n+1 \models \varphi$.

A formula φ is called *satisfiable* iff there exists a model $\mathfrak{M}$ and a moment $n \in N$ such that $\mathfrak{M}, n \models \varphi$. We can always assume that n equals 0. Let us relate the notion of formula satisfiability introduced here to the standard reasoning problems in description logic: A concept C is said to be satisfiable if there exists a model $(\mathbb{N}, <, I)$ and $n \in \mathbb{N}$ with $C^{I,n} \neq \emptyset$. Obviously, C is satisfiable iff the formula $\neg(\neg C = \top)$ is satisfiable. Recall that a TBox is a set of concept equations. Now, a concept C is said to be satisfiable relative to a TBox T if there exists a model $(\mathbb{N}, <, I)$ and $n \in \mathbb{N}$ such that $\mathfrak{M}, n \models T$ and $C^{I,n} \neq \emptyset$. Clearly, C is satisfiable relative to T iff the formula

$$(\bigwedge_{D_1 = D_2 \in T} D_1 = D_2) \wedge \neg(\neg C = \top)$$

is satisfiable. Thus, concept satisfiability relative to TBoxes is reducible to formula satisfiability. In a similar manner, subsumption, subsumption relative to TBoxes, and instance checking can be reduced to formula satisfiability.

Without loss of generality we can assume that every equality is of the form $C = \top$. For an equality $C = D$ in which D is different from $\top$ can be transformed into the equality $(\neg C \sqcup D) \sqcap (\neg D \sqcup C) = \top$ which is equivalent to the former one. Often we shall write $C \neq \top$ instead of $\neg(C = \top)$.

The first lemma in this paper tells us that $\mathcal{TL}_{\mathcal{ALC}}$ lacks the *bounded domain property* in the following sense: There exist formulas φ which are satisfiable, but not in a model with bounded domain. Here a model $\mathfrak{M}$ is said to have bounded domain iff there exists $n \in \mathbb{N}$ such that $|\Delta^m| \leq n$, for all $m \in \mathbb{N}$.

To show the existence of such formulas, we consider an example similar to one introduced in [30]. Abbreviate $\Box C = \neg(\top \mathcal{U} \neg C)$, $\Diamond C = \top \mathcal{U} C$, $\Box\varphi = \neg(\top\mathcal{U}\neg\varphi)$, $\Diamond\varphi = \top\mathcal{U}\varphi$, and let

- $\varphi_1 = ((\neg(C \sqcap \neg\Box C)) = \top)$,
- $\varphi_2 = (\exists R.\neg C = \top)$,
- $\varphi_3 = (\Diamond C = \top)$,
- $\varphi_4 = (\neg C \neq \top)$.

Finally, put $\xi = (\Box\varphi_1 \wedge \Box\varphi_2 \wedge \Box\varphi_3 \wedge \varphi_4)$.

Lemma 2.2 *ξ is satisfiable; but not in a model with bounded domain.*

Proof. We first show that ξ is not satisfiable in a model with bounded domain. Suppose $\mathfrak{M} = (\mathbb{N}, <, I)$ satisfies ξ in 0. By φ_4 and $\Box\varphi_2$ we find $c_0, c_1 \in \Delta^0$ with $c_0 \in C^{I,0}$ and $c_1 \notin C^{I,0}$. By $\Box\varphi_3$ there exists $m > 0$ such that $c_1 \in C^{I,m}$. With the help of φ_2 we find $c_2 \in \Delta^m$ such that $c_2 \notin C^{I,m}$. $c_2 \neq c_0$, since $c_0 \in C^{I,m}$ by φ_1. In this way we find infinite sequences m_i, $i \in \mathbb{N}$, and c_i, $i \in \mathbb{N}$, such that $c_i, c_{i+1} \in \Delta^{m_i}$, $c_i \in C^{I,m_i}$, $c_{i+1} \in (\neg C)^{I,m_i}$, and the c_i, $i \in \mathbb{N}$, are pairwise distinct. This proves the first part. The second part, that ξ is satisfiable, is left to the reader. However, at the end of Section 8 we will give a proof by means of our tableau method. $\dashv$

That $\mathcal{TL}_{\mathcal{ALC}}$ does not have the bounded domain property is a serious problem for every attempt to find a decision procedure based on tableaux. For, intuitively, in a finite tableau only a finite, hence bounded number of individuals is introduced, and, moreover, a decision procedure requires that for each formula the procedure terminates. However, this difficulty can be overcome by making use of the method of quasimodels. For an intuitive sketch of the main idea, see the beginning of Section 4.

3 Constraint systems

As has been emphasized in the introduction, the main purpose of this paper is to provide a tableau based procedure for deciding the satisfiability of $\mathcal{TL}_{\mathcal{ALC}}$-formulas. Following common practice in description logic (see e.g. [8, 21, 22]), the tableau rules operate on constraint systems, where a constraint system is nothing but a finite set of formulas taken from a special language L_C extending $\mathcal{TL}_{\mathcal{ALC}}$ by means of variables (for a precise definition see below). From a decision procedure for the "satisfiability" of constraint systems we immediately obtain a decision procedure for the satisfiability of $\mathcal{TL}_{\mathcal{ALC}}$-formulas.

To begin the formal presentation of constraint systems, we first have to introduce the language L_C from which constraint systems are build up.

Definition 3.1 *The constraint language L_C: Let V be some fixed countable set of (individual) variables. We assume V to be disjoint from the set of object names (from the language of $\mathcal{TL}_{\mathcal{ALC}}$). The variables together with the object names form the set of L_C-terms. The set of L_C-formulas is defined by means of the following clause: If φ is some $\mathcal{TL}_{\mathcal{ALC}}$-formula, C some concept, and x some L_C-term, then φ and $x : C$ are L_C-formulas.*

For the following it will be useful to assume that V comes equipped with a well-order, abbreviated by $<_V$. Let X be a non-empty subset of V. Then $min(X)$ is defined as the first variable from X with respect to $<_V$.

Definition 3.2 *A constraint system S is defined as a finite (non-empty) set of L_C-formulas. For convenience sake, we assume that the formula $min(V) : \top$ is contained in every constraint system.*

A special class of constraint systems, so-called saturated ones, will be of great importance throughout the whole paper. This class consists of all constraint systems which satisfy a number of closure conditions with respect to formulas. With a few exceptions, these conditions require that if S contains a formula φ of a certain syntactical shape, then S contains certain formulas, which, in most cases, are (negations of) subformulas of φ respectively, in case $\varphi \doteq x : C$, formulas of the form $x : (\neg)D$, where D is some subconcept of C. Which formulas exactly depends on the syntactical shape of φ. To give an example, consider the rule for conjunction: S is said to be closed under conjunctions if whenever S contains a formula φ of the form $\psi_1 \wedge \psi_2$, then S contains the two conjuncts ψ_1 and ψ_2 as well. At the end of this section the reader will realize that some of our rules are more complicated (see, for instance, the locally generating rules).

For our purposes it will prove useful to formulate such closure conditions explicitly by means of the *saturation rules* listed in Figures 1-3 below. The set of all saturation rules is denoted by $\mathcal{R}$. A constraint system S is called saturated if none of these rules can be applied to it. To make this more precise, we make use of several auxiliary notions of saturation.

Definition 3.3 *Let S be a constraint system and x be some L_C-term which occurs in S. Then*

- *S is said to be f-saturated iff none of the saturation rules operating on formulas is applicable to S.*
- *A term x is locally saturated in S iff no saturation rule operating on concepts can be applied in S to any L_C-formula of the form $x : C$.*
- *A constraint system S is saturated iff S is f-saturated, each L_C-term that occurs in S is locally saturated in S, and no locally generating rule can be applied to S.*
- *A constraint system S is clash-free iff it neither contains any pair of the form $x : C, x : \neg C$ nor any pair of the form $\varphi, \neg\varphi$ nor formulas of the form $\neg\top$ respectively $x : \neg\top$.*

We write $S \longrightarrow S'$ to say that the constraint system S' can be obtained from S by an application of the saturation rule $\longrightarrow$.

Later in this paper most of the saturation rules will serve as tableau rules. When building up the tableau we will follow a certain strategy.

Definition 3.4 *Let $S_0, \ldots, S_n$ be a sequence of constraint systems such that, for every $i < n$, there exists a saturation rule $\longrightarrow$ such that $S_i \longrightarrow S_{i+1}$ and if the rule is a locally generating rule then no non-generating rule is applicable to S_i. Then $S_0, \ldots, S_n$ is said to be built according to the saturation strategy.*

The following lemma is required later when we are concerned with the termination of our tableau procedure.

Lemma 3.5 *Let $S_0, \ldots, S_n$ be a sequence built according to the saturation strategy and suppose $S_0 = \{\vartheta\}$ for some formula ϑ. Then $|S_n| \leq \sharp_f(\vartheta) + 2^{\sharp_c(\vartheta)} \cdot \sharp_c(\vartheta) + |ob(\vartheta)| \cdot \sharp_c(\vartheta)$.*

Proof. We have $S_i \subset S_{i+1}$, for every $i \leq n$. So it suffices to show that the cardinality of S_n is bounded by $\sharp_f(\vartheta) + 2^{\sharp_c(\vartheta)} \cdot \sharp_c(\vartheta) + |ob(\vartheta)| \cdot \sharp_c(\vartheta)$. Obviously it suffices to show that the number of variables in S_n is bounded by $2^{\sharp_c(\vartheta)}$. And to this end it is enough to show that we cannot find different variables v, v' such that

$$\{C \mid (v : C) \in S_n\} = \{C \mid (v' : C) \in S_n\}.$$

Assume, to the contrary, that v, v' with those properties exist. We may assume there exists S_j, $j < n$, such that v occurs in S_j and a locally generating rule applied to S_j introduces v'. If the generating rule is $\longrightarrow_\exists$, then

$$\{C \mid (v : C) \in S_n\} = \{C \mid (v : C) \in S_j\} \not\supseteq \{C \mid (v' : C) \in S_j\} \subseteq \{C \mid (v' : C) \in S_n\}$$

which contradicts the assumption concerning v, v'. If the generating rule is $\longrightarrow_{\neq}$, then we find a concept C such that $(v : \neg C) \notin S_j$ but $(v' : \neg C) \in S_n$. Again we have a contradiction since $(v : \neg C) \notin S_n$ follows from $(v : \neg C) \notin S_j$. Remember that $S_0, \ldots, S_n$ is build according to the strategy fixed in Definition 3.4. $\dashv$

4 Quasimodels

In Section 2 we have shown that $\mathcal{TL}_{\mathcal{ALC}}$ lacks the bounded domain property. This seems to prohibit tableau-based decision procedures from the scratch. However, by a closer inspection of the situation things turn out to be less hopeless: Though we need infinitely many individuals in order to satisfy a formula like ξ, those individuals only fall into a *finite* number of classes with respect to which concepts, from the fixed fragment $Fg(\xi)$, they satisfy. In other words, a finite number of concept types might be sufficient to represent an infinite domain.

In order to describe this idea more precisely, we bring quasimodels into play. By simplifying a bit, one can say that a quasimodel is a sequence $(S_n \mid n \in \mathbb{N})$ of saturated constraint systems which satisfies various conditions regulating the interaction between the S_n's (for a formal definition see below). These conditions ensure that quasimodels show well-behaviour with respect to satisfiability, and, hence, can serve as representations of models. As for individuals, in contrast to common practice, a variable occuring in a constraint system does

Modal rules on formulas

$S \longrightarrow_{\mathcal{U}} X \cup S$, if

- $(\varphi \mathcal{U} \psi) \in S$
- $\psi \notin S$ and $\{\varphi, \bigcirc(\varphi \mathcal{U} \psi)\} \not\subseteq S$
- $X = \{\psi\}$ or $X = \{\varphi, \bigcirc(\varphi \mathcal{U} \psi)\}$

$S \longrightarrow_{\neg\mathcal{U}} X \cup S$, if

- $\neg(\varphi \mathcal{U} \psi) \in S$
- $\{\neg\psi, \neg\varphi\} \not\subseteq S$ and $\{\neg\psi, \bigcirc\neg(\varphi \mathcal{U} \psi)\} \not\subseteq S$
- $X = \{\neg\psi, \neg\varphi\}$ or $X = \{\neg\psi, \bigcirc\neg(\varphi \mathcal{U} \psi)\}$

$S \longrightarrow_{\neg\bigcirc} \{\bigcirc\neg\varphi\} \cup S$, if

- $\neg \bigcirc \varphi \in S$ and $\bigcirc\neg\varphi \notin S$

$\mathcal{ALC}$-rules on formulas

$S \longrightarrow_{\neg\neg} \{\varphi\} \cup S$, if

- $\neg\neg\varphi \in S$ and $\varphi \notin S$

$S \longrightarrow_{\wedge} \{\varphi, \psi\} \cup S$, if

- $\varphi \wedge \psi \in S$
- $\varphi \notin S$ or $\psi \notin S$

$S \longrightarrow_{\neg\wedge} \{\neg\theta\} \cup S$, if

- $\neg(\varphi \wedge \psi) \in S$
- $\neg\varphi \notin S$ and $\neg\psi \notin S$
- $\theta \doteq \varphi$ or $\theta \doteq \psi$

Figure 1: Saturation rules: Rules that operate on formulas.

Modal rules on concepts

$S \longrightarrow_{\mathcal{U}c} X \cup S$, if

- $x : (C \mathcal{U} D) \in S$
- $x : D \notin S$ and $\{x : C, x : \bigcirc(C\mathcal{U}D)\} \not\subseteq S$
- $X = \{x : D\}$ or $X = \{x : C, x : \bigcirc(C\mathcal{U}D)\}$

$S \longrightarrow_{\neg\mathcal{U}c} X \cup S$, if

- $x : \neg(C\mathcal{U}D) \in S$
- $\{x : \neg D, x : \neg C\} \not\subseteq S$ and $\{x : \neg D, x : \bigcirc\neg(C\mathcal{U}D)\} \not\subseteq S$
- $X = \{x : \neg D, x : \neg C\}$ or $X = \{x : \neg D, x : \bigcirc\neg(C\mathcal{U}D)\}$

$S \longrightarrow_{\neg\bigcirc c} \{x : \bigcirc\neg C\} \cup S$, if

- $x : \neg \bigcirc C \in S$ and $x : \bigcirc\neg C \notin S$

$\mathcal{ALC}$-rules on concepts

$S \longrightarrow_{\neg\neg c} \{x : C\} \cup S$, if

- $x : \neg\neg C \in S$ and $x : C \notin S$

$S \longrightarrow_{\sqcap} \{x : C, x : D\} \cup S$, if

- $x : C \sqcap D \in S$
- $x : C \notin S$ or $x : D \notin S$

$S \longrightarrow_{\neg\sqcap} \{x : \neg E\} \cup S$, if

- $x : \neg(C \sqcap D) \in S$
- $x : \neg C \notin S$ and $x : \neg D \notin S$
- $E \doteq C$ or $E \doteq D$

$S \longrightarrow_{=} \{x : C\} \cup S$, if

- $C = \top \in S$, x occurs in S, and $x : C \notin S$

Figure 2: Saturation rules: Rules that operate on concepts.

$S \longrightarrow_{\neq} \{v : \neg C\} \cup S$, if

- $C \neq \top \in S$
- there exists no y with $y : \neg C \in S$
- v is the first new variable from V

$S \longrightarrow_{\exists} X \cup S$, if

- $x : \exists R.C \in S$
- $X = \{v : C\} \cup \{v : \neg D \,|\, x : \neg\exists R.D \in S\} \cup \{v : E \,|\, E = \top \in S\}$
- v is the first new variable from V
- v is not blocked in S, that is to say that there is no z in S such that $\{F \,|\, v : F \in X\} \subseteq \{F \,|\, z : F \in S\}$

Figure 3: Saturation rules: Locally generating rules.

not stand for an individual, but encodes a concept type, i.e. a set of concepts which coincides with all concepts (from the fixed fragment) realized by some individual at a certain moment of time. An individual d is then represented as a function r the domain of which is a set of the form $\{m \in \mathbb{N} \,|\, n \leq m\}$, for some natural number n — intuitively, the domain of r fixes the set of moments at which d is supposed to exist — and, for each $m \in dom(r)$, it selects a variable v_m from S_m, namely that variable which encodes the concept type that is realized, so to speak, by the individual d at m. Functions of this sort will be called runs. Of course, that r represents an individual d, requires a bit more (see Definition 4.2). For instance, if S_m contains the formula $r(m) : \bigcirc C$, that is, the individual d represented by r satisfies $\bigcirc C$ at moment m, then S_{m+1} must contain $r(m+1) : C$, which means that d satisfies C at the next moment.

With respect to the tableau calculus under development, all this has the following 'simple' consequence. Constructing tableaux should be regarded as an efficient procedure to search for quasimodels. This will be reflected by the fact that from a complete tableau for a formula φ, assumed it is satisfiable, we do not directly obtain a satisfying model, but only a quasimodel. However, by the main result of this section this is sufficient. For Theorem 4.5 says that a formula is satisfiable if and only if it has a quasimodel.

We start with the formal presentation. Throughout this section we keep some $\mathcal{TL}_{\mathcal{ALC}}$-formula ϑ fixed.

Definition 4.1 *A quasiworld for ϑ is a saturated clash-free constraint system S that satisfies the following conditions:*

- $\{a \,|\, \exists C((a : C) \in S)\} = ob(\vartheta)$.
- $con(S) \subseteq Fg(\vartheta)$.
- *For every formula $\varphi \in S$, if φ is a $\mathcal{TL}_{\mathcal{ALC}}$-formula then $\varphi \in Fg(\vartheta)$.*

Definition 4.2 *A sequence of quasiworlds $Q = (S_n \mid n \in \mathbb{N})$ for ϑ is called a ϑ-sequence. A run in Q is a function r with domain $\{m \in \mathbb{N} \mid m \geq n\}$, for some $n \in \mathbb{N}$, which associates a term $r(m)$ from S_m with every $m \geq n$, and satisfies the following two conditions:*

1. *For all $m \in dom(r)$ and for all concepts C, if $(r(m) : \bigcirc C) \in S_m$ then $(r(m+1) : C) \in S_{m+1}$.*
2. *For all $m \in dom(r)$, if $(r(m) : C\mathcal{U}D) \in S_m$, then there is some $k \geq m$ such that: $(r(k) : D) \in S_k$ and for every $m \leq i < k$ it holds that $(r(i) : C) \in S_i$.*

Definition 4.3 *A ϑ-sequence Q is called a quasimodel for ϑ if the following hold:*

(i) For every object name a, $r_a(n) := a$, for $n \in \mathbb{N}$, defines a run in Q.

(ii) For every $n \in \mathbb{N}$ and every variable v in S_n there is a run r in Q such that $r(n) = v$.

(iii) For every $n \in \mathbb{N}$ and every $\bigcirc\varphi \in S_n$, φ is contained in S_{n+1}.

(iv) For every $n \in \mathbb{N}$ and every $(\varphi\mathcal{U}\psi) \in S_n$ there is some $m \geq n$ such that $\psi \in S_m$ and for every $n \leq k < m$ it holds that $\varphi \in S_k$.

Definition 4.4 *We say that ϑ is quasi-satisfiable if there is some quasimodel $Q = (S_n \mid n \in \mathbb{N})$ for ϑ and some $n \in \mathbb{N}$ such that $\vartheta \in S_n$.*

Theorem 4.5 *A $\mathcal{TL}_{\mathcal{ALC}}$-formula ϑ is satisfiable if and only if it is quasi-satisfiable.*

Proof. For the direction from left to right suppose that ϑ is satisfiable. Then there is some model $\mathfrak{M} = (\mathbb{N}, <, I)$ with $\mathfrak{M}, 0 \models \vartheta$.

Fix $n \in \mathbb{N}$. For each $d \in \Delta^n$ put $t^{I,n}(d) = \{C \in Fg(\vartheta) \mid d \in C^{I,n}\}$. Next define an equivalence relation $\sim_n$ on Δ^n as follows: put $d \sim_n d'$ iff $t^{I,n}(d) = t^{I,n}(d')$. Consider the equivalence classes modulo $\sim_n$, abbreviated by $[d]_n$. Obviously, $\{[d]_n \mid d \in \Delta^n\}$ is finite. Choose for each equivalence class $[d]_n$ an individual variable $v_{[d]_n}$. The constraint system S_n corresponding to n is defined as the union of the following three sets:

- $\{\varphi \in Fg(\vartheta) \mid \mathfrak{M}, n \models \varphi\}$,
- $\{a : C \mid a \in ob(\vartheta), C \in Fg(\vartheta), a^{I,n} \in C^{I,n}\}$,
- $\{v_{[d]_n} : C \mid d \in \Delta^n, C \in Fg(\vartheta), d \in C^{I,n}\}$.

It is not difficult to prove that S_n is a quasiworld for ϑ. We first show that S_n is saturated. For this purpose we have to show that S_n is f-saturated and that each term x that occurs in S_n is saturated in S_n. Let us start with f-saturation. We have to distinguish a number of cases. Because all cases can be proved by a rather similar procedure, we will content with one example: Suppose $\neg(\varphi\mathcal{U}\psi) \in S_n$. This means, by definition, that $\mathfrak{M}, n \models \neg(\varphi\mathcal{U}\psi)$. By an easy semantic calculation we infer that either $\mathfrak{M}, n \models \neg\varphi$ and $\mathfrak{M}, n \models \neg\psi$ or $\mathfrak{M}, n \models \neg\psi$ and $\mathfrak{M}, n \models \bigcirc\neg(\varphi\mathcal{U}\psi)$. Moreover, all formulas under consideration are members of $Fg(\vartheta)$. Hence, in both cases we get the right result: in the first case $\{\neg\varphi, \neg\psi\} \subseteq S_n$ and in the second $\{\neg\psi, \bigcirc\neg(\varphi\mathcal{U}\psi)\} \subseteq S_n$. By similar arguments one can show that S_n is closed under the remaining modal rules as well as under the $\mathcal{ALC}$-rules operating on formulas. So it has been shown that S_n is f-saturated.

Next, fix some term v from S_n. To show that v is locally saturated in S_n is left to the reader. We only consider the rule $\longrightarrow_=$: Suppose $C = \top \in S_n$ and let x be some term that occurs in S_n. Assume $x = v_{[d]_n}$, for some $d \in \Delta^n$. From $C = \top \in S_n$ we conclude $\mathfrak{M}, n \models C = \top$, by definition of S_n. Hence $d \in C^{I,n}$. Moreover, $C \in Fg(\vartheta)$. Taken together this yields $(v_{[d]_n} : C) \in S_n$. If x is some object name the argument is the same.

It remains to prove that S_n is closed under the two locally generating rules. The case $\longrightarrow_{\neq}$ is obvious. For the other case we must show that there is no formula in S_n to which $\longrightarrow_{\exists}$ can be applied. Suppose $(v : \exists R.C) \in S_n$. By definition there is some $d \in \Delta^n$ such that $v = v_{[d]_n}$. Hence, $d \in (\exists R.C)^{I,n}$. So there is some $d' \in \Delta^n$ with (i) $dR^{I,n}d'$ and (ii) $d' \in C^{I,n}$. Since $C \in Fg(\vartheta)$, we get $(v_{[d']_n} : C) \in S_n$ from (ii). Further, as an easy consequence of (i), we obtain $(v_{[d']_n} : \neg D) \in S_n$ for every D with $(v : \neg\exists R.D) \in S_n$. Finally, it is easy to see that if $(E = \top) \in S_n$, then $(v_{[d']_n} : E) \in S_n$. So $v_{[d']_n}$ satisfies all the required conditions. This completes the proof that v is saturated in S_n. That every object name a from S_n is saturated in S_n can be proved in a similar way.

From the above we infer that S_n is saturated. Clearly, S_n is clash-free as well. To conclude this part of the proof, we need to show that S_n satisfies the three conditions from Definition 4.1. But this is an immediate consequence of how S_n has been defined. Therefore, it is shown that S_n is a quasiworld for ϑ.

Now, define Q as the sequence $(S_n \mid n \in \mathbb{N})$. We need to show that Q is a quasimodel for ϑ. That ϑ is quasi-satisfiable is then an immediate consequence of the fact that $\vartheta \in S_0$. So it only remains to check that Q satisfies the conditions from Definition 4.3. The first condition is obvious. For the second, let v be some variable that occurs in S_n. By definition, there is some $d \in \Delta^n$ with $v = v_{[d]_n}$. For each $m \geq n$ we define $r_v(m) = v_{[d]_m}$. Since $\mathfrak{M}$ has increasing domains, r_v is well-defined. It is easy to see that r_v is a run in Q (in the sense of Definition 4.2) and $r_v(n) = v$. This proves the second condition. The two remaining ones are proved from the definition of the quasiworlds S_n by using the semantics of $\mathcal{TL}_{\mathcal{ALC}}$ together with the closure conditions on $Fg(\vartheta)$.

Now we turn to the direction from right to left, which forms the interesting part of the theorem. Suppose ϑ is quasi-satisfied in a quasimodel $Q = (S_n \mid n \in \mathbb{N})$. Denote by Ω the set of all runs in Q and put, for each $n \in \mathbb{N}$:

$$\Delta^n = \{r \mid r \in \Omega, n \in \mathrm{dom}(r)\}$$

Let, for $n \in \mathbb{N}$,

$$I(n) = (\Delta^n, R_0^{I,n}, \ldots, C_0^{I,n}, \ldots, a_0^{I,n}, \ldots)$$

where

- $a^{I,n} = r_a$, for every $a \in ob(\theta)$,
- $C_i^{I,n} = \{r \in \Delta^n \mid r(n) : C_i \in S_n\}$,
- for $r_1, r_2 \in \Delta^n$ we put $r_1 R_i^{I,n} r_2$ iff there exists a concept C such that

$$\{r_1(n) : \exists R_i.C, r_2(n) : C\} \cup \{r_2(n) : \neg D \mid r_1(n) : \neg\exists R_i.D \in S_n\} \subseteq S_n.$$

We show that $\mathfrak{M} = (\mathbb{N}, <, I)$ is as required. To this end we first prove:

(I) For every $n \in \mathbb{N}$ and every $r \in \Delta^n$, if $(r(n) : C) \in S_n$ then $r \in C^{I,n}$.

This is proved by induction on the construction of C. Throughout the proof we heavily exploit the fact that the S_n are saturated, hence closed under all saturation rules. Note that in the induction one has to make use of a different measure of the complexity of concepts. According to this measure a concept C that has not the form of a negation and its negation $\neg C$ are of the same complexity.

Fix $n \in \mathbb{N}$ and suppose $r \in \Delta^n$. For atomic C the claim follows by the choice of the model. Let $C \doteq \neg D$, with D atomic. Since S_n is clash-free, $(r(n) : D) \notin S_n$. Hence $r \notin D^{I,n}$, which implies $r \in C^{I,n}$. Next, consider $C \doteq (D \sqcap E)$. By the closure of S_n under the rule $\longrightarrow_{\sqcap}$ we obtain $(r(n) : D) \in S_n$ and $(r(n) : E) \in S_n$. From the induction hypothesis we get $r \in D^{I,n}$ and $r \in E^{I,n}$, hence, by semantics, $r \in (D \sqcap E)^{I,n}$. The cases $C \doteq \neg(D \sqcap E)$ and $C \doteq \neg\neg D$ are proved in a similar way. One just has to use the fact that S_n is closed under the rules $\longrightarrow_{\neg\sqcap}$ and $\longrightarrow_{\neg\neg c}$, respectively.

Suppose $C \doteq \exists R.D$. The closure under the rule $\longrightarrow_{\exists}$ provides some variable v with $\{v : D\} \cup \{v : \neg E \mid r(n) : \neg\exists R.E \in S_n\} \subseteq S_n$. We find a run $r' \in \Delta^n$ with $r'(n) = v$. From this we obtain $rR^{I,n}r'$, by definition, as well as $r' \in D^{I,n}$, by induction hypothesis. Hence $r \in (\exists R.D)^{I,n}$. Now the case $C \doteq \neg\exists R.D$ has to be checked. Suppose $rR^{I,n}r'$. By definition there is some concept E such that $(r(n) : \exists R.E) \in S_n$, $(r'(n) : E) \in S_n$, and $\{r'(n) : \neg E' \mid (r(n) : \neg\exists R.E') \in S_n\} \subseteq S_n$. From the latter we obtain $(r'(n) : \neg D) \in S_n$, and, hence, by induction hypothesis, $r' \in (\neg D)^{I,n}$. Since r' was chosen arbitrarily, it is shown that $r \in (\neg\exists R.D)^{I,n}$.

Next, let $C \doteq \bigcirc D$ and suppose $(r(n) : \bigcirc D) \in S_n$. By the first clause in Definition 4.2, $(r(n+1) : D) \in S_{n+1}$. Hence, by induction hypothesis, $r \in D^{I,n+1}$. From the latter we obtain $r \in (\bigcirc D)^{I,n}$ by semantics. Now consider $C \doteq \neg \bigcirc D$. Since S_n is saturated, we get $(r(n) : \bigcirc\neg D) \in S_n$. The rest follows from the preceding case.

Let $C \doteq (D\mathcal{U}E)$. Then, by the second clause of Definition 4.2, there is some $m \geq n$ such that: $(r(m) : E) \in S_m$ and for every $n \leq i < m$ it holds that $(r(i) : D) \in S_i$. From this the result can be easily obtained by making use of the induction hypothesis. To complete the proof, the case $C \doteq \neg(D\mathcal{U}E)$ need to be considered. So suppose $(r(n) : \neg(D\mathcal{U}E)) \in S_n$. By induction on m we show the following claim:

(A) For all $m \geq n$, $\{(r(m) : \neg E), (r(m) : \neg(D\mathcal{U}E))\} \subseteq S_m$ or there is some i such that $n \leq i < m$ and $(r(i) : \neg D) \in S_i$.

For the start, let $m = n$. The claim is an immediate consequence of the assumption together with the fact that S_n is closed under the rule $\longrightarrow_{\neg\mathcal{U}c}$. For the induction step assume that the claim has already been proved for $m = k$. We distinguish two cases: First, suppose there is some i such that $n \leq i < k$ and $(r(i) : \neg D) \in S_i$. In this case the induction step follows immediately. Suppose now there is no i with this property. Then $\{(r(k) : \neg E), (r(k) : \neg(D\mathcal{U}E))\} \subseteq S_k$. By the closure under the rule $\longrightarrow_{\neg\mathcal{U}c}$, one of the following holds: (i) $(r(k) : \neg D) \in S_k$ or (ii) $(r(k) : \neg\bigcirc(D\mathcal{U}E)) \in S_k$. In the first case the induction step is trivial. In the second case we get $(r(k) : \bigcirc\neg(D\mathcal{U}E)) \in S_k$, by the closure under the rule $\longrightarrow_{\neg\bigcirc c}$. By making use of the first clause of Definition 4.2, we infer $(r(k+1) : \neg(D\mathcal{U}E)) \in S_{k+1}$. The result is obtained by the closure of S_{k+1} under $\longrightarrow_{\neg\mathcal{U}c}$. This completes the induction step, and hence the proof of (A).

Now we come back to the proof of (I). It is easy to see that (A) yields:

(B) For all $m \geq n$, $r \in (\neg E)^{I,m}$ or there is some i such that $n \leq i < m$ and $r \in (\neg D)^{I,i}$,

by induction hypothesis. From the latter we infer the desired result $r \in (\neg(D\mathcal{U}E))^{I,n}$ by a simple semantical argument. Hence (I) has been shown.

In the next step we show the following claim:

(II) For every $n \in \mathbb{N}$ and every $\varphi \in Fg(\vartheta)$, if $\varphi \in S_n$ then $\mathfrak{M}, n \models \varphi$.

Again, the claim is shown by induction. Let φ be atomic and suppose $\varphi \in S_n$. We must distinguish two cases: First, there is some object name a and some concept C such that $\varphi \doteq (a : C)$. By the first clause of Definition 4.3, we obtain $(r_a(n) : C) \in S_n$. Hence, by (I), $r_a \in C^{I,n}$. Note that $a^{I,n}$ was defined as r_a. So, by semantics, $\mathfrak{M}, n \models a : C$. Second, there is some concept C with $\varphi \doteq (C = \top)$. Fix $r \in \Delta^n$. Now, consider the term $r(n)$. Since S_n is closed under the rule $\longrightarrow_=$, we get $(r(n) : C) \in S_n$. An application of (I) yields $r \in C^{I,n}$.

Next, consider $\varphi \doteq \neg\psi$, with ψ atomic. Again, two cases need to be distinguished. The first case, where ψ has the form $a : C$, is clear. For the second case, suppose $\psi \doteq (C = \top)$. Assume $(C \neq \top) \in S_n$. That S_n is closed under the rule $\longrightarrow_{\neq}$ supplies a variable v with $(v : \neg C) \in S_n$. By the second clause in Definition 4.3 we have a run r such that $r(n) = v$. Moreover, by an application of (I), it follows that $r \in (\neg C)^{I,n}$. So there is some $d \in \Delta^n$ with $d \in (\neg C)^{I,n}$, that is $d \notin C^{I,n}$. But from the latter we infer $\mathfrak{M}, n \models C \neq \top$, what had to be shown.

The proof of the induction step can be left to the reader. The different clauses are all rather similar to the corresponding ones in the proof of (I). The only interesting cases are $\varphi \doteq (\psi\mathcal{U}\chi)$ and $\varphi \doteq \bigcirc\psi$ together with their negations. Here one uses clause (iii) and (iv) from Definition 4.3, where clause 1 and 2 from Definition 4.2 have been used in the proof of (I).

In order to complete the proof of the theorem we reason as follows: By assumption, there is some $n \in \mathbb{N}$ such that $\vartheta \in S_n$. So an application of (II) yields $\mathfrak{M}, n \models \vartheta$, which means that ϑ is satisfiable in $\mathfrak{M}$. $\dashv$

5 Tableau rules

In this section we introduce the set of tableau rules on which our tableau calculus for $\mathcal{TL}_{\mathcal{ALC}}$ is based. The tableau calculus uses two different kinds of rules: rules that expand a constraint system and rules that add a new constraint system to the tableau. In our case the latter class consists of only one rule, the rule $\longrightarrow_{\bigcirc}$ (see below), whereas the second class divides into a number of subclasses.

This distinction among the rules reflects the following ontological distinction: A constraint system can be regarded as a partial description of the facts that hold at a moment. All the rules except $\longrightarrow_{\bigcirc}$ add new information with respect to the moment under consideration. In this sense they can be regarded as local rules. The rule $\longrightarrow_{\bigcirc}$ does something different; it adds a new moment to the structure under construction and, therefore, starts to build up some new saturated constraint system. In our tableau calculus it can only be applied to leafs which are saturated.

As for the local rules, we are in a rather comfortable situation. Except for two cases, we can use the saturation rules from Section 3. These exceptions concern the rules involving the until operator, that is $\longrightarrow_{\mathcal{U}}$ and $\longrightarrow_{\mathcal{U}c}$. We have to change these two rules slightly when we want to use them as part of the tableau algorithm. Though the reason will become obvious in Section 7, it might be useful to give some hint here. Consider the three conditions under which the rule $\longrightarrow_{\mathcal{U}}$ is applied to a formula $(\varphi\mathcal{U}\psi) \in S$. Note that, as a consequence of the second condition, we have that in a situation in which S already contains ψ or both φ and $\bigcirc(\varphi\mathcal{U}\psi)$ we are not allowed to apply the rule.

We illustrate this observation by means of a simple example. Here we use the temporal operators $\Diamond$, $\Box$, and $\bigcirc$ only. For those simple operators the rules $\longrightarrow_{\mathcal{U}}$ and $\longrightarrow_{\neg\mathcal{U}}$ translate to:

$S \longrightarrow_{\Diamond} S \cup X$ iff $\Diamond\varphi \in S$, $\varphi \notin S$ and $\bigcirc\Diamond\varphi \notin S$, and $X = \{\varphi\}$ or $X = \{\bigcirc\Diamond\varphi\}$.

$S \longrightarrow_{\Box} S \cup X$ iff $\Box\varphi \in S$, $\{\bigcirc\Box\varphi, \varphi\} \not\subseteq S$, and $X = \{\bigcirc\Box\varphi, \varphi\}$.

Consider now the satisfiable constraint system

$$\{\Box \bigcirc \bigcirc\Diamond(C = \top), \Diamond(C = \top), \bigcirc\Diamond(C = \top)\} \tag{1}$$

The rule $\longrightarrow_{\Diamond}$ cannot be applied to (1) since $\bigcirc\Diamond(C = \top)$ belongs to (1). Hence, the only rule applicable to (1) is $\longrightarrow_{\Box}$. The application of this rule to (1) returns

$$\{\bigcirc\Box \bigcirc \bigcirc\Diamond(C = \top), \bigcirc \bigcirc \Diamond(C = \top), \Diamond(C = \top), \bigcirc\Diamond(C = \top)\} \tag{2}$$

No local rule is applicable to (2), so the only applicable rule is the globally generating rule $\longrightarrow_{\bigcirc}$ (see below), which creates a new world by removing the operator $\bigcirc$ from the front of every formula. Applying it to (2) returns (1) again. Thus, by leaving the rules as they are now we would never introduce $(C = \top)$ as a witness for $\Diamond(C = \top)$ and the tableau would not be correct. The reason is that $\longrightarrow_{\Diamond}$ will never be applied because $\bigcirc\Diamond(C = \top)$ belongs to both (1) and (2).

Clearly, to solve this problem, it is not enough just to drop the second condition, because then the rule could be applied forever, preventing the tableau algorithm from terminating. What we are looking for is a kind of mechanism that ensures that in saturating a given constraint system, that occurs in a tableau, the until rule will be applied once, and only once. Following Wolper in [29] we use the method of marking formulas for this purpose:

Each formula of the form $\varphi\mathcal{U}\psi$ or $x : C\mathcal{U}D$ which occurs in a constraint system S within a tableau will be classified with respect to the following feature: either it occurs *marked* in S or it occurs *unmarked* in S, but not both. That a formula φ is marked in S will be communicated by $\varphi^* \in S$. By utilizing this device, we replace $\longrightarrow_{\mathcal{U}}$ by

$S \longrightarrow^t_{\mathcal{U}} X \cup \{(\varphi\mathcal{U}\psi)^*\} \cup (S \setminus \{\varphi\mathcal{U}\psi\})$, if

- $(\varphi\mathcal{U}\psi)$ occurs unmarked in S
- $X = \{\psi\}$ or $X = \{\varphi, \bigcirc(\varphi\mathcal{U}\psi)\}$

It should be clear that a similar problem arises in the case of $\longrightarrow_{\mathcal{U}c}$. Therefore, we have to change this rule as well:

$S \longrightarrow^t_{\mathcal{U}c} X \cup \{(x : (C\mathcal{U}D))^*\} \cup (S \setminus \{x : (C\mathcal{U}D)\})$, if

- $x : (C\mathcal{U}D)$ occurs unmarked in S
- $X = \{x : D\}$ or $X = \{x : C, x : \bigcirc(C\mathcal{U}D)\}$

Here a short remark is in order. By changing the rules one has also changed the notion of a saturated constraint system. So, strictly speaking, we have to distinguish between saturated systems in the old sense and saturated systems in the sense of the new tableau rules. We will not do so explicitly, however, since it will always be clear from the context which type of saturation is considered.

We now come to the only non-local rule. As has been emphasized before, this rule has a completely different character. Before we can formulate it in a precise way, we have to introduce some new notions. Let S be a constraint system and x some term that occurs in S. We use $A_x(S)$ to designate the set $\{C \mid (x : \bigcirc C) \in S\}$. Further, we define an equivalence relation $\sim_S$ on the set of *variables* from S by setting $v \sim_S u$ iff $A_v(S) = A_u(S)$. We denote the equivalence class of a variable v with regard to this relation by $[v]_S$. Finally, $[S]_\sim$ is used to denote the set of all equivalence classes $[v]_S$.

The globally generating rule or the rule for adding a new world

$S \longrightarrow_{\bigcirc} S'$, if S' is the union of the following sets:

- $\{a : \top\} \cup \{a : C \mid (a : \bigcirc C) \in S\}$, for $a \in ob(S)$,
- $\{\psi \mid \bigcirc \psi \in S\}$,
- $\{min([v]_S) : \top\} \cup \{min([v]_S) : C \mid (min([v]_S) : \bigcirc C) \in S\}$, for $[v]_S \in [S]_\sim$.

Note that all formulas in S' are unmarked. We define the set of tableau rules, $\mathcal{R}^t$ for short, as that set of rules which we obtain from the set of saturation rules by replacing $\longrightarrow_{\mathcal{U}}$ and $\longrightarrow_{\mathcal{U}c}$ by $\longrightarrow^t_{\mathcal{U}}$, $\longrightarrow^t_{\mathcal{U}c}$ respectively, and by adding the globally generating rule from above.

6 Tableaux and complete tableaux

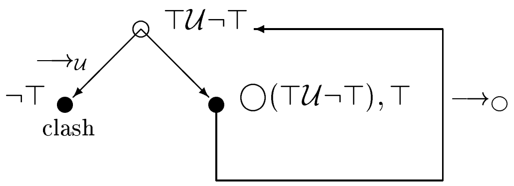
In this section we give a formal account of tableaux and related notions. For preparation, it might be useful to start with an informal description of how the tableau rules will be used to build up a (complete) tableau for a given formula ϑ. To keep our presentation lucid, we follow the process of building a (complete) tableau only along one branch.

So let ϑ be some $\mathcal{TL}_{\mathcal{ALC}}$-formula. Define $\{\vartheta\}$ as the initial node of the tableau under construction. The first main step is to apply the tableau rules so as to build up a saturated set. Throughout this process we apply the rules in the following order:[3] (1) Apply the modal rules and $\mathcal{ALC}$-rules for formulas until you reach an f-saturated set. (2) Reiterate the following three steps until you reach a saturated set S: first, use the rules operating on concepts to each term x; second, apply $\longrightarrow_{\neq}$, and, third, apply $\longrightarrow_{\exists}$. Suppose one has met a saturated set S on the branch under construction, then (3) one applies $\longrightarrow_{\bigcirc}$ and obtains a set S'. With respect to the ontological meaning of the nodes in the tableau, this step corresponds to the creation of a new moment. After this step, the saturation process will start again, this time one uses S' instead of $\{\vartheta\}$ as its base. We saturate S', apply $\longrightarrow_{\bigcirc}$, saturate again, and so on. This process will be continued until one reaches a node g which fulfills one of the following two conditions: (i) The constraint system associated with g contains a clash. (ii)

[3]Note that in our formal account (see Definition 6.2) we leave the order in which the saturation rules should be applied even less specified. We just require that the locally generating rules are applicable only if no non-generating rule is applicable.

The rule next to be applied is the globally generating rule and its application would provide (modulo renaming of the variables) a constraint system which already exists somewhere in the tableau (not necessarily on the same branch). Below it will be shown that after a finite number of steps we always meet a situation which satisfies one of the termination conditions. Since the tableau has the finite branching property, this yields termination for the whole tableau algorithm.[4]

So suppose we have built up a complete tableau $\mathcal{G}$ for ϑ, and assume that $\mathcal{G}$ has at least one clash-free leaf. Usually, this means that ϑ is satisfiable. However, for linear temporal logic, and hence for $\mathcal{TL}_{\mathcal{ALC}}$ as well, this criterion does not apply. There exist formulas the complete tableaux of which have open branches, but the formulas themselves are not satisfiable. Counterexamples of this sort can be found among formulas of the form $\varphi\mathcal{U}\psi$ or $x : C\mathcal{U}D$. The reason is the following: By applying the tableau rules for Until formulas it is possible to generate a branch b in a tableau — starting with a set S that contains the formula $\varphi\mathcal{U}\psi$, for instance — which, on the one hand side, contains no clash, but, on the other side, also contains no set S' with $\psi \in S'$. In other words, in the construction of b the attempt to realize $\varphi\mathcal{U}\psi$ by means of choosing ψ has always been postponed in favor of the second alternative, adding $\bigcirc(\varphi\mathcal{U}\psi)$, until the whole process of building up the tableau terminated. Now suppose that this holds for every branch that starts in S and contains no clash. Below we describe this situation as one in which S contains some eventuality that is not realized. In such a case S is not satisfiable in spite of the fact that it fulfills the above criterion. For an easy example, consider the complete tableau for the formula $\top\mathcal{U}\neg\top$:



In order to overcome this difficulty, Wolper ([29]) added a second component to his tableau procedure. After having built up a complete tableau for ϑ, one has to eliminate those nodes that contain a clash or some unrealized eventuality. Only nodes that survive this elimination process may be used to build up a model for satisfying ϑ. Wolper proved that ϑ is satisfiable if and only if the initial node of its complete tableau has not been eliminated. It will turn out that Wolper's elimination rules can be used for $\mathcal{TL}_{\mathcal{ALC}}$ as well (see the end of this section).

Definition 6.1 *Let S, S' and S'' be constraint systems. S' is called a variant of S, if there is some bijective function π from the variables occuring in S onto the variables occuring in S' such that S' is obtained from S by replacing each variable v from S with $\pi(v)$. We call π a renaming. We write $S'' \longrightarrow_\bigcirc^\pi S'$, if there is some variant S of S' such that $S'' \longrightarrow_\bigcirc S$, and π is the respective renaming.*

Definition 6.2 *Let ϑ be a $\mathcal{TL}_{\mathcal{ALC}}$-formula. A tableau for ϑ is a triple $\mathcal{G} = (G, \prec, l)$, where $(G, \prec)$ is a finite tree and l is a labelling function associating with each $g \in G$ some constraint system $l(g)$ for ϑ and $\{\vartheta\} \cup \{min(V) : \top\} \cup \{a : \top \mid a \in ob(\vartheta)\}$ to the root of $\mathcal{G}$.*

[4] At this point a short remark is in order. In this paper we are concerned with the conceptual presentation of the tableau-algorithm, and we are not primarily concerned with its computational efficiency or possible optimizations of the procedure. This is left for future work.

Given two tableaux $\mathcal{G} = (G, \prec, l)$ and $\mathcal{G}' = (G', \prec', l')$ for ϑ and a tableau rule $T \in \mathcal{R}^t$ we say that $\mathcal{G}'$ is obtained from $\mathcal{G}$ by an application of T iff there exists a leaf $g^* \in G$ and a non-empty set X disjoint from G with

- $G' = G \cup X$,
- $\prec' = \prec \cup \{\langle g^*, h\rangle : h \in X\}$,
- *for every $g \in G$, $l(g) = l'(g)$,*
- *for every constraint system S obtained from $l(g^*)$ by an application of T there exists precisely one $h \in X$ such that $l'(h) = S$ (hence X consists of two nodes iff T is a disjunctive rule and otherwise of one node),*

and the following holds:

- *if T is $\longrightarrow_{\bigcirc}$, then $l(g^*)$ is saturated, no other rule is applicable to a leaf of $\mathcal{G}$, and there does not exist a $g' \in G$ and a renaming π with $l(g^*) \longrightarrow^{\pi}_{\bigcirc} l(g')$ (i.e., the application of the rule $\longrightarrow_{\bigcirc}$ is not blocked),*
- *if $T \in \{\longrightarrow_{\exists}, \longrightarrow_{\neq}\}$, then no non-generating rule is applicable to any leaf in $\mathcal{G}$.*

A tableau is complete if no tableau rule is applicable to it. The tableau associated with ϑ is $(\{g^r\}, \prec^r, l)$, where $\prec^r = \emptyset$ and $l(g^r) = \{\vartheta\} \cup \{min(V) : \top\} \cup \{a : \top \mid a \in ob(\vartheta)\}$. A tableau $\mathcal{G}$ for ϑ is called a completion of ϑ iff it is complete and there exists a sequence $\mathcal{G}_i$, $0 \le i \le n$, with $\mathcal{G}_0$ the tableau associated with ϑ, $\mathcal{G}_n = \mathcal{G}$, and such that for every $i < n$ there exists a tableau rule $T \in \mathcal{R}^t$ such that $\mathcal{G}_{i+1}$ is obtained from $\mathcal{G}_i$ by an application of T.

A node $g \in G$ is called a pre-state iff it is the root of $\mathcal{G}$ or $l(g')$ is saturated for the unique g' with $g' \prec g$.

Definition 6.3 *Let $\mathcal{G}$ be a tableau for ϑ. We define a new relation $\overline{\prec}$ by putting, for $g, g' \in G$, $g \overline{\prec} g'$ if either $g \prec g'$, or g is a leaf in $\mathcal{G}$ and g' is a pre-state with $l(g) \longrightarrow^{\pi}_{\bigcirc} l(g')$ for some renaming π.*

Recall that the number of concepts and formulas in $Fg(\vartheta)$ is bounded by $\sharp_c(\vartheta)$ and $\sharp_f(\vartheta)$, respectively.

Lemma 6.4 *The process of completing a given tableau always terminates; that means, that by applying the tableau rules successively to a given tableau associated with ϑ we always reach a completion after finitely many steps.*

Proof. We are going to show the following: Given a completion $\mathcal{G}$ of ϑ, the number of pre-states in G is bounded by

$$2^{\sharp_f(\vartheta)} \cdot 2^{2^{\sharp_c(\vartheta)}} \cdot 2^{\sharp_c(\vartheta) \cdot |ob(\vartheta)|}.$$

The length of the longest $\prec$-path without $\longrightarrow_{\bigcirc}$-edges is bounded by

$$\sharp_f(\vartheta) + (2 \cdot 2^{\sharp_c(\vartheta)} + |ob(\vartheta))|) \cdot \sharp_c(\vartheta).$$

First observe that for any two pre-states g and g' we know that $l(g)$ is not a variant of $l(g')$. It follows that every pre-state g is determined (modulo variants) by the union of the sets:

- $l(g) \cap \bigcirc(\neg(Sb\{\vartheta\}))$,
- $\{\{C \mid (v : C) \in l(g)\} \mid v$ occurs in $l(g)\}$,
- $\{(a : C) \mid (a : C) \in l(g), a \in ob(\vartheta)\}$.

The number of such sets is bounded by $2^{\sharp_f(\vartheta)} \cdot 2^{2^{\sharp_c(\vartheta)}} \cdot 2^{\sharp_c(\vartheta)\cdot|ob(\vartheta)|}$. Hence the number of pre-states in G is bounded by this number as well.

The number of variables in $l(g)$, g a pre-state, is bounded by $2^{\sharp_c(\vartheta)}$. Suppose there is a $\prec$-path without $\longrightarrow_{\bigcirc}$-edges from g to g'. By the definition of the rules $\longrightarrow_{\exists}$ and $\longrightarrow_{\neq}$, (which are the only rules which introduce new variables) the number of variables in $l(g')$ which are not in $l(g)$, is bounded by $2^{\sharp_c(\vartheta)}$, see Lemma 3.5. Hence, the number of all variables that occur in $l(g')$ is bounded by $2 \cdot 2^{\sharp_c(\vartheta)}$. Now the second claim follows from the fact that $l(g_1) \subset l(g_2)$ whenever $g_1 \prec g_2$ by a rule different from $\longrightarrow_{\bigcirc}$. $\dashv$

An L_C-formula α is called an *eventuality* for a constraint system S iff $\alpha \in S$ and α is of the form $(x : C\mathcal{U}D)$ or of the form $\varphi\mathcal{U}\psi$.

Definition 6.5 *Let $\mathcal{G} = (G, \prec, l)$ be a tableau for ϑ, $g \in G$, and α be some eventuality for $l(g)$. Then α is realized for g in $\mathcal{G}$ if there exists a sequence $g_0 \overline{\prec} g_1 \cdots \overline{\prec} g_n$ in G with $g = g_0$, $n \geq 0$, such that (depending on the form of α) the following holds:*

(1) for α of the form $\varphi\mathcal{U}\psi$, $\psi \in l(g_n)$;

(2) for α of the form $(v : C\mathcal{U}D)$, for some variable v, there exist variables v_i from $l(g_i)$, $i \leq n$, with $v_0 = v$, $(v_n : D) \in l(g_n)$ and for all $i < n$:

- *if $l(g_i)$ is saturated, then $\{C \mid (v_i : \bigcirc C) \in l(g_i)\} \subseteq \{C \mid (v_{i+1} : C) \in l(g_{i+1})\}$,*
- *if $l(g_i)$ is not saturated, then $v_i = v_{i+1}$.*

(3) for α of the form $(a : C\mathcal{U}D)$, for some object name a, $(a : D) \in l(g_n)$.

The conditions (1) and (3) above are straightforward 'translations' from Wolper. Condition (2) is more complex since in the step from a saturated $l(g_i)$ to its $\overline{\prec}$-successor $l(g_{i+1})$ we often have to choose a variable v_{i+1} different from v_i. If we wouldn't allow this we would be forced to introduce infinitely many variables in our tableau.

Definition 6.6 *Let $\mathcal{G} = (G, \prec, l)$ be a tableau for ϑ. We use the following rules to eliminate points in $\mathcal{G}$ (see [29, p. 124]):*

(E1) If $l(g)$ contains a clash, eliminate g.

(E2) If all the $\overline{\prec}$-successors of a point g have been eliminated, eliminate g.

(E3) If g is a pre-state and $l(g)$ contains some eventuality that is not realized for g, eliminate g.

We describe the elimination procedure in a more precise way. To this end say that a tableau $\mathcal{G}_1 = (G_1, \prec_1, l_1)$ is a *subtableau* of $\mathcal{G}_2 = (G_2, \prec_2, l_2)$ if $G_2 \supseteq G_1$ and $\mathcal{G}_1$ is the restriction of $\mathcal{G}_2$ to G_1. Obviously, if $\mathcal{G}_2$ is a tableau for ϑ and G_1 contains the root of G_2, then $\mathcal{G}_1$ is a tableau for ϑ.

Suppose now that $\mathcal{G} = (G, \prec, l)$ is a completion of ϑ. We construct a decreasing sequence of subtableaux $\mathcal{G} = \mathcal{G}_0, \mathcal{G}_1, \ldots$ by eliminating iteratively nodes from G according to the above three rules, whereby $(E1)$ only needs to be applied in the first step. The two other rules will then be used by turns. Since, by Lemma 6.4, G can assumed to be finite, this process stops after finitely many steps, that means we reach some subset G' of G to which none of the rules can be applied. We say that the root g^r is not eliminated iff it belongs to G'.

Theorem 6.7 *The following conditions are equivalent, for every $\mathcal{TL}_{\mathcal{ALC}}$-formula ϑ:*

(1) ϑ is satisfiable,

(2) there does not exist a completion of ϑ in which the root is eliminated,

(3) there exists a completion of ϑ in which the root is not eliminated.

In the rest of the paper we will prove that (1) implies (2) (Correctness) and (3) implies (1) (Completeness). The implication (2) $\Rightarrow$ (3) is obvious.

First, however, we have the following consequence of Lemma 6.4 and Theorem 6.7:

Theorem 6.8 *There is an effective procedure which, given a formula ϑ, decides whether ϑ is satisfiable.*

7 Correctness

Theorem 7.1 *Suppose ϑ is a satisfiable $\mathcal{TL}_{\mathcal{ALC}}$-formula and $\mathcal{G}$ is a completion of ϑ. Then the root of $\mathcal{G}$ is not eliminated.*

The theorem is a consequence of the following lemma. Here and in what follows we denote by $[n, m]$ the set $\{k \in \mathbb{N} \mid n \leq k \leq m\}$.

Lemma 7.2 *Let $\mathcal{G} = (G, \prec, l)$ be a completion of ϑ and suppose that ϑ is satisfiable. Then there exists a sequence $\vec{g} = (g_i \mid i \in \mathbb{N})$ of clash-free nodes in G with $g_0 = g^r$, $g_i \overline{\prec} g_{i+1}$, for all $i \in \mathbb{N}$, such that the following holds for all $n \in \mathbb{N}$:*

- *if α is an eventuality of the form $\varphi \mathcal{U} \psi$ and $\alpha \in l(g_n)$, then there exists $m \geq n$ with $\psi \in l(g_m)$ (in this case we say that α is realized for g_n in $\vec{g}$ until m);*
- *if α is an eventuality of the form $(a : C \mathcal{U} D)$ and $\alpha \in l(g_n)$, then there exists $m \geq n$ with $(a : D) \in l(g_m)$ (in this case we say that α is realized for g_n in $\vec{g}$ until m);*
- *if α is an eventuality of the form $(v : C \mathcal{U} D)$, for some variable v, then there exist $m \geq n$ and variables v_i from $l(g_i)$, for all $i \in [n, m]$, with $v_0 = v$, $(v_m : D) \in l(g_m)$, and for all $i \in [n, m-1]$:*
  - *if $l(g_i)$ is saturated, then $\{C \mid (v_i : \bigcirc C) \in l(g_i)\} \subseteq \{C \mid (v_{i+1} : C) \in l(g_{i+1})\}$,*
  - *if $l(g_i)$ is not saturated, then $v_i = v_{i+1}$,*

  (in this case we say that α is realized for g_n by $\vec{g}$ and the sequence $(v_i \mid n \leq i \leq m)$).

For suppose the lemma has been proved. We then argue as follows:

Proof. Assume ϑ is satisfiable and let $\mathcal{G} = (G, \prec, l)$ be a completion of ϑ.

An application of Lemma 7.2 supplies a sequence $\vec{g} = (g_n \mid n \in \mathbb{N})$ satisfying the conditions listed in the lemma. In particular, $g^r = g_0$. So it suffices to show that no g_n from $\vec{g}$ is eliminated. Let

$$G = G_0 \supseteq G_1 \supseteq \cdots$$

be the sequence produced by the elimination procedure. We show by induction that

- $\{g_n \mid n \in \mathbb{N}\} \subseteq G_i$, for all $i \in \mathbb{N}$.

The induction base $i = 0$ is clear. Suppose the claim has been proved for $j = i$. So $\{g_n \mid n \in \mathbb{N}\} \subseteq G_j$. We have to show that none of the rules E_1 to E_3 can be applied to any g_n. (E1) is clear and (E2), (E3) follow immediately from the induction hypothesis. $\dashv$

The next lemma will play an important role in the proof of Lemma 7.2. Consider a constraint systems S for ϑ and a model $\mathfrak{M} = (\mathbb{N}, <, I)$ with

$$I(n) = (\Delta^n, R_0^{I,n}, \ldots, C_0^{I,n}, \ldots, a_0^{I,n}, \ldots),$$

for $n \in \mathbb{N}$. A mapping ρ from the set of terms in S into Δ^n is called *n-satisfying*, in symbols $S \subseteq_n^\rho \mathfrak{M}$, if

- $\rho(a) = a^{I,n}$, for all object names a in S,
- $\{C | (x : C) \in S\} \subseteq \{C \mid \rho(x) \in C^{I,n}\}$, for all terms x in S,
- if $\psi \in S$, then $\mathfrak{M}, n \models \psi$, for all $\mathcal{TL}_{\mathcal{ALC}}$-formulas ψ.

Lemma 7.3 *Suppose $\mathfrak{M} = (\mathbb{N}, <, I)$ with $I(n) = (\Delta^n, R_0^{I,n}, \ldots, C_0^{I,n}, \ldots, a_0^{I,n}, \ldots)$, for all $n \in \mathbb{N}$, is a model for ϑ. Let $n \in \mathbb{N}$ and suppose $\mathcal{G} = (G, \prec, l)$ is a completion of ϑ and $g \in G$.*

(1) Suppose $l(g)$ is saturated, $l(g) \subseteq_n^\rho \mathfrak{M}$, and v is a variable in $l(g)$. Then there exist sequences

- *$g_0 \overline{\prec} g_1 \prec \cdots \prec g_m$ of nodes in $\mathcal{G}$, $g = g_0$, such that $l(g_i)$, $i \in [1, m-1]$, are non-saturated and $l(g_m)$ is saturated,*
- *$\rho_0, \ldots, \rho_m$ of mappings with $\rho = \rho_0$, and $\rho_1 \subseteq \cdots \subseteq \rho_m$ such that $l(g_i) \subseteq_{n+1}^{\rho_i} \mathfrak{M}$, for all $i \in [1, m]$*

and a variable v' in $l(g_1)$ such that

(a) for all $\varphi \mathcal{U} \psi \in l(g_m)$, if $\mathfrak{M}, n+1 \models \psi$ then $\psi \in l(g_m)$,

(b) for all terms x with $(x : C\mathcal{U}D) \in l(g_m)$, if $\rho_m(x) \in D^{I,n+1}$ then $(x : D) \in l(g_m)$,

(c) $\rho_1(v') = \rho(v)$ and $\{C \mid (v : \bigcirc C) \in l(g)\} \subseteq \{C \mid (v' : C) \in l(g_1)\}$.

(2) Suppose $l(g)$ is not saturated, $l(g) \subseteq_n^\rho \mathfrak{M}$, and v is a variable in $l(g)$. Then there exist sequences

- *$g_0 \prec g_1 \prec \cdots \prec g_m$ of nodes in $\mathcal{G}$, $g = g_0$, such that $l(g_i)$, $i \in [1, m-1]$, are non-saturated, and $l(g_m)$ is saturated,*

- $\rho_0, \ldots, \rho_m$ *of mappings with* $\rho = \rho_0$ *and* $\rho_0 \subseteq \rho_1 \subseteq \cdots \subseteq \rho_m$ *such that* $l(g_i) \subseteq_n^{\rho_i} \mathfrak{M}$*, for all* $i \in [1, m]$

such that

(a) for all $\varphi \mathcal{U} \psi \in l(g_m)$*, if* $\mathfrak{M}, n \models \psi$ *then* $\psi \in l(g_m)$*,*

(b) for all terms x *with* $(x : C\mathcal{U}D) \in l(g_m)$*, if* $\rho(x) \in D^{I,n}$ *then* $(x : D) \in l(g_m)$*.*

Proof. (1) Let $l(g) \subseteq_n^{\rho} \mathfrak{M}$ and $\rho(v) = d \in \Delta^n$. Take the node $g_1 \in G$ with $g \,\overline{\prec}\, g_1$. We have $l(g_0) \longrightarrow_{\bigcirc}^{\pi} l(g_1)$ for some renaming π. So $l(g_1)$ is a variant of the union of the following sets:

- $\{a : \top\} \cup \{a : C \mid (a : \bigcirc C) \in l(g)\}$, for $a \in ob(\vartheta)$,
- $\{\psi \mid \bigcirc \psi \in l(g)\}$,
- $\{(v_i : \top)\} \cup \{v_i : C \mid (v_i : \bigcirc C) \in l(g)\}$, for $0 < i \leq n$,

where $\{v_1, \ldots, v_n\} = \{min([w]_{l(g)}) \mid [w]_{l(g)} \in [l(g)]_{\sim}\}$. Note that $\{v_1, \ldots, v_n\}$ is a set of variables from $l(g)$ which contains exactly one representative for each equivalence class $[w]_{l(g)}$. Take the variable $v'' \in \{v_1, \ldots, v_n\}$ with $v'' \in [v]_{l(g)}$.

Put, for every variable w in $l(g_1)$, $\rho_1(w) = \rho(\pi^{-1}(w))$, and $\rho_1(a) = \rho(a)$, for every $a \in ob(\vartheta)$. Notice that the terms in $l(g_1)$ are precisely the members of $\{\pi(v_1), \ldots, \pi(v_n)\} \cup ob(\vartheta)$. Obviously, ρ_1 is well-defined. Observe that $l(g_1) \subseteq_{n+1}^{\rho_1} \mathfrak{M}$. This follows from $l(g) \subseteq_n^{\rho} \mathfrak{M}$ and $a^{I,n} = a^{I,n+1}$, for all $a \in ob(\vartheta)$. Note also that $\rho(v) = \rho_1(v')$, where $v' = \pi(v'')$.

We now construct the sequences $g_2, \ldots, g_m$ and $\rho_2, \ldots, \rho_m$ with $g_1 \prec \cdots \prec g_m$ and $\rho_1 \subseteq \cdots \subseteq \rho_m$. This is done by induction. Suppose $g_1, \ldots, g_i$ and $\rho_1, \ldots, \rho_i$ have already been constructed. The choice of g_{i+1} depends on what rule is applied to g_i in the construction of the tableau $\mathcal{G}$:

Case A. Some rule operating on formulas is applied to g_i.

Case A.1. $\longrightarrow_{\mathcal{U}}^{t}$ is applied to $\varphi \mathcal{U} \psi \in l(g_i)$. Then $\mathfrak{M}, n+1 \models \psi$ or $\mathfrak{M}, n+1 \models \varphi, \bigcirc(\varphi \mathcal{U} \psi)$. If the first appears, let g_{i+1} be the node with $g_i \prec g_{i+1}$ and $l(g_{i+1}) = \{\psi\} \cup \{(\varphi \mathcal{U} \psi)^*\} \cup (l(g_i) \setminus \{\varphi \mathcal{U} \psi\})$. Otherwise take g_{i+1} with $g_i \prec g_{i+1}$ such that $l(g_{i+1}) = \{\varphi, \bigcirc(\varphi \mathcal{U} \psi)\} \cup \{(\varphi \mathcal{U} \psi)^*\} \cup (l(g_i) \setminus \{\varphi \mathcal{U} \psi\})$. Put $\rho_{i+1} = \rho_i$.

Case A.2. $\longrightarrow_{\neg \mathcal{U}}$ is applied to $\neg(\varphi \mathcal{U} \psi) \in l(g_i)$. Then $\mathfrak{M}, n+1 \models \neg\psi, \neg\varphi$ or $\mathfrak{M}, n+1 \models \neg\psi, \neg \bigcirc (\varphi \mathcal{U} \psi)$. If the first appears, let g_{i+1} be the node with $g_i \prec g_{i+1}$ and $l(g_{i+1}) = l(g_i) \cup \{\neg\psi, \neg\varphi\}$. Otherwise let g_{i+1} be the node with $g_i \prec g_{i+1}$ and $l(g_{i+1}) = l(g_i) \cup \{\neg\psi, \neg\bigcirc(\varphi \mathcal{U} \psi)\}$. Put $\rho_{i+1} = \rho_i$.

Case A.3. The deterministic cases where one of the rules $\longrightarrow_{\neg\bigcirc}$, $\longrightarrow_{\neg\neg}$, or $\longrightarrow_{\wedge}$ is applied to g_i are considered in the obvious manner.

Case A.4. $\longrightarrow_{\neg\wedge}$ is applied to $\neg(\varphi \wedge \psi) \in l(g_i)$. Let g_{i+1} be the node with $g_i \prec g_{i+1}$ and $l(g_{i+1}) = l(g_i) \cup \{\neg\varphi\}$ if $\mathfrak{M}, n+1 \models \neg\varphi$. Otherwise let g_{i+1} be the node with $g_i \prec g_{i+1}$ and $l(g_{i+1}) = l(g_i) \cup \{\neg\psi\}$. Let $\rho_{i+1} = \rho_i$.

Case B. Some rule operating on concepts is applied to g_i in the construction of $\mathcal{G}$.

Case B.1. $\longrightarrow_{\mathcal{U}c}^{t}$ is applied to $(x : C\mathcal{U}D) \in l(g_i)$. Then $\rho_i(x) \in D^{I,n+1}$ or $\rho_i(x) \in C^{I,n+1}$ and $\rho_i(x) \in (\bigcirc(C\mathcal{U}D))^{I,n+1}$. If the first appears, take g_{i+1} such that $g_i \prec g_{i+1}$ and $l(g_{i+1}) = \{(x : D)\} \cup \{(x : (C\mathcal{U}D))^*\} \cup (l(g_i) \setminus \{x : C\mathcal{U}D\})$. Otherwise take g_{i+1} with $g_i \prec g_{i+1}$ and $l(g_{i+1}) = \{x : C, x : \bigcirc(C\mathcal{U}D)\} \cup \{(x : (C\mathcal{U}D))^*\} \cup (l(g_i) \setminus \{x : C\mathcal{U}D\})$. Put $\rho_{i+1} = \rho_i$.

Case B.2. $\longrightarrow_{\neg\mathcal{U}c}$ is applied to $x : \neg(C\mathcal{U}D) \in l(g_i)$. Then $\rho_i(x) \in (\neg D)^{I,n+1}$ and $\rho_i(x) \in (\neg C)^{I,n+1}$ or $\rho_i(x) \in (\neg D)^{I,n+1}$ and $\rho_i(x) \in (\neg \bigcirc (C\mathcal{U}D))^{I,n+1}$. If the first appears, take g_{i+1} with $g_i \prec g_{i+1}$ and $l(g_{i+1}) = l(g_i) \cup \{x : \neg D, x : \neg C\}$. Otherwise take g_{i+1} with $g_i \prec g_{i+1}$ and $l(g_{i+1}) = l(g_i) \cup \{x : \neg D, x : \neg \bigcirc (C\mathcal{U}D)\}$. Let $\rho_{i+1} = \rho_i$.

Case B.3. The cases in which one of the rules $\longrightarrow_{\neg\bigcirc c}, \longrightarrow_{\neg\neg c}, \longrightarrow_{\sqcap}, \longrightarrow_{\neg\sqcap}$, or $\longrightarrow_{=}$ is applied to g_i are considered in the obvious manner.

Case C. Some locally generating rule is applied to g_i in the construction of $\mathcal{G}$.

Case C.1. $\longrightarrow_{\neq}$ is applied to $C \neq \top \in l(g_i)$. Then we have $\mathfrak{M}, n+1 \models C \neq \top$ and so we can take $e \in \Delta^{n+1}$ with $e \in (\neg C)^{I,n+1}$. We know that $l(g_{i+1}) = l(g_i) \cup \{w : \neg C\}$, for some new variable w. Put $\rho_{i+1} = \rho_i \cup \{(w, e)\}$.

Case C.2. $\longrightarrow_{\exists}$ is applied to $(x : \exists R.C) \in l(g_i)$. We have $\rho_i(x) \in (\exists R.C)^{I,n+1}$. Hence we find $e \in \Delta^{n+1}$ with $\rho_i(x) R^{I,n+1} e$ and $e \in C^{I,n+1}$. We know that $l(g_{i+1}) = l(g_i) \cup X$ (as it is defined in the rule). Let w be the variable from X which does not occur in $l(g_i)$. Put $\rho_{i+1} = \rho_i \cup \{(w, e)\}$.

Suppose now that we have reached some m so that no non-globally generating rule is applicable to g_m in $\mathcal{G}$. Lemma 3.5 guarantees the existence of m. We show that the constructed sequences $g_1, \ldots, g_m$ and $\rho_1, \ldots, \rho_m$ are as required. Firstly, it is clear from the construction that $g_i \prec g_{i+1}$, for all $i < m$, and that $l(g_m)$ is the only saturated node in this list. Moreover, $\rho_i \subseteq \rho_{i+1}$ is also clear and $l(g_i) \subseteq^{\rho_i}_{n+1} \mathfrak{M}$ is easily proved by induction on i. It remains to verify the conditions (a)-(c).

(a) Suppose $\varphi\mathcal{U}\psi \in l(g_m)$ and $\mathfrak{M}, n+1 \models \psi$. The rule $\longrightarrow^t_{\mathcal{U}}$ is not applicable to g_m, so $\varphi\mathcal{U}\psi$ is marked in g_m. But $\varphi\mathcal{U}\psi$ is not marked in g_1 and so there exists $i < m$ such that $\longrightarrow^t_{\mathcal{U}}$ has been applied to $\varphi\mathcal{U}\psi$ in g_i. But then, by the above procedure, $\psi \in l(g_{i+1}) \subseteq l(g_m)$.

(b) Suppose $(x : C\mathcal{U}D) \in l(g_m)$ and $\rho_m(x) \in D^{I,n+1}$. The rule $\longrightarrow^t_{\mathcal{U}c}$ is not applicable to g_m hence $(x : C\mathcal{U}D)$ is marked in g_m. But $(x : C\mathcal{U}D)$ is not marked in g_1 and so, again, there exists $i < m$ such that $\longrightarrow_{\mathcal{U}c}$ has been applied to $(x : C\mathcal{U}D)$ in $l(g_i)$. But then $(x : D) \in l(g_{i+1}) \subseteq l(g_m)$.

(c) follows from the definition.

This completes the proof of the first claim of the Lemma.

The proof of the second claim can be conducted in precisely the same manner as the proof above starting from g_1. It is left to the reader. $\dashv$

Now we are ready for proving Lemma 7.2.

Proof. Suppose ϑ is satisfiable. We find $\mathfrak{M} = (\mathbb{N}, <, I)$ with

$$I(n) = (\Delta^n, R_0^{I,n}, \ldots, C_0^{I,n}, \ldots, a_0^{I,n}, \ldots),$$

for all $n \in \mathbb{N}$, and $\mathfrak{M}, 0 \models \vartheta$. We show the following

Claim. There exists a sequence of intervals $\{[n(i), m(i)] \mid i \in \mathbb{N}\}$ of natural numbers with $n(0) = 0$ and $m(i) + 1 = n(i+1)$, for all $i \in \mathbb{N}$, such that there exist sequences

- $\vec{g} = (g_n \mid n \in \mathbb{N})$ of nodes in G with $g_0 = g^r$, and
- $\vec{\rho} = (\rho_n \mid n \in \mathbb{N})$ of mappings

such that the following hold:

(c1) for all $n \in \mathbb{N}$: $g_n \overline{\prec} g_{n+1}$, (c2) for all $n \in \mathbb{N}$: $l(g_n)$ is saturated iff there exists $i \in \mathbb{N}$ with $n = m(i)$,

(c3) for all $i \in \mathbb{N}$ and $n \in [n(i), m(i) - 1]$: $\rho_n \subseteq \rho_{n+1}$,

(c4) for all $i \in \mathbb{N}$ and $n \in [n(i), m(i)]$: $l(g_n) \subseteq_i^{\rho_n} \mathfrak{M}$,

(c5) for all $i \in \mathbb{N}$ and all $\varphi \mathcal{U} \psi \in l(g_{m(i)})$, if $\mathfrak{M}, i \models \psi$ then $\psi \in l(g_{m(i)})$,

(c6) for all $i \in \mathbb{N}$ and all $(x : C\mathcal{U}D) \in l(g_{m(i)})$, if $\rho_{m(i)}(x) \in D^{I,i}$ then $(x : D) \in l(g_{m(i)})$,

(c7) for all $n \in \mathbb{N}$ every eventuality for g_n is realized by $\vec{g}$.

Because of (c4) all $l(g_n)$ are clash-free. Hence, by (c7), the sequence $(g_n \,|\, n \in \mathbb{N})$ is as required for proving Lemma 7.2.

The construction of the sequences is by induction. We start with g_0 and ρ_0: let $\rho_0(a) = a^{I,0}$, for all $a \in ob(\vartheta)$, and put $\rho_0(min(V)) \in \Delta^0$, arbitrary. Obviously $l(g_0) \subseteq_0^{\rho_0} \mathfrak{M}$, for $g_0 = g^r$.

Suppose now that we have constructed a sequence of intervals $\{[n(i), m(i)] \,|\, i \leq k\}$ and sequences

- $(g_n \,|\, n \leq m(k))$ and $(\rho_n \,|\, n \leq m(k))$

satisfying the conditions (c1)-(c6) stated in the claim above until $m(k)$ — save that $l(g_{m(k)})$ is possibly non-saturated. In the latter case, using Lemma 7.3 (2), it is straightforward to extend these sequences by means of nodes $g_{m(k)+1}, \ldots, g_{m(k)+l}$ and mappings $\rho_{m(k)+1}, \ldots, \rho_{m(k)+l}$ such that $l(g_{m(k)+l})$ is saturated and the extended sequence still has all properties (c1)-(c6). So, we can assume without loss of generality that $l(g_{m(k)})$ is saturated. We now distinguish two cases.

Case 1. There exists an eventuality for some $l(g_n)$, $n \leq m(k)$, which is not realized by $(g_n \,|\, n \leq m(k))$ until $m(k)$.

Take a minimal $l \leq m(k)$ such that there exists an eventuality α for $l(g_l)$ which is not realized until $m(k)$. To begin with suppose α is of the form $(v : C'\mathcal{U}D')$. We take $v_l, \ldots, v_{m(k)}$ with $v_l = v$ such that, for all $i \in [l, m(k)]$,

- v_i occurs in $l(g_i)$,
- $v_i = v_{i+1}$, whenever $i \leq m(k)$ and g_i is non-saturated,
- $\{C \,|\, (v_i : \bigcirc C) \in l(g_i\} \subseteq \{C | (v_{i+1} : C) \in l(g_{i+1})\}$, whenever $l(g_i)$ saturated.

Assume $\rho_{m(k)}(v_{m(k)}) = d \in \Delta^k$. We have $(v_{m(k)} : C'\mathcal{U}D') \in l(g_{m(k)})$ and $(v_{m(k)} : D') \notin l(g_{m(k)})$, since $(v : C'\mathcal{U}D')$ is not realized until $m(k)$. Hence $d \in (C'\mathcal{U}D')^{I,k}$, by (c4), and $d \notin (D')^{I,k}$, by (c6). There exists $n > k$ such that $d \in (D')^{I,n}$. Assume that n is minimal with this property.

Iterated application of Lemma 7.3 (1) provides intervals $\{[n(j), m(j)] \,|\, k+1 \leq j \leq n\}$ with $n(k+1) = m(k)+1$ and $m(j)+1 = n(j+1)$ for all $j \in [k+1, n-1]$, and sequences

- $g_{n(k+1)} \overline{\prec} \ldots \overline{\prec} g_{m(n)}$ and $\rho_{n(k+1)}, \ldots, \rho_{m(n)}$

such that (c1)-(c6) above hold until $m(n)$, and there is a sequence of variables $v_{n(k+1)}, \ldots, v_{m(n)}$, v_j from $l(g_j)$ for all $j \in [n(k+1), m(n)]$ such that the following hold for all $j \in [n(k+1), m(n)]$:

1. $\rho_j(v_j) = d$,

2. $v_j = v_{j+1}$, when g_j is non-saturated,

3. $\{C \mid (v_{m(i)} : \bigcirc C) \in l(g_{m(i)}\} \subseteq \{C \mid (v_{n(i+1)} : C) \in l(g_{n(i+1)})\}$, for all $i \in [k, n-1]$.

We show that $(v : C'\mathcal{U}D')$ is realized for g_l by $(g_j \mid 0 \leq j \leq m(n))$ and the sequence

$$v_l, \ldots, v_{m(k)}, v_{n(k+1)}, \ldots, v_{m(n)}.$$

To this end it suffices to show that $(v_{m(n)} : D') \in l(g_{m(n)})$. But, by the minimality of n, we have $(v_j : C'\mathcal{U}D') \in l(g_j)$ for all $j \in [l, m(n)]$. Moreover, $\rho_{m(n)}(v_{m(n)}) \in (D')^{I,n}$. Using (c6) we infer $(v_{m(n)} : D') \in l(g_{m(n)})$. This concludes the case where α has the form $(v : (C'\mathcal{U}D'))$.

The cases where the eventuality α is of the form $(a : C'\mathcal{U}D')$ or $\varphi\mathcal{U}\psi$ are treated similarly and can be left to the reader.

Case 2. Every eventuality for every g_n, $n \leq m(k)$, is realized by $(g_n \mid n \leq m(k))$ until $m(k)$.

This case is easier than the first one since no eventuality has to be realized. We just take the unique pre-state g' with $l(g_{m(k)})\overline{\prec}l(g')$ and add it to the list $(g_n \mid n \leq m(k))$. The required mapping ρ' showing $l(g') \subseteq^{\rho'}_{k+1} \mathfrak{M}$ is obtained by putting, for every object name a in $l(g')$, $\rho'(a) = \rho_{m(k)}(a)$, and for every variable w in $l(g')$, $\rho'(w) = \rho_{m(k)}(\pi^{-1}(w))$, where π is the renaming with $l(g_{m(k)}) \longrightarrow^{\pi}_{\bigcirc} l(g')$.

Now, in the limit we obtain sequences $(g_n \mid n \in \mathbb{N})$ and $(\rho_n | n \in \mathbb{N})$ which obviously satisfy the conditions (c1)-(c6). (c7) can be shown as follows: suppose α is an eventuality in $l(g_m)$. Then, since the number of eventualities in each $l(g_n)$ is finite, eventually α will be realized in the construction of $(g_n \mid n \in \mathbb{N})$. $\dashv$

8 Completeness

Theorem 8.1 *Let ϑ be some $\mathcal{TL}_{ALC}$-formula. If there exists a completion of $\{\vartheta\}$ in which the root is not eliminated then ϑ is satisfiable.*

Proof. Assume $\mathcal{G} = (G, \prec, l)$ is some complete tableau for ϑ the root g^r of which is not eliminated. Let $B \subseteq G$ be the set of nodes which remain after the execution of the elimination procedure. We have $g^r \in B$. We must show that ϑ is satisfiable. By Theorem 4.5 it is sufficient to prove the existence of some quasimodel that satisfies ϑ. In order to do this we are going to show the following

Claim. There exists a sequence of intervals $\{[n(i), m(i)] \mid i \in \mathbb{N}\}$ of natural numbers with $n(0) = 0$ and $m(i)+1 = n(i+1)$, for all $i \in \mathbb{N}$, such that there exists a sequence $\vec{g} = (g_n \mid n \in \mathbb{N})$ of nodes in B with $g_0 = g^r$ and the following holds for all $n \in \mathbb{N}$:

(d1) $g_n \overline{\prec} g_{n+1}$,

(d2) $l(g_n)$ is saturated iff there exists $i \in \mathbb{N}$ with $n = m(i)$,

(d3) every eventuality for g_n is realized by $\vec{g}$.

Suppose the sequence has been constructed. We then reason as follows: By assumption, (E1) is not applicable to B. Hence all $l(g_n)$ are clash-free. By utilizing (d1)-(d3) it is not difficult to show that $(l(g_{m(i)}) \mid i \in \mathbb{N})$ is a quasimodel satisfying ϑ. We leave the details to the reader.

The construction of the sequences is by induction. We start with $g_0 = g^r$. Suppose now that we have constructed a sequence of intervals $\{[n(i), m(i)] \mid i \leq k\}$ and sequences $(g_n \mid n \leq m(k))$ satisfying the conditions (d1)-(d2) stated in the claim above until $m(k)$ save that $l(g_{m(k)})$ is possibly non-saturated. Using the condition that (E2) cannot be applied to B it is straightforward to extend this sequence by means of nodes $g_{m(k)+1}, \ldots, g_{m(k)+l}$ from B such that $l(g_{m(k)+l})$ is a pre-state and the extended sequence still has properties (d1),(d2). So, we can assume without loss of generality that we have $\{[n(i), m(i)] \mid i \leq k\}$, a sequence $(g_n \mid n \leq m(k))$ satisfying (d1) and (d2), and a pre-state $g_{n(k+1)}$ with $g_{m(k)} \overline{\prec} g_{n(k+1)}$. We distinguish two cases.

Case 1. There exists an eventuality for some $l(g_n)$, $n \leq n(k+1)$, which is not realized by $(g_n \mid n \leq n(k+1))$ until $n(k+1)$.

Choose $l \leq n(k+1)$ minimal such that there exists an eventuality α for $l(g_l)$ which is not realized until $n(k+1)$. First suppose α is of the form $(v : C'\mathcal{U}D')$.

We take $v_l, \ldots, v_{n(k+1)}$ with $v_l = v$, $v_{n(k+1)}$ occurs in $l(g_{n(k+1)}$, and such that, for all $i \in [l, m(k)]$,

- v_i occurs in $l(g_i)$,
- $v_i = v_{i+1}$, whenever $l(g_i)$ is non-saturated,
- $\{C \mid (v_i : \bigcirc C) \in l(g_i\} \subseteq \{C \mid (v_{i+1} : C) \in l(g_{i+1})\}$, whenever $l(g_i)$ saturated.

We have $(v_{n(k+1)} : C'\mathcal{U}D') \in l(g_{n(k+1)})$ and $(v_{n(k+1)} : D') \notin l(g_{n(k+1)})$, since $(v : C'\mathcal{U}D')$ is not realized until $n(k+1)$. (E3) is not applicable to any node in B, in particular not to $g_{n(k+1)}$. So we find $\{[n(j), m(j)] \mid k+1 \leq j \leq n\}$ with $m(j) + 1 = n(j+1)$ for all $j \in [k+1, n-1]$, and sequences

- $g_{n(k+1)} \overline{\prec} \ldots \overline{\prec} g_{m(n)}$ of nodes in B and $\rho_{n(k+1)}, \ldots, \rho_{m(n)}$

such that (d1), (d2) above hold until $m(n)$, and there is a sequence of variables $v_{n(k+1)}, \ldots, v_{m(n)}$, v_j from $l(g_j)$ for all $j \in [n(k+1), m(n)]$, such that $(g_j \mid l \leq j \leq m(n))$ and $(v_j | l \leq j \leq m(n))$ realize $C\mathcal{U}D'$ until $m(n)$.

The cases where the eventuality α is of the form $(a : C'\mathcal{U}D')$ or $\varphi\mathcal{U}\psi$ are treated similarly and left to the reader.

Case 2. Every eventuality for some $l(g_n)$, $n \leq n(k+1)$, is realized by $(g_n \mid n \leq n(k+1))$ until $n(k+1)$.

In this case no eventuality has to be realized. Thus, using that (E2) is not applicable to B, we can extend the sequence $(g_n \mid n \leq g_{n(k+1)})$ by a new node $g_{n(k+1)+1} \in B$ with $g_{n(k+1)} \overline{\prec} g_{n(k+1)+1}$.

In the limit we obtain a sequence $\vec{g}$ satisfying (d1)-(d3). $\dashv$

9 Example

In this section we illustrate how the tableau calculus works in a concrete case. For this purpose, we show by means of our tableau calculus that the formula ξ that has been introduced in Section 2 as an counterexample to the bounded domain property, is indeed satisfiable. For the proof we just have to draw the graph of a fragment of a complete tableau with root $\Xi = \{\Box\varphi_1, \Box\varphi_2, \Box\varphi_3, \varphi_4\}$, where

- $\varphi_1 = ((\neg(C \sqcap \neg\Box C)) = \top)$,
- $\varphi_2 = (\exists R.\neg C = \top)$,
- $\varphi_3 = (\Diamond C = \top)$,
- $\varphi_4 = (\neg C \neq \top)$.

Put

- $\Gamma = \{\bigcirc\Box\varphi_i, \varphi_i, \Box\varphi_i : i \leq 3\}$,
- $a_x = \{x : C, x : \neg\neg C, x : \Box C, x : \neg\neg\Box C, x : \neg(C \sqcap \neg\Box C), x : \Diamond C, x : \exists R.\neg C, x : \bigcirc\Box C\}$,
- $b_x = \{x : \neg C, x : \Diamond C, x : \exists R.\neg C, x : \neg(C \sqcap \neg\Box C), x : \bigcirc\Diamond C\}$.

A few remarks should suffice to make the following picture understandable:

(i) The big black nodes represent saturated states, the white ones pre-states, and the remaining states are represented by small black nodes.

(ii) Most of the arrows drawn in this picture comprise applications of several rules.

(iii) We treat concepts of the form $\Box C$, $\Diamond C$ respectively formulas of the form $\Box\varphi$, $\Diamond\varphi$ as full expressions, not as metalinguistic abbreviations. Consequently, we make use of the derived rules (for concepts)

   - $S \longrightarrow_{\Diamond c} X \cup \{(x : \Diamond C)^*\} \cup (S \setminus \{x : \Diamond C\})$, if $x : \Diamond C$ occurs unmarked in S, and $X = \{x : C\}$ or $X = \{x : \bigcirc\Diamond C\}$.
   - $S \longrightarrow_{\Box c} S \cup X$, if $(x : \Box C) \in S$, $\{x : \bigcirc\Box C, x : C\} \not\subseteq S$, and $X = \{x : \bigcirc\Box C, x : C\}$,

   and the rule $\longrightarrow_{\Box}$ for formulas, where the latter was already introduced in Section 5.

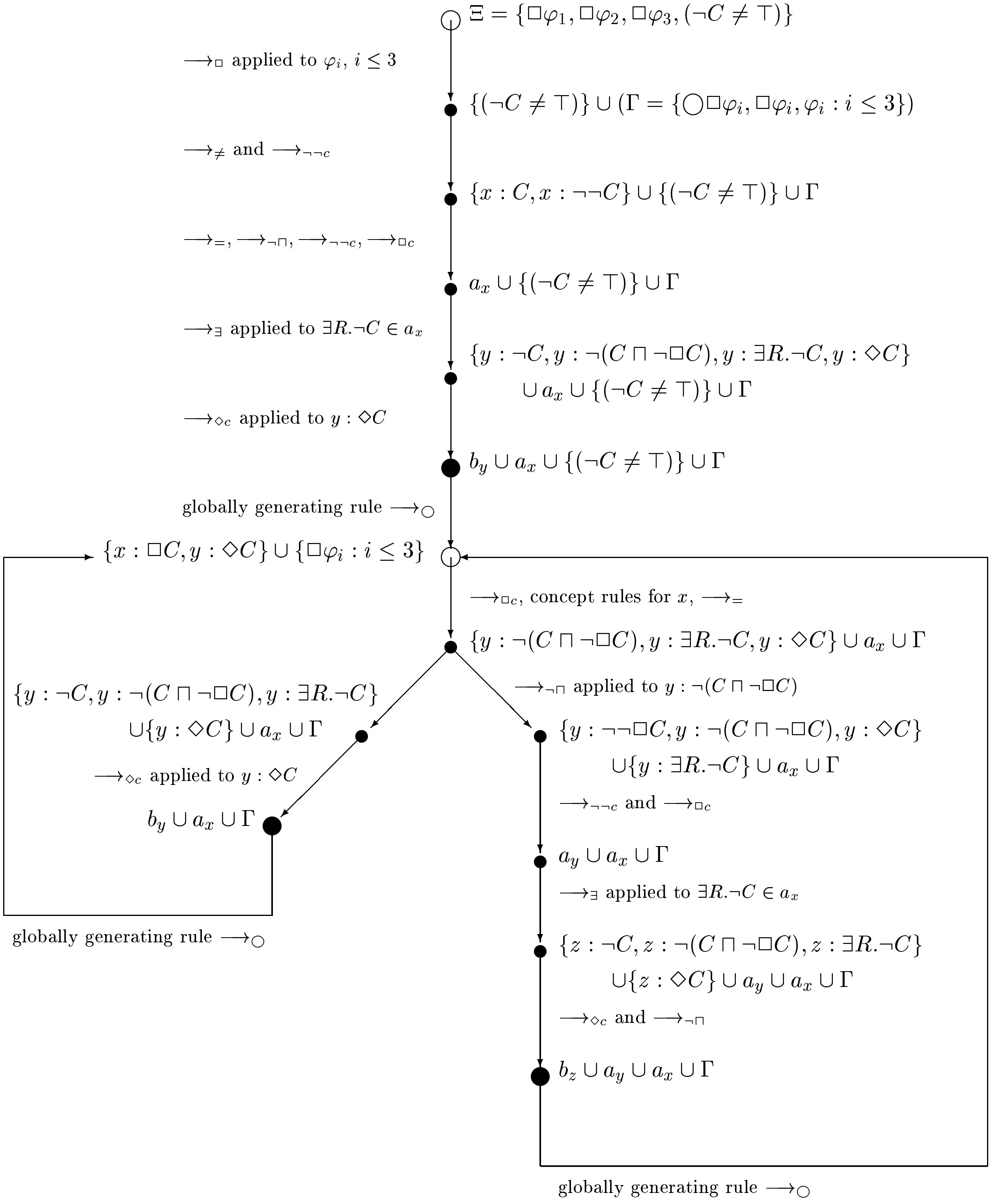
(iv) For convenience sake, we associate the root of the tableau with Ξ, and not, as it would be formally correct, with $\Xi \cup \{min(V) : \top\}$.

(v) For the same reason we ignore the distinction between marked and unmarked formulas. This concerns, in particular, applications of $\longrightarrow_{\Diamond c}$. For instance, in the first node on the right branch an application of this rule would lead to a second successor node containing $y : \bigcirc\Diamond C$, but not $y : C$. Accordingly, an application of $\longrightarrow_{\Diamond c}$ to the fourth node from the upper part of the graph would generate a successor node containing $x : \bigcirc\Diamond C$.

(vi) The picture shows only branches of the complete tableau that do not contain a clash. That is why, with one exception, the graph does not branch out though the non-deterministic rules $\longrightarrow_{\neg\sqcap}$ and $\longrightarrow_{\Diamond c}$ were applied several times. It is easy to see that, apart from the cases covered in (v), all the other branches that are not depicted finally end in a clash.

(vii) In the last node on the right branch — labelled with $b_z \cup a_y \cup a_x \cup \Gamma$ — further applications of $\longrightarrow_{\exists}$ are blocked due to the fact that b_z contains $z : \neg C$, $z : \neg(C \sqcap \neg\Box C)$, $z : \exists R : \neg C$, and $z : \Diamond C$. Accordingly, in the first saturated node as well as in the last node on the left branch $\longrightarrow_{\exists}$ is blocked due to the presence of y.

(viii) In order to build up a quasimodel for Ξ, it is sufficient to use, in addition to the nodes that belong to the upper part of the tableau, either nodes from the left part or nodes from the right part.


$\Xi = \{\Box\varphi_1, \Box\varphi_2, \Box\varphi_3, (\neg C \neq \top)\}$
$\longrightarrow_{\Box}$ applied to φ_i, $i \leq 3$
$\{(\neg C \neq \top)\} \cup (\Gamma = \{\bigcirc\Box\varphi_i, \Box\varphi_i, \varphi_i : i \leq 3\})$
$\longrightarrow_{\neq}$ and $\longrightarrow_{\neg\neg c}$
$\{x : C, x : \neg\neg C\} \cup \{(\neg C \neq \top)\} \cup \Gamma$
$\longrightarrow_{=}, \longrightarrow_{\neg\sqcap}, \longrightarrow_{\neg\neg c}, \longrightarrow_{\Box c}$
$a_x \cup \{(\neg C \neq \top)\} \cup \Gamma$
$\longrightarrow_{\exists}$ applied to $\exists R.\neg C \in a_x$
$\{y : \neg C, y : \neg(C \sqcap \neg\Box C), y : \exists R.\neg C, y : \Diamond C\}$
$\cup\, a_x \cup \{(\neg C \neq \top)\} \cup \Gamma$
$\longrightarrow_{\Diamond c}$ applied to $y : \Diamond C$
$b_y \cup a_x \cup \{(\neg C \neq \top)\} \cup \Gamma$
globally generating rule $\longrightarrow_{\bigcirc}$
$\{x : \Box C, y : \Diamond C\} \cup \{\Box\varphi_i : i \leq 3\}$
$\longrightarrow_{\Box c}$, concept rules for x, $\longrightarrow_{=}$
$\{y : \neg(C \sqcap \neg\Box C), y : \exists R.\neg C, y : \Diamond C\} \cup a_x \cup \Gamma$
$\{y : \neg C, y : \neg(C \sqcap \neg\Box C), y : \exists R.\neg C\}$
$\cup\{y : \Diamond C\} \cup a_x \cup \Gamma$
$\longrightarrow_{\Diamond c}$ applied to $y : \Diamond C$
$b_y \cup a_x \cup \Gamma$
globally generating rule $\longrightarrow_{\bigcirc}$
$\longrightarrow_{\neg\sqcap}$ applied to $y : \neg(C \sqcap \neg\Box C)$
$\{y : \neg\neg\Box C, y : \neg(C \sqcap \neg\Box C), y : \Diamond C\}$
$\cup\{y : \exists R.\neg C\} \cup a_x \cup \Gamma$
$\longrightarrow_{\neg\neg c}$ and $\longrightarrow_{\Box c}$
$a_y \cup a_x \cup \Gamma$
$\longrightarrow_{\exists}$ applied to $\exists R.\neg C \in a_x$
$\{z : \neg C, z : \neg(C \sqcap \neg\Box C), z : \exists R.\neg C\}$
$\cup\{z : \Diamond C\} \cup a_y \cup a_x \cup \Gamma$
$\longrightarrow_{\Diamond c}$ and $\longrightarrow_{\neg\sqcap}$
$b_z \cup a_y \cup a_x \cup \Gamma$
globally generating rule $\longrightarrow_{\bigcirc}$

10 Related Work

In this paper we have presented a tableau-based decision procedure for a temporalized description logic. As was mentioned in the introduction, this is not the first work in this direction. Very close to our work are [5, 25, 34] and we briefly relate their algorithms to the present one.

- Schild [25] considers a temporal description logic in which temporal operators can be applied to concepts but not to axioms. As in the present approach the flow of time consists of the natural numbers. Schild develops a tableau algorithm deciding the satisfiability of concepts. It is based on the tableau of [29]. However, in Schilds language there is no real interaction between the temporal operators and $\mathcal{ALC}$. This is reflected by the fact that the concept satisfiability problem is still EXPTIME-complete while the satisfiability problem for the language considered in the present paper is EXPSPACE-complete (Mosurovicz and Zakharyaschev: personal communication).

- Baader and Laux [5] develop a tableau-based decision procedure for the basic modal logic $\mathbf{K}$ based on $\mathcal{ALC}$ and interpreted in models with expanding domains. They allow the application of modal operators to concepts and axioms, as we do. On the other hand, the tableau in [5] is much simpler than the present one since neither quasimodels ($\mathbf{K}$ based on $\mathcal{ALC}$ has the finite model property and a complete clash-free constraint system is directly transformed into a finite model) nor modal "blocking" are required. Again this is reflected by the complexity of the decision problem which is known to be NEXPTIME-complete (Mosurovicz and Zakharyaschev: personal communication).

- Wooldridge, Dixon and Fisher [34] develop a tableau-based decision procedure for temporal logic based on epistemic modal logics like $\mathbf{S5}_n$. They also extend the propositional temporal tableaux of [17, 29]. In their temporal epistemic logic we do not have interaction between the temporal and epistemic operators which is reflected by the fact that the satisfiability problem for their system is still PSPACE-complete.

Finally, we should like to mention that [2] develops a decision procedure for temporal description logics the temporal part of which is based on intervals and not points (and so requires a technique completely different from what is developed in the present paper) and that [16] introduces and investigates a resolution method for temporalized epistemic logics.

11 Concluding remarks

The work presented in this paper is mainly conceptual: it is shown that the standard tableau-algorithms for linear propositional temporal logic and $\mathcal{ALC}$ can be combined in such a way that one obtains a tableau-algorithm for temporalized $\mathcal{ALC}$ *with interaction* between the two dimensions. In particular, we show how quasimodels can be used to bridge the gap between models provided by complete tableaux and the intended models for the language. The tableau presented here has the deficit that it works for expanding (and therefore varying) domains only but not for constant domain models. In a follow-up paper we are going to extend our tableau-method so as to give a solution to this problem. More generally, we believe that our method of combining tableaux can be fruitfully applied also for obtaining algorithms for various spatio-temporal logics in the sense of [33] and, even more generally, for monodic fragments of first-order logics in the sense of [20].

The work is mainly conceptual, of course, since it still requires some work to obtain an efficient "implementable" tableau from the one presented here. This is left for the future. It should be clear, however, that our tableau provides us with the basic ingredients required for such an algorithm.

References

[1] S. Abiteboul, L. Herr, and J. van den Bussche. Temporal versus first-order logic in query temporal databases. In *ACM Symposium on Principles of Database Systems*, pages 49–57, Montreal, Canada, 1996.

[2] A. Artale and E. Franconi. A computational account for a description logic of time and action. In *Proceedings of the fourth Conference on Principles of Knowledge Representation and Reasoning*, pages 3–14, Montreal, Canada, 1994. Morgan Kaufman.

[3] A. Artale and E. Franconi. Temporal description logics. In *Handbook of Time and Temporal Reasoning.* 1999. To appear.

[4] Alessandro Artale and Enrico Franconi. A temporal description logic for reasoning about actions and plans. *Journal of Artificial Intelligence Research (JAIR)*, (9), 1998.

[5] F. Baader and A. Laux. Terminological logics with modal operators. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 808–814, Montreal, Canada, 1995. Morgan Kaufman.

[6] C. Bettini. Time-dependent concepts: representation and reasoning using temporal description logics. *Data and Knowledge Engineering*, 22:1 – 38, 1997.

[7] A. Borgida. Description logics in data management. *IEEE Trans. on Knowledge and Data Engineering*, 7:671–682, 1995.

[8] M. Buchheit, F. Donini, and A. Schaerf. Decidable reasoning in terminological knowledge representation systems. *Journal of Artificial Intelligence Research*, 1:109 – 138, 1993.

[9] D. Calvanese, G. De Giacomo, and M. Lenzerini. On the decidability of query containment under constraints. In *Proc. of the 17th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS'98)*, pages 149–158, 1998.

[10] Diego Calvanese, Maurizio Lenzerini, and Daniele Nardi. Description logics for conceptual data modeling. In Jan Chomicki and Günter Saake, editors, *Logics for Databases and Information Systems*, pages 229–263. Kluwer Academic Publisher, 1998.

[11] T. Catarci and M. Lenzerini. Representing and using interschema knowledge in cooperative information systems. *J. of Intelligent and Cooperative Information Systems*, 2:375–398, 1993.

[12] J. Chomicki. Temporal query languages: a survey. In D. Gabbay and H.J. Ohlbach, editors, *Temporal Logic, First International Conference*, pages 506–534, Montreal, Canada, 1994. Springer-Verlag.

[13] R. Fagin, J. Halpern, Y. Moses, and M. Vardi. *Reasoning about Knowledge.* MIT Press, 1995.

[14] M. Finger and D. Gabbay. Adding a temporal dimension to a logic system. *Journal of Logic, Language and Information*, 2:203–233, 1992.

[15] M. Finger and D. Gabbay. Combining temporal logic systems. *Notre Dame Journal of Formal Logic*, 37:204–232, 1996.

[16] M. Fisher, M. Wooldridge, and C. Dixon. A resolution based proof method for temporal logics of knowledge and belief. In *Proceedings of the International Conference on Formal and Applied Practical Reasoning*, Bonn, Germany, 1996.

[17] G. Gough. *Decision Procedures for temporal logics.* PhD thesis, University of Manchester, 1984.

[18] J. Halpern and Y. Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54:319–379, 1992.

[19] J. Halpern and M. Vardi. The complexity of reasoning about knowledge and time, lower bounds. *Journal of Computer and System Sciences*, 38:195 – 237, 1989.

[20] I. Hodkinson, F. Wolter, and M. Zakharyaschev. Fragments of first-order temporal logics. *Annals of Pure and Applied Logic*, 2000. To appear.

[21] I. Horrocks. *Optimising tableaux decision procedures for description logics.* PhD thesis, University of Manchester, 1997.

[22] I. Horrocks and U. Sattler. A description logic with transitive and inverse roles and role hierarchies. *Journal of Logic and Computation*, 9(3):385–410, 1999.

[23] F. Kröger. On the interpretability of arithmetic in temporal logic. *Journal of Theoretical Computer Science*, 73:46–60, 1990.

[24] S. Merz. Decidability and incompleteness results for first-order temporal logics of linear time. *Journal of Applied Non-classical Logic*, 2, 1992.

[25] K. Schild. Combining terminological logics with tense logic. In *Proceedings of the 6th Portuguese Conference on Artificial Intelligence*, pages 105–120, Porto, 1993.

[26] A. Schmiedel. A temporal terminological logic. In *Proceedings of the 9th National Conference of the American Association for Artificial Intelligence*, pages 640–645, Boston, 1990.

[27] E. Spaan. *Complexity of Modal Logics.* PhD thesis, Department of Mathematics and Computer Science, University of Amsterdam, 1993.

[28] A. Szalas and L. Holenderski. Incompleteness of first-order temporal logic with until. *Theoretical Computer Science*, 57:317–325, 1988.

[29] P. Wolper. The tableau method for temporal logic: an overview. *Logique et Analyse*, 110 -111:119 – 136, 1985.

[30] F. Wolter and M. Zakharyaschev. Satisfiability problem in description logics with modal operators. In *Proceedings of the fifth Conference on Principles of Knowledge Representation and Reasoning*, Montreal, Canada, 1998. Morgan Kaufman.

[31] F. Wolter and M. Zakharyaschev. Modal description logics: modalizing roles. *Fundamentae Informaticae*, 39:411 – 438, 1999.

[32] F. Wolter and M. Zakharyaschev. Temporalizing description logic. In D. Gabbay and M. de Rijke, editors, *Frontiers of Combining Systems*, pages 379 – 402. Studies Press/Wiley, 1999.

[33] F. Wolter and M. Zakharyaschev. Spatio-temporal representation and reasoning based on RCC-8. In *Proceedings of the sixth Conference on Principles of Knowledge Representation and Reasoning*, Montreal, Canada, 2000. Morgan Kaufman.

[34] M. Wooldridge, C. Dixon, and M. Fisher. A tableau-based proof method for temporal logics of knowledge and belief. *Journal of Applied Non-classical Logics*, 8:225 – 258, 1998.

[35] G.T. Wright, E.S. Weixelbaum, G.T. Vesonder, K.E. Brown, S.R. Palmer, J.I. Berman, and H.H. Moore. A knowledge-based configurator that supports sales, engineering, and manufacturing at AT&T network systems. *AI Magazine*, 14:69–80, 1993.