A Tableau Calculus for a Temporal Logic with Temporal Connectives

Wolfgang May

Institut für Informatik, Universität Freiburg, Germany,
may@informatik.uni-freiburg.de

Abstract. The paper presents a tableau calculus for a linear time temporal logic for reasoning about processes and events in concurrent systems. The logic is based on temporal connectives in the style of Transaction Logic [BK94] and explicit quantification over states. The language extends first-order logic with sequential and parallel conjunction, parallel disjunction, and temporal implication. Explicit quantification over states via state variables allows to express temporal properties which cannot be formulated in modal logics.
Using the tableau representation of temporal Kripke structures presented for CTL in [MS96] which represents states by prefix terms, explicit quantification over states is integrated into the tableau calculus by an adaptation of the δ-rule from first-order tableau calculi to the linear ordering of the universe of states.
Complementing the CTL calculus, the paper shows that this tableau representation is both suitable for *modal temporal logics* and for logics using *temporal connectives*.

1 Introduction

When extending first-order logic to temporal logic, most approaches are based on *modal operators*, such as LTL/CTL or Dynamic Logic. Here, formulas are modified via modalities – inducing an implicit quantification over states. Formulas are evaluated wrt. states or (infinite) paths, thus they do not support an intuitive notion of sequentiality or parallelism.

For reasoning about processes and events in concurrent systems, *temporal connectives* such as sequential, parallel, and alternative composition or iteration are well-known from process algebraic formalisms. First-order-logic based formalisms using temporal connectives (which implies evaluating formulas wrt. finite path segments) are rare, although they have obvious advantages when reasoning about temporal behavior of processes. For Transaction Logic [BK94], it has been shown how to write executable specifications in such a formalism.

There are some temporal constraints which cannot be expressed in temporal modal logics, e.g., that "if some state is reached such that a given predicate p has the same extension as now, then q holds in this state" (cf. [TN96,CT98]). In [CT98], it is shown that this can be expressed in 2-FOL which is a two-sorted first order language for dealing with a linear temporal state space by

$$\forall s_1, s_2 : (\forall x : p(x, s_1) \leftrightarrow p(x, s_2) \wedge s_1 < s_2) \rightarrow q(s_2)\ .$$

This example motivates that an explicit quantification and addressing of states via state variables would be useful in a temporal logic.

In [MS96], a tableau semantics for first-order Kripke structures has been presented, together with a tableau calculus for first-order CTL. There, states have been described by *prefix terms* which provide a natural way to adapt the γ and δ-rule to quantification by state variables.

In the present paper, it is shown how the same approach applies for this temporal logic based on *temporal connectives* and explicit quantification over states.

The paper is structured as follows: After introducing some basic notions in Section 2, a (linear-time) logic for formulating complex events and dynamic constraints is presented in Section 3. Section 4 contains the tableau semantics for linear Kripke structures, and the tableau calculus is given in Section 5. Section 6 closes with some concluding remarks.

Related Work. Most of the work in Temporal Logics focuses on modal logics, e.g., CTL, modal μ-calculus, or Dynamic Logic. An overview of tableau calculi for (modal) temporal logics have been summarized in [Wol85], a recent one is described in [MP95]. *Interval Logics* contain operators for sequential composition and iteration similar to those known from programming languages [Mos86]. A tableau method for interval logic has, e.g., been presented in [BT98]. Other formalisms for expressing temporal constraints in non-modal logics are dealt with in [Sin95], [BK94], [Pra90], [Jab94], and [TN96,CT98].

2 Basic Notions

Let Σ be a *signature* consisting of a set Σ_{func} of function symbols a set Σ_{pred} of predicate symbols with fixed arities $\mathsf{ord}(f)$ resp. $\mathsf{ord}(p)$, and $\mathsf{Var} := \{x_1, x_2, \dots\}$ an infinite set of variables. Let Term_Σ denote the set of terms over Σ and Var. The notions of bound and free variables are defined as usual, $\mathsf{free}(\mathcal{F})$ denoting the set of variables occurring free in a set $\mathcal{F}$ of formulas.

A *substitution* (over a signature Σ) is a mapping $\sigma : \mathsf{Var} \to \mathsf{Term}_\Sigma$ where $\sigma(x) \neq x$ for only finitely many $x \in \mathsf{Var}$, here denoted by $[\sigma(x)/x]$. Substitutions are extended to terms and formulas as usual.

A first-order structure $\boldsymbol{I} = (I, \boldsymbol{U})$ over a signature Σ consists of a universe $\boldsymbol{U}$ and a *first-order interpretation* I of Σ which maps every function symbol $f \in \Sigma$ to a function $I(f) : \boldsymbol{U}^{\mathsf{ord}(f)} \to \boldsymbol{U}$ and every predicate symbol $p \in \Sigma$ to a relation $I(p) \subseteq \boldsymbol{U}^{\mathsf{ord}(p)}$.

A *variable assignment* is a mapping $\chi : \mathsf{Var} \to \boldsymbol{U}$. For a variable assignment χ, a variable x, and $d \in \boldsymbol{U}$, the *modified* variable assignment χ_x^d is identical with χ except that it assigns d to x. Let Ξ denote the set of variable assignments.

Every interpretation induces an *evaluation* $\boldsymbol{I} : \mathsf{Term}_\Sigma \times \Xi \to \boldsymbol{U}$ s.t. $\boldsymbol{I}(x, \chi) := \chi(x)$ for $x \in \mathsf{Var}$, and $\boldsymbol{I}(f(t_1, ..., t_n), \chi) := (I(f))(\boldsymbol{I}(t_1, \chi), ..., \boldsymbol{I}(t_n, \chi))$ for $f \in \Sigma$, $\mathsf{ord}(f) = n$ and $t_1, ..., t_n \in \mathsf{Term}_\Sigma$. The truth of a formula F in a first-order structure $\boldsymbol{I}$ under a variable assignment χ, $(\boldsymbol{I}, \chi) \models F$ is defined as usual.

3 Temporal Connectives and Quantification

Temporal specifications consist of combining subformulas by *temporal connectives*. The logic presented here incorporates the following facilities:

– $\otimes$ denotes sequential conjunction via subsequent path segments and $\times$ denotes a parallel conjunction on the same path segment.
– $\mathrm{E}^\star$ denotes finite iteration, mostly limited by a first-order condition on a state which is tested between each two iterations.
– Constraints are specified by constraint formulas in an if-then-style. In temporal context, there are three possible implication constructs (if ... then before/later/sometimes ...). With a suitable synchronization formalism, they can be formulated by a single causal implication $\mathrm{E} \rightsquigarrow \mathrm{F}$.
– Via temporal quantification, formulas introduce "private" synchronization points which represent agreements local to processes.

From the modeling point of view, synchronization points are *virtual* entities. This idea shows some similarities with first-order existential quantification of a variable x: there is a local agreement (binding), which entity is meant by x, without identifying it extensionally. This information is kept local to the scope of the quantifier. Following this idea, synchronization points are handled via a set SVar of *state variables* s_i which can be bound to states.

The logic is interpreted by linear first-order Kripke structures which are augmented by a *transition oracle* (cf. [BK94]) representing the actions which are executed in the state transitions. A *linear first-order Kripke structure* over a signature Σ is a pair $\boldsymbol{K} = (\boldsymbol{U}, \boldsymbol{M})$, where $\boldsymbol{U}$ is a universe and $\boldsymbol{M}$ is a mapping from the natural numbers to first-order interpretations $\boldsymbol{M}(n)$ over Σ. Since a constant universe is presumed, the notion of a *variable assignment* is defined as in the first-order case. For every transition from n to $n{+}1$, the *transition oracle* yields an interpretation $\boldsymbol{N}(n)$ of a set Σ_A of *action symbols* similar to predicates. Having $(u_1, \dots u_n) \in (\boldsymbol{N}(n))(a)$ for $a \in \Sigma_A$ in the transition oracle means that $a(u_1, \dots, u_n)$ is executed in the transition from n to $n{+}1$.

Definition 1 (Action Formulas). Action formulas *are first-order formulas over the signature* $\Sigma_A \cup \Sigma_{func}$. *Action formulas are evaluated wrt.* transitions $\boldsymbol{K}\langle i, i{+}1\rangle$ *as* $(\boldsymbol{K}\langle i, i{+}1\rangle, \chi) \models a(t_1, \dots, t_n) \;\Leftrightarrow\; (\boldsymbol{M}(i)(t_1, \chi), \dots, \boldsymbol{M}(i)(t_n, \chi)) \in \boldsymbol{N}(n)(a)$, *using the transition oracle.*

Definition 2 (ECL-Event Formulas). *The language of* ECL formulas *over a first-order signature* Σ *and an action signature* Σ_A *is defined inductively. With every ECL formula* E, *a* length $\mathsf{len}(\mathrm{E}) \subseteq \mathbb{N} \cup \{\infty\}$, *is associated; the addition "+" on subsets of* $\mathbb{N} \cup \{\infty\}$ *is defined as* $N{+}M := \{k : \exists n \in N, m \in N \mid k = n{+}m\}$.

1. *Every first-order formula* φ *is an ECL formula of length* $\mathsf{len}(\varphi) = \{0\}$.
2. ∂ *and* $\bot$ *are ECL formulas with length* $\mathsf{len}(\partial) = \mathsf{len}(\bot) = \mathbb{N} \cup \{\infty\}$. ∂ *denotes idling for an arbitrary time,* $\bot$ *denotes an action which can never be executed successfully (often called* deadlock*).*

3. *Every action formula* E *is an ECL formula of length* $\mathsf{len}(\mathrm{E}) = \{1\}$.
4. *With* s *a state variable and* E *an ECL formula such that* s *is not free in* E,
   - $s \rhd \mathrm{E}$ *is an ECL formula with* $\mathsf{len}(s \rhd \mathrm{E}) = \mathsf{len}(\mathrm{E})$, *and*
   - *if* $\mathsf{len}(\mathrm{E}) \cap \mathbb{N} \neq \emptyset$, $\mathrm{E} \lhd s$ *is an ECL formula with* $\mathsf{len}(\mathrm{E} \lhd s) = \mathsf{len}(\mathrm{E}) \cap \mathbb{N}$.

   Then, s *is free in* $s \rhd \mathrm{E}$ *and* $\mathrm{E} \lhd s$ *(prefixing or postfixing with a synchronization point).*
5. *With* E *an ECL formula with* $\mathsf{len}(\mathrm{E}) \cap \mathbb{N} \neq \emptyset$ *and* F *an ECL formula,* $\mathrm{E} \otimes \mathrm{F}$ *is an ECL formula,* $\mathsf{len}(\mathrm{E} \otimes \mathrm{F}) = (\mathsf{len}(\mathrm{E}) \cap \mathbb{N}) + \mathsf{len}(\mathrm{F})$ *(sequential composition).*
6. *With* E *and* F *ECL formulas with* $\mathsf{len}(\mathrm{E}) \cap \mathsf{len}(\mathrm{F}) \neq \emptyset$, $\mathrm{E} \times \mathrm{F}$ *is an ECL formula with* $\mathsf{len}(\mathrm{E} \times \mathrm{F}) = \mathsf{len}(\mathrm{E}) \cap \mathsf{len}(\mathrm{F})$ *(parallel composition).*
7. *With* $\mathrm{E}_1, \ldots, \mathrm{E}_n$ *ECL formulas,* $\sum_{1 \leq i \leq n} \mathrm{E}_i$ *is an ECL formula with* $\mathsf{len}(\sum_{1 \leq i \leq n} \mathrm{E}_i) = \bigcup_{1 \leq i \leq n} \mathsf{len}(\mathrm{E}_i)$ *(alternative composition).*
8. *With* E *an ECL formula,* $\mathrm{E}^\star$ *is an ECL formula with* $\mathsf{len}(\mathrm{E}^\star) = \{n \mid \exists k \in \mathbb{N}, n_1, \ldots, n_k \in \mathsf{len}(\mathrm{E})$ *and* $n = n_1 + \ldots + n_k\}$ *(finite iteration).*
9. *With* E *an ECL formula and* x *a variable,* $\forall x : \mathrm{E}$ *and* $\exists x : \mathrm{E}$ *are ECL formulas,* $\mathsf{len}(\forall x : \mathrm{E}) = \mathsf{len}(\exists x : \mathrm{E}) = \mathsf{len}(\mathrm{E})$.
10. *With* s *a state variable and* E *an ECL formula,* $\blacklozenge s : \mathrm{E}$ *and – if* $\infty \in \mathsf{len}(\mathrm{E})$ *–* $\blacksquare s : \mathrm{E}$ *are ECL formulas,* $\mathsf{len}(\blacklozenge s : \mathrm{E}) = \mathsf{len}(\mathrm{E})$, $\mathsf{len}(\blacksquare s : \mathrm{E}) = \{\infty\}$ *(introducing a state variable).*

- *Each of the above ECL formulas is an ECL-*event *formula.*

ECL-event formulas hold on a segment of a Kripke structure. For a linear Kripke structure $\boldsymbol{K}$ and $i \leq j$ such that $i \in \mathbb{N}$, $j \in \mathbb{N} \cup \{\infty\}$, the expression $\boldsymbol{K}[i,j]$ denotes the whole structure, with the *focus* on the segment $[i,j]$.

Definition 3 (Assignment of State Variables). *For a linear Kripke structure, an* assignment ζ of state variables *is a function which maps every state variable to a state of* $\boldsymbol{K}$*, i.e., a natural number.*
The modification of an assignment ζ *of state variables at one element is defined in the same way as for first-order variable-assignments: For an assignment* ζ *of state variables, a state variable* x*, and* $n \in \mathbb{N}$*, the* modification ζ_s^n *is identical with* ζ *except that it assigns* n *to the state variable* s*:*

$$\zeta_s^n : \mathsf{SVar} \to \mathbb{N} : \begin{cases} t \mapsto \zeta(t) & \text{if } t \neq s \quad , \\ s \mapsto n & \text{otherwise} \ . \end{cases}$$

Definition 4 (Semantics of Event Formulas). *Let* $\boldsymbol{K}$ *be a linear Kripke structure with a universe* $\boldsymbol{U}$*,* χ *a variable assignment,* ζ *an assignment of state variables. Then,* $\models$ *is extended to ECL formulas as follows:*

1. *For* φ *a first-order formula:* $(\boldsymbol{K}[i,j], \chi) \models \varphi \ :\Leftrightarrow\ i = j$ *and* $(\boldsymbol{M}(i), \chi) \models \varphi$.
2. $(\boldsymbol{K}[i,j], \chi, \zeta) \models \partial$ *for all* $i \leq j$*, and* $(\boldsymbol{K}[i,j], \chi, \zeta) \models \bot$ *for no* i, j .
3. *For* φ *an action formula:* $(\boldsymbol{K}[i,j], \chi) \models \varphi \ :\Leftrightarrow\ j = i+1$ *and* $(\boldsymbol{K}\langle i,j \rangle, \chi) \models \varphi$.
4. $(\boldsymbol{K}[i,j], \chi, \zeta) \models s \rhd \mathrm{E} \ :\Leftrightarrow\ \zeta(s) = i$ *and* $(\boldsymbol{K}[i,j], \chi, \zeta) \models \mathrm{E}$ *, and*
   $(\boldsymbol{K}[i,j], \chi, \zeta) \models \mathrm{E} \lhd s \ :\Leftrightarrow\ \zeta(s) = j \in \mathbb{N}$ *and* $(\boldsymbol{K}[i,j], \chi, \zeta) \models \mathrm{E}$.
5. $(\boldsymbol{K}[i,j], \chi, \zeta) \models \mathrm{E} \otimes \mathrm{F} \ :\Leftrightarrow$ *there is a* $k \in \mathbb{N}$ *s.t.* $(\boldsymbol{K}[i,k], \chi, \zeta) \models \mathrm{E}$
   and $(\boldsymbol{K}[k,j], \chi, \zeta) \models \mathrm{F}$.

6. $(\boldsymbol{K}[i,j],\chi,\zeta) \models \mathrm{E} \times \mathrm{F}$ $:\Leftrightarrow$ $(\boldsymbol{K}[i,j],\chi,\zeta) \models \mathrm{E}$ *and* $(\boldsymbol{K}[i,j],\chi,\zeta) \models \mathrm{F}$.
7. $(\boldsymbol{K}[i,j],\chi,\zeta) \models \sum_{1\ldots n} \mathrm{E}_i$ $:\Leftrightarrow$ *there is a* $1 \leq k \leq n$ *s.t.* $(\boldsymbol{K}[i,j],\chi,\zeta) \models \mathrm{E}_k$.
8. $(\boldsymbol{K}[i,j],\chi,\zeta) \models \mathrm{E}^\star$ $:\Leftrightarrow$ *there is a* $k \geq 0$ *s.t.* $(\boldsymbol{K}[i,j],\chi,\zeta) \models \underbrace{\mathrm{E} \otimes \mathrm{E} \otimes \ldots \otimes \mathrm{E}}_{k\text{-times}}$.
(note that for $k = 0$*, this means* true*)*
9. $(\boldsymbol{K}[i,j],\chi,\zeta) \models \exists x : \mathrm{E}$ $:\Leftrightarrow$ *there is a* $d \in \boldsymbol{U}$ *s.t.* $(\boldsymbol{K}[i,j],\chi_x^d,\zeta) \models \mathrm{E}$,
$(\boldsymbol{K}[i,j],\chi,\zeta) \models \forall x : \mathrm{E}$ $:\Leftrightarrow$ *for all* $d \in \boldsymbol{U}$, $(\boldsymbol{K}[i,j],\chi_x^d,\zeta) \models \mathrm{E}$.
10. $(\boldsymbol{K}[i,j],\chi,\zeta) \models \blacklozenge s : \mathrm{E}$ $:\Leftrightarrow$ *there is a* $k : i \leq k \leq j$ *and* $(\boldsymbol{K}[i,j],\chi,\zeta_s^k) \models \mathrm{E}$,
$(\boldsymbol{K}[i,j],\chi,\zeta) \models \blacksquare s : \mathrm{E}$ $:\Leftrightarrow$ $j = \infty$ *and for all* $k \geq i$, $(\boldsymbol{K}[i,j],\chi,\zeta_s^k) \models \mathrm{E}$.

As short notations, $\mathrm{E} \cdot \mathrm{F} := \mathrm{E} \otimes \partial \otimes \mathrm{F}$ *and* $\mathrm{E}_1 + \cdots + \mathrm{E}_n := \sum_{i \in \{1,\ldots,n\}} \mathrm{E}_i$ *are used.*

Remark 1 (Semantics of ECL). Note, that in this definition, if F is an ECL formula and E is a subformula of F, the segment of the Kripke structure which is looked at for E is always contained in the segment which is looked at for F. This will not be the case in Definition 6 where the semantics of temporal implication is defined.

ECL-event formulas without universal temporal quantification describe *finitely detectable* events:

Proposition 1 (Finite Satisfiability). *Let* E *be an ECL-event formula which does not contain* $\blacksquare s$*. Then, if a (possibly infinite computation) satisfies* E *(i.e., an event described by* E *occurs in the computation) then already a finite prefix of this computation satisfies* E*. Formally,* $(\boldsymbol{K}[i,j],\chi,\zeta) \models \mathrm{E}$ *implies that there is a (unique least)* $i \leq k \leq j$ *such that* $k < \infty$ *and* $(\boldsymbol{K}[i,k],\chi,\zeta) \models \mathrm{E}$.

For the proof, one must consider that an infinite sequence only satisfies formulas which contain either $\blacksquare$ or a final delay (this will change with the introduction of temporal implication in Sec. 5).
As a consequence, events of infinite length, i.e., $\mathsf{len}(\mathrm{E}) = \{\infty\}$ (note that these are exactly those events which contain $\blacksquare$ in all their alternatives), cannot be postfixed: there is no state where an action "raising" such an event can be terminated.

Example 1 (Event Formulas).

- Iteration: a state will be reached where F holds, an then, E is iterated until G is satisfied: $\partial \otimes F \otimes \mathrm{E}^\star \otimes G$.
- Conditional execution: If the set of elements for which some ECL formula $\mathrm{E}(x)$ should be satisfied should be restricted by some first-order formula $\varphi(x)$, this can be formulated as $\blacklozenge s : \forall x : ((\varphi(x) \otimes \mathrm{E}(x)) + (\neg\varphi(x) \otimes \partial)) \lhd s$.
- Consider a workflow, consisting of several jobs, each of them can be performed by several ways, consisting of a first and a second part. The additional condition is that all first parts are finished before some second part is started: This is done by a synchronization variable local to the process:

$$\mathrm{Workflow} = \blacklozenge s : (\mathrm{Job}_1 \times \mathrm{Job}_2 \times \ldots \times \mathrm{Job}_n) \ ,$$
$$\mathrm{Job}_i = \mathrm{Way}_{i,1} + \cdots + \mathrm{Way}_{i,k_i} \ ,$$
$$\mathrm{Way}_{i,j} = \partial \otimes \mathrm{First}_{i,j} \otimes \partial \lhd s \otimes \partial \otimes \mathrm{Second}_{i,j} \otimes \partial \ .$$

3.1 Constraints

Constraints can be used for describing processes from another point of view: in a declarative way, computations not satisfying a given set of constraints can be ruled out. Temporal constraints are specified by constraint formulas in an if-then-style. With the above synchronization formalism, the temporal implication constructs "*if ... then before/later/sometimes ...*" can be formulated by a single implication and state variables.

Definition 5 (Constraint Formulas). *For expressing temporal constraints, an additional connective is added to the syntax of ECL formulas:*

11. For ECL formulas E,F, $\mathrm{E} \leadsto \mathrm{F}$ *is an ECL (constraint) formula with* $\mathsf{len}(\mathrm{E} \leadsto \mathrm{F}) = 0$ *(temporal implication).*

Definition 6 (Semantics of Constraints). *Let* $\boldsymbol{K}$ *be a linear Kripke structure,* χ *a variable assignment,* ζ *an assignment of state variables. Then,* $\models$ *is extended to* $\leadsto$ *as follows:*

11. $(\boldsymbol{K}[i,j],\chi,\zeta) \models (\mathrm{E} \leadsto \mathrm{F}) \;:\Leftrightarrow\; i = j$ *and if* $(\boldsymbol{K}[i,k_1],\chi,\zeta) \models \mathrm{E}$ *for some* k_1, *then* $(\boldsymbol{K}[0,k_2],\chi,\zeta) \models \mathrm{F}$ *for some* k_2 .

Note that $\mathrm{E} \leadsto \bot$ is ECL's negation, i.e., requires that an event E is *not* detected. In contrast to ECL-event formulas (cf. Remark 1), for a general ECL-constraint formula, the whole path has to be considered for evaluating the consequence. For negation via temporal implication, the focused segment has *not* to be extended. A constraint is evaluated wrt. one state and has length 0.

Proposition 2 (Connectives of [Jab94]). *The constraint connectives used in [Jab94] can be defined as derived symbols:*

- *"Deadline": eventually* D *will be satisfied, and before,* C *has to be satisfied:*
$$D < C \;:\Leftrightarrow\; (\partial \otimes D \otimes \partial) \times \blacksquare s : ((\partial \otimes s \rhd D) \leadsto (\partial \otimes C \otimes \partial \lhd s)) \;.$$
- *"Delay":* C *can only occur if* D *occured before:*
$$C > D \;:\Leftrightarrow\; \blacksquare s : (\partial \otimes s \rhd C) \leadsto ((\partial \otimes D \otimes \partial \lhd s)) \;.$$

Example 2 (Constraints). (assume E,F, and G to be action formulas)

- After F, E will never be satisfied: $(\partial \otimes \mathrm{F} \cdot \mathrm{E}) \leadsto \bot$,
- if E and later F are satisfied, then between them, G must eventually be satisfied: $\blacksquare s_1, s_2 : (\partial \otimes \mathrm{E} \lhd s_1 \cdot s_2 \rhd \mathrm{F}) \leadsto (\partial \otimes s_1 \rhd \partial \otimes \mathrm{G} \otimes \partial \lhd s_2)$.
- between E and the next F, G is not satisfied:
$$\blacksquare s_1, s_2 : \; ((\partial \otimes \mathrm{E} \lhd s_1 \cdot s_2 \rhd \mathrm{F}) \times ((\partial \otimes s_1 \rhd \partial \otimes \mathrm{F} \otimes \partial \lhd s_2) \leadsto \bot)) \leadsto ((\partial \otimes s_1 \rhd \partial \otimes \mathrm{G} \otimes \partial \lhd s_2) \leadsto \bot) \;.$$
- if E is satisfied, in the state before, F holds: $\blacksquare s : (\partial \otimes s \rhd \mathrm{E}) \leadsto (\partial \otimes F \lhd s)$.

Example 3 (2-FOL). The 2-FOL formula given in the introduction which is not expressible in LTL, CTL etc. can be expressed in ECL by

$$\blacksquare s_1, s_2 (\forall x((\partial \otimes (s_1 \rhd p(x)) \otimes \partial \otimes (s_2 \rhd p(x)) \otimes \partial) + (\partial \otimes (s_1 \rhd (\neg p(x))) \otimes \partial \otimes (s_2 \rhd (\neg p(x))) \otimes \partial))) \leadsto \partial \otimes (s_2 \rhd q) \;.$$

4 Tableau Semantics for Linear Kripke Structures

The tableau semantics and -calculus is a linear-time adaptation of the one presented for first-order CTL in [MS96] which uses branching time first-order Kripke structures as underlying semantics. For the first-order part, the well-known first-order tableau calculus is embedded into the tableau calculus which is constructed. It is necessary to describe many individual states as well as the relations between them in the tableau, including the ordering of states. Thus, three kinds of entities have to be described: Elements of the universe inside states, states, and the path with its transitions. In the chosen semantics, elements of the universe and states will be explicitly named when their existence is stated by a formula:

- Elements of the universe: a new constant resp. function symbol is introduced by the usual δ-rule when an existential quantifier is processed.
- States: states are named when their existence is required by a complex event.

In general, between two known states there can be many other still unknown states. These can be named when needed. Thus, a straightforward resolving of eventualities at any time is possible. A similar approach for PLTL where only the relevant states are generated has been used in [SGL97].

To allow the naming of states at any position of the model, the description of the path contains, apart from the (partial) ordering of known states, additional information about formulas which have to be true in still unknown states on the segments in-between. These are used when new states are explicitly named.

Representation. As a conceptional extension of first-order tableaux, every branch of the tableau corresponds to a linear Kripke structure with a transition oracle. Apart from the first-order portion, information about the frame and the transitions has to be coded in tableau nodes. For distinguishing and naming of states, a tableau calculus based on the free variable tableau calculus given in [Fit90] augmented with *prefixes* is employed: A formula F, assumed to be true in a certain state, occurs in the tableau as *state prefixed formula* $\gamma{:}F$. Additionally, path information formulas contain the information about the prefixes situated on the path.

Thus, the signature $\Sigma_{\mathcal{T}}$ used in the tableau is partitioned into Σ_L (first-order part), Σ_A (action symbols), and Σ_F (prefix symbols).

Σ_L is obtained by augmenting Σ with a countable infinite set of n-ary skolem function symbols for every $n \in \mathbb{N}$ and a countable infinite set of variables X_i.

Σ_F is a set of *prefix symbols* containing an infinite set of n-ary prefix symbols for every $n \in \mathbb{N}$. The construction of prefixes corresponds to the use of skolem functions in the first-order tableau calculus. Here the prefix symbols take the role of the skolem function symbols. Analogously to the skolem terms containing free variables resulting from invocations of the γ-rule, prefixes γ are terms consisting of a prefix symbol $\hat{\gamma}$ of an arity n and an n-tuple of terms as arguments. Additionally, there is a 0-ary symbol $\hat{\infty}$ which is no prefix symbol but is used in a similar way.

Definition 7. *Let Σ_F be the set of prefix symbols and Σ^c the subset of Σ which is interpreted state-independently. Then the following sets Σ_L and Γ are simultaneously recursively defined:*

$\Sigma_L := \Sigma \cup \{f \mid f \text{ an } n\text{-ary skolem function symbol}\} \cup \{f_\gamma \mid f \in \Sigma \backslash \Sigma^c, \gamma \in \Gamma\}$,

with $\mathsf{ord}(f_\gamma) = \mathsf{ord}(f)$ *and skolem functions and all* f_γ *interpreted state-independently, thus*

$\Sigma_L^c := \Sigma^c \cup \{f \mid f \text{ an } n\text{-ary skolem function symbol}\} \cup \{f_\gamma \mid f \in \Sigma \backslash \Sigma^c, \gamma \in \Gamma\}$.

The set of prefixes *is given as*
$\Gamma := \{\hat{\gamma}(t_1, \ldots, t_n) \mid \hat{\gamma} \in \Sigma_F \text{ is an } n\text{-ary prefix symbol, } t_1, \ldots, t_n \in \mathsf{Term}_{\Sigma_L^c}\}$.

For $\Gamma \subset \mathsf{Term}_{\Sigma_\mathcal{T}}$, it is precisely the leading function symbol which is a prefix symbol taken from Σ_F, and all argument terms are in $\mathsf{Term}_{\Sigma_L^c}$. For Σ_L^c, $\boldsymbol{K}$ induces a state-independent interpretation $(K, \boldsymbol{U})$.

An interpretation of $\Sigma_\mathcal{T}$ – describing a Kripke structure $\boldsymbol{K} = (\boldsymbol{U}, \boldsymbol{M})$ – is accordingly partitioned: The interpretation of Σ_L is taken over by the set $\{\boldsymbol{M}(i) \mid i \in \mathbb{N}\}$ of first-order interpretations, and the symbols of Σ_A are interpreted by the transition oracle. Complementary to this, an "interpretation" of the prefix symbols in Σ_F is defined. The corresponding evaluation maps prefixes to natural numbers:

Definition 8. *A prefix interpretation of a set Σ_F of prefix symbols to a linear Kripke structure $\boldsymbol{K} = (\boldsymbol{U}, \boldsymbol{M})$ is a mapping $\pi : (\Sigma_F \cup \{\hat{\infty}\}) \to \boldsymbol{U}^m \to \mathbb{N} \cup \{\infty\}$ which maps every m-ary prefix symbol $\hat{\gamma} \in \Sigma_F$ to a function $\pi(\hat{\gamma}) : \boldsymbol{U}^m \to \mathbb{N} \cup \{\infty\}$ with $\pi(\gamma) = \infty \Leftrightarrow \gamma = \hat{\infty}$. Defining $\Pi_{\boldsymbol{K}} := (\pi \cup K, \mathbb{N} \cup \{\infty\})$, Π is organized similarly to a first-order interpretation $\boldsymbol{I} = (I, \boldsymbol{U})$ with a mapping π and the "universe" $\mathbb{N} \cup \{\infty\}$, inducing an* evaluation *$\Pi_{\boldsymbol{K}} : (\Gamma \cup \{\hat{\infty}\}) \times \mathsf{Term}_{\Sigma_L^c} \to \mathbb{N} \cup \{\infty\}$ of prefixes as follows:*
For $\gamma = \hat{\gamma}(t_1, \ldots, t_n) \in \Gamma$ (thus $t_i \in \mathsf{Term}_{\Sigma_L^c}$ evaluated state-independently by $\boldsymbol{K}$),

$$\Pi_{\boldsymbol{K}}(\gamma, \chi) := (\pi(\hat{\gamma}))(\boldsymbol{K}(t_1, \chi), \ldots, \boldsymbol{K}(t_n, \chi)) \ .$$

Finally, the interpretation of the derived function symbols f_γ is defined state-independently for all $i \in \mathbb{N}$ as

$$(\boldsymbol{M}(n))(f_\gamma(t_1, \ldots, t_n), \chi) := (\boldsymbol{M}(\Pi_{\boldsymbol{K}}(\gamma, \chi)))(f(t_1, \ldots, t_n), \chi) \quad .$$

Tableau formulas. Logical formulas occur in the tableau as prefixed formulas, additionally, the Kripke frame is encoded in *path information formulas* (*pif*s): An additional symbol $\triangledown$ can occur instead of prefixes and state variables as an auxiliary "generic prefix". $\triangledown$ is instantiated by a prefix when a state on the respective segment is explicitly named.

Definition 9 (Syntax of $\mathcal{TE}$ Tableau Formulas).

1. *For an ECL formula P with state variables $s_1, \ldots, s_n$ and prefixes $\gamma_1, \ldots, \gamma_n \in \Gamma \cup \{\triangledown\}$, $P[\gamma_1/s_1, \ldots, \gamma_n/s_n]$ is a $\mathcal{TE}$-path formula.*
2. *With $\blacklozenge s : P$ and $\blacksquare s : P$ a $\mathcal{TE}$-path formula and γ a prefix, $\blacklozenge s_{\geq\gamma} : P$ and $\blacksquare s_{\geq\gamma} : P$ are $\mathcal{TE}$-path formulas.*

3. *For a $\mathcal{TE}$-path formula P and $\gamma \in \Gamma \cup \{\nabla\}$, $\gamma{:}P$ is a $\mathcal{TE}$-prefixed formula.*
4. *Every $\mathcal{TE}$-prefixed formula which does not contain ∇ is also a $\mathcal{TE}$-node formula.*
5. *With $\gamma_0, \gamma_1, \ldots$ prefixes and L_i a conjunction of $\mathcal{TE}$-prefixed formulas, $[\gamma_0, L_0, \gamma_1, L_1, \ldots, \gamma_n, L_n, \hat{\infty}]$ is a $\mathcal{TE}$-path information formula.*
6. *Every $\mathcal{TE}$-path information formula is a $\mathcal{TE}$-node formula.*

Remark 2 (Properties of $\mathcal{TE}$ formulas). Note that, $\mathcal{TE}$-path formulas do *not* contain any free state variables, and the restricted quantifiers $\blacklozenge s_{\geq\gamma}$ and $\blacksquare s_{\geq\gamma}$ occur only in the outermost positions of $\mathcal{TE}$-path formulas.

Definition 10 (Semantics of $\mathcal{TE}$ Tableau Formulas). *The relation $\models\!\!\models$ of a linear Kripke structure with a transition oracle $\boldsymbol{K} = (\boldsymbol{U}, \boldsymbol{M})$, a prefix interpretation Π, a variable assignment χ, and a $\mathcal{TE}$-node formula is defined as follows:*

– *For every $\mathcal{TE}$-path formula E neither containing ∇ nor beginning with $\blacklozenge s_{\geq\gamma}$ or $\blacksquare s_{\geq\gamma}$ and every prefix γ: let $\gamma_1, \ldots, \gamma_n$ be the prefixes occurring in E, $s_1, \ldots, s_n$ new state variables and $\zeta = \{s_j \mapsto \Pi(\gamma_j, \chi) \mid 1 \leq j \leq n\}$. Then,*

$$(\boldsymbol{K}, \Pi, \chi) \models\!\!\models \gamma{:}\mathrm{E} \;:\Leftrightarrow\; \text{there is an } i \in \mathbb{N} \cup \{\infty\} \text{ s.t. } (\boldsymbol{K}[\Pi(\gamma, \chi), i], \chi, \zeta) \models \mathrm{E}[s_1/\gamma_1, \ldots, s_n/\gamma_n] \;.$$

(In particular, for an action formula E,

$$(\boldsymbol{K}, \Pi, \chi) \models\!\!\models \gamma : \mathrm{E} \;:\Leftrightarrow\; (\boldsymbol{K}\langle \Pi(\gamma, \chi), \Pi(\gamma, \chi){+}1\rangle, \chi) \models \mathrm{E} \;,$$

i.e., if the transition from $\Pi(\gamma, \chi)$ to $\Pi(\gamma, \chi){+}1$ satisfies E under χ.)

– *For every $\mathcal{TE}$-path formula beginning with $\blacklozenge s_{\geq\gamma}$ or $\blacksquare s_{\geq\gamma}$ and every prefix α,*

$$(\boldsymbol{K}, \Pi, \chi) \models\!\!\models \alpha : \blacklozenge s_{\geq\gamma}{:}P \;:\Leftrightarrow\; (\boldsymbol{K}, \Pi \cup \{\hat{\delta} \mapsto i\}, \chi) \models\!\!\models \alpha{:}P[\hat{\delta}/s] \text{ for some } i \geq \Pi(\gamma, \chi),$$
$$(\boldsymbol{K}, \Pi, \chi) \models\!\!\models \alpha : \blacksquare s_{\geq\gamma}{:}P \;:\Leftrightarrow\; (\boldsymbol{K}, \Pi \cup \{\hat{\delta} \mapsto i\}, \chi) \models\!\!\models \alpha{:}P[\hat{\delta}/s] \text{ for all } i \geq \Pi(\gamma, \chi) \;.$$

– *For every path information formula $I = [\gamma_0, L_0, \gamma_1, L_1, \ldots, \gamma_n, L_n, \hat{\infty}]$:*

$$(\boldsymbol{K}, \Pi, \chi) \models\!\!\models [\gamma_0, L_0, \gamma_1, L_1, \ldots, \gamma_n, L_n, \hat{\infty}]$$

iff $\Pi(\gamma_0, \chi) = 0$ and for all $0 \leq i \leq n$:

$$\Pi(\gamma_i, \chi) < \Pi(\gamma_{i+1}, \chi) \;, \text{ and for all } j \text{ with } \Pi(\gamma_i, \chi) < j < \Pi(\gamma_{i+1}, \chi) :$$
$$(\boldsymbol{K}, \Pi \cup \{\hat{\delta} \mapsto j\}, \chi) \models\!\!\models L_i[\hat{\delta}/\nabla] \text{ with } \hat{\delta} \text{ a new 0-ary prefix symbol.}$$

This condition means that for all (finitely, but arbitrary many) states j situated between $\Pi(\gamma_i, \chi)$ and $\Pi(\gamma_{i+1}, \chi)$ in $\boldsymbol{K}$, the instantiation of L_i for a new prefix which is mapped to j holds in $\boldsymbol{K}$.
Note that $L_i = \mathsf{false}$ implies $\Pi(\gamma_{i+1}, \chi) = \Pi(\gamma_i, \chi){+}1$.

Note that the semantics for $\mathcal{TE}$-prefixed formulas containing ∇ is not defined. They occur only in the list components of path information formulas, and ∇ is instantiated by a prefix when the list is used.

A set $\mathcal{F}$ of path information formulas and prefixed formulas is *valid* in a linear Kripke structure $\boldsymbol{K} = (\boldsymbol{U}, \boldsymbol{M})$ under a variable assignment χ if there is a prefix-interpretation Π such that $(\boldsymbol{K}, \Pi, \chi) \models\!\!\!\models \mathcal{F}$. Since a branch of a tableau is a set of formulas like this, validity is a relation on Kripke structures with transition oracles and branches.

The construction of Kripke structures and consistent prefix-interpretations to a given set of formulas plays an important role in the proof of correctness.

5 The Tableau Calculus $\mathcal{TE}$

As usual, the tableau is initialized with a set of formulas which should be proven to be inconsistent, i.e., it is shown that there is no linear Kripke structure $\boldsymbol{K} = (\boldsymbol{U}, \boldsymbol{M})$ such that F does holds in $\boldsymbol{M}(0)$. Thus the initialization of the tableau is

$$\frac{}{\begin{array}{c}[\hat{0}, \mathsf{true}, \infty] \\ \hat{0} : F\end{array}}$$

The α-, β-, γ-, δ-rules for first-order formulas are as usual (extended with prefixes), boolean combinations of prefixed formulas are resolved analogously.

Atomic Closure Rule. For a substitution σ and a prefix γ, σ_γ is the γ-*localization*, i.e. $\sigma_\gamma(X)$ is obtained from $\sigma(X)$ by replacing every function symbol $f \in \Sigma \backslash \Sigma^c$ by its localized symbol f_γ. So the substitutes in σ_γ contain only function symbols which are interpreted state-independently. In the rule shown at the right, σ is a substitution and A, B are atomic formulas.

$$\frac{\begin{array}{c}\gamma : A \\ \gamma : B \\ \sigma(A) = \neg\sigma(B)\end{array}}{\begin{array}{c}\mathsf{close} \\ \text{apply } \sigma_\gamma \text{ to the whole tableau.}\end{array}}$$

For resolving modalities, the information about the frame of the Kripke structure, which is encoded in the *pif* s, is used. In a single step, a prefixed formula is resolved "along" a *pif*, inducing the following form of tableau rules:

$$\frac{\begin{array}{c}\text{prefixed formula} \\ \text{path information formula}\end{array}}{\begin{array}{c}\text{prefixed formulas} \\ \text{path information formulas}\end{array}}$$

where the premise takes the latest *pif* on the current branch. The connection between the prefixed formula being resolved and the *pif* is established by the prefix.

In the sequel, T denotes the current branch of the tableau, $\mathsf{free}(T)$ the free variables on T, $\hat{\gamma}$ is a new prefix symbol, and E is an ECL-formula.

Prefixed Event Formulas. For a prefixed event formula, the prefix must coincide with the respective prefix of the prefixed formula in the tableau (i.e., $\alpha : \beta \rhd \mathrm{E}$) requires $\alpha \equiv \beta$.[1] (Here and in the following, $\beta \equiv \alpha$ evaluates to close if α and β are different prefixes, and to true otherwise.)

$$\frac{\alpha : (\beta \rhd \mathrm{E})}{\begin{array}{c}\beta \equiv \alpha \\ \alpha : \mathrm{E}\end{array}}$$

Postfixed Event Formulas. For a postfixed formula $\mathrm{E} \lhd \beta$, it has to be distinguished whether E is elementary, i.e., a delay, a deadlock, an action formula,

[1] here, the reader is asked to distinguish between a prefix $\gamma : \mathrm{E}$ and prefixing or postfixing a formula by $\gamma \rhd \mathrm{E}$ or $\mathrm{E} \lhd \gamma$.

or a first-order formula, or whether it is itself a complex event. The postfixing-operator $\lhd$ associates a synchronization point with the last elementary component of an event. Thus, postfixes can only be resolved if this final event is associated with a transition, otherwise, the formula has to be rewritten. The base cases also act as closure rules if the assignment of prefixes is inconsistent:

$$\frac{\begin{array}{c}\alpha : \mathrm{E} \lhd \gamma\ ,\ \mathsf{len}(\mathrm{E}) = \{1\}\ , \\ [..., \alpha, L, \beta, ...]\end{array}}{\begin{array}{c}[..., \alpha, \mathsf{false}, \beta, ...] \\ \beta \equiv \gamma \\ \alpha : \mathrm{E}\end{array}} \qquad \frac{\alpha : F \lhd \beta\ ,\ \mathsf{len}(F) = \{0\}}{\begin{array}{c}\beta \equiv \alpha \\ \alpha : F\end{array}} \qquad \frac{\begin{array}{c}\alpha : \partial \lhd \beta \\ [..., \beta, ..., \alpha, ...]\end{array}}{\mathsf{close}}$$

Definition 11 (Normal Form wrt. Postfixes). *The* syntactic *operator Θ rewrites postfixed complex ECL formulas such that* $\mathsf{len}(\mathrm{E}) \cap \{2, 3, ...\} \neq \emptyset$ *by moving the postfix inside the outermost event connective:*

$$\Theta((\mathrm{E} \lhd t) \lhd s) := \begin{cases} (\mathrm{E} \lhd t) \lhd s & \text{if } \mathsf{len}(\mathrm{E}) \in \{\{0\}, \{1\}\}\ , \\ \Theta(\Theta(\mathrm{E} \lhd t) \lhd s) & \text{otherwise}\ , \end{cases}$$

$$\begin{array}{ll}
\Theta((t \rhd \mathrm{E}) \lhd s)\ := t \rhd (\mathrm{E} \lhd s)\ , & \Theta((\mathrm{E}^{\star}) \lhd s) := \mathsf{true} \lhd s + \mathrm{E}^{\star} \otimes (\mathrm{E} \lhd s)\ , \\
\Theta((\mathrm{E} \otimes \mathrm{F}) \lhd s) := \mathrm{E} \otimes (\mathrm{F} \lhd s)\ , & \Theta((\exists x : \mathrm{E}) \lhd s)\ := \exists x : (\mathrm{E} \lhd s)\ , \\
\Theta((\mathrm{E} \times \mathrm{F}) \lhd s) := (\mathrm{E} \lhd s) \times (\mathrm{F} \lhd s)\ , & \Theta((\forall x : \mathrm{E}) \lhd s)\ := \forall x : (\mathrm{E} \lhd s)\ , \\
\Theta((\mathrm{E} + \mathrm{F}) \lhd s) := (\mathrm{E} \lhd s) + (\mathrm{F} \lhd s)\ , & \Theta((\blacklozenge s' : \mathrm{E}) \lhd s) := \blacklozenge s' : (\mathrm{E} \lhd s)\ .
\end{array}$$

Note that formulas of length $\mathsf{len}(\mathrm{E}) = \{0\}$ are (possibly pre- or postfixed) first-order formulas or temporal implications, and formulas of $\mathsf{len}(\mathrm{E}) = \{1\}$ are (possibly pre- or postfixed) action formulas. Thus, Θ is the identity on temporal implications $\mathrm{E} \rightsquigarrow \mathrm{F}$ (since $\mathsf{len}(\mathrm{E} \rightsquigarrow \mathrm{F}) = \{0\}$). Postfixing of formulas of the form $\blacksquare s : \mathrm{E}$ is not defined. ∂ is not a complex event formula.
The above definition is *not* recursive, but denotes only a one-level rewriting of the parse-tree of the formula (which is typical for tableau calculi) – Θ does not occur on the right hand side. Only when Θ is iterated, a *normal form* is obtained:

Proposition 3 (Normal Form wrt. Prefixes and Postfixes). *For every linear Kripke structure $\boldsymbol{K}$, ECL formula* E*, variable assignment χ and assignment ζ of state variables,* $(\boldsymbol{K}[i, j], \chi, \zeta) \models \mathrm{E} \Leftrightarrow (\boldsymbol{K}[i, j], \chi, \zeta) \models \Theta(\mathrm{E})$ *and* iterated *application of Θ yields an expression where only formulas* E *s.t.* $\mathsf{len}(\mathrm{E}) \in \{\{0\}, \{1\}\}$ *or delays are postfixed.*

In non-base cases, the operator Θ is applied for rewriting the formula:

$$\frac{\alpha : \mathrm{E} \lhd \beta, \quad \mathsf{len}(\mathrm{E}) \cap \{2, 3, ...\} \neq \emptyset}{\alpha : \Theta(\mathrm{E} \lhd \beta)}$$

Alternative and parallel compositions.
These connectives are disjunctive and conjunctive, respectively, and are resolved analogously to the α- and β-rules of first-order tableau calculi:

$$\frac{\alpha : \sum_{i=1...n} \mathrm{E}_i}{\alpha : \mathrm{E}_1 \ \Big|\ \ ... \ \ \Big|\ \alpha : \mathrm{E}_n} \qquad \frac{\alpha : (\mathrm{E}_1 \times ... \times \mathrm{E}_n)}{\begin{array}{c}\alpha : \mathrm{E}_1 \\ \vdots \\ \alpha : \mathrm{E}_n\end{array}}$$

Sequential Composition: $\text{E} \otimes \text{F}$. Semantically, the brackets around composite events of the form $(\sum \ldots \text{E}_i)$ and $(\text{E}_1 \times \ldots \times \text{E}_n)$ can be regarded as synchronization points, i.e., (virtual) entities in the sense of the tableau representation: Detecting an event $\text{E} \otimes \text{F}$ in α is equivalent to the event $\blacklozenge s : (\text{E} \lhd s) \otimes (s \rhd \text{F})$ where s must be bound to a state after the current state.
With the rules for $\blacklozenge_{\geq \gamma}$ (see below), the state variable in $\alpha : \blacklozenge s_{\geq \alpha} : (\text{E} \lhd s) \otimes (s \rhd \text{F})$ is instantiated systematically with the prefixes. Then, the resulting formula $\alpha : (\text{E} \lhd \beta) \otimes (\beta \rhd \text{F})$ can be split into two parts ("sequential conjunction").

$$\frac{\alpha : \text{E} \otimes \text{F}}{\alpha : \blacklozenge s_{\geq \alpha} : (\text{E} \lhd s) \otimes (s \rhd \text{F})} \qquad \frac{\alpha : (\text{E} \lhd \beta) \otimes (\beta \rhd \text{F})}{\begin{array}{c}\alpha : \text{E} \lhd \beta \\ \beta : \text{F}\end{array}}$$

Shortcuts can be defined when one of the events is a base case, i.e., a delay, a first-order formula, or an action formula.

Iterative Composition. Iterative composition is reduced to sequential composition:

$$\frac{\alpha : \text{E}^\star}{\alpha : \mathsf{true} \;\Big|\; \alpha : \text{E} \otimes \text{E}^\star}$$

Temporal Quantification.
Due to the fact that there is a linear ordering between states, the γ- and δ-rules cannot be adapted to temporal quantification of synchronization points; instead, every synchronization point must be integrated into this ordering. For this, analogous to the resolving of eventuality formulas in the tableau calculus for CTL, synchronization points are shifted along path formulas. The rules correspond to a systematic application of the γ- or δ-rules, known from first-order tableau calculi for universal and existential quantification, along the state sequence.

$$\frac{\alpha : \blacksquare s : \text{E}}{\alpha : \blacksquare s_{\geq \alpha} : \text{E}} \qquad \frac{\alpha : \blacklozenge s : \text{E}}{\alpha : \blacklozenge s_{\geq \alpha} : \text{E}}$$

$$\frac{\begin{array}{c}\alpha : \blacksquare s_{\geq \beta} : \text{E} \\ [\ldots, \beta, L, \gamma, \ldots]\end{array}}{\begin{array}{c}[\ldots, \beta, L \wedge \alpha : \text{E}[\nabla/s], \gamma, \ldots] \\ \alpha : \text{E}[\beta/s] \\ \text{if } \gamma \neq \infty\text{:} \quad \alpha : \blacksquare s_{\geq \gamma} : \text{E}\end{array}}$$

Expl.: $\text{E}[\nabla/s]$ holds for *all* subsequent states – i.e., for α, for all unknown states between α and γ, and for all states $\geq \gamma$.

$$\frac{\begin{array}{c}\alpha : \blacklozenge s_{\geq \beta} : \text{E} \\ [\ldots, \beta, L, \gamma, \ldots]\end{array}}{\alpha : \text{E}[\beta/s] \;\Bigg|\; \begin{array}{c}[\ldots, \beta, L \wedge \alpha : \text{E}[\nabla/s] \leadsto \bot, \\ \hat{\delta}(\mathsf{free}(T)), L, \gamma, \ldots] \\ L[\hat{\delta}(\mathsf{free}(T))/\nabla] \\ \alpha : (\text{E}[\beta/s] \leadsto \bot) \\ \alpha : \text{E}[\hat{\delta}(\mathsf{free}(T))/s]\end{array} \;\Bigg|\; \begin{array}{c}\text{if } \gamma \neq \infty\text{:} \\ [\ldots, \beta, L \wedge \alpha : \text{E}[\nabla/s] \leadsto \bot, \gamma, \ldots] \\ \alpha : \text{E}[\beta/s] \leadsto \bot \\ \alpha : \blacklozenge s_{\geq \gamma} : \text{E}\end{array}}$$

Expl.: $\text{E}[\nabla/s]$ holds for *some* subsequent state – thus, this must be α, or some state between α and γ which becomes named in this case, or some state $\geq \gamma$.

Here, the second part of the consequence of the rule for $\blacklozenge s_{\geq \beta}$ names a state $\hat{\delta}(\mathsf{free}(T))$ which is considered to be relevant (since it is a possible instantiation of s). In this case, the list L in the path information formula states some requirements on all states on a path segment – thus, also on $\hat{\delta}(\mathsf{free}(T))$. These are made explicit by instantiating L with $\hat{\delta}(\mathsf{free}(T))/\nabla$.

Temporal Implication. When resolving constraints $\textsc{e} \rightsquigarrow \textsc{f}$, in the base case, $\textsc{e}$ is a first-order formula or an action formula or a delay:

for $\textsc{e}$ a first-order formula or an action formula:
$$\frac{\alpha : \textsc{e} \rightsquigarrow \textsc{f}}{\begin{array}{c|c} \alpha : \neg\textsc{e} & \alpha : \textsc{e} \\ & \hat{0} : \textsc{f} \end{array}} \qquad \frac{\alpha : \partial \rightsquigarrow \textsc{f}}{\hat{0} : \textsc{f}}$$

By applying the following rules, constraints are rewritten until the antecedent consists only of a first-order formula or an action formula:

$$\frac{\alpha : (\beta \rhd \textsc{e}) \rightsquigarrow \textsc{f}}{\begin{array}{c|c} \alpha \equiv \beta & \alpha \not\equiv \beta \\ \alpha : \textsc{e} \rightsquigarrow \textsc{f} & \end{array}} \qquad \frac{\alpha : (\textsc{e}{+}\textsc{f}) \rightsquigarrow \textsc{g}}{\begin{array}{c} \alpha : \textsc{e} \rightsquigarrow \textsc{g} \\ \alpha : \textsc{f} \rightsquigarrow \textsc{g} \end{array}} \qquad \frac{\alpha : (\blacklozenge s : \textsc{e}) \rightsquigarrow \textsc{f}}{\alpha : \blacksquare s_{\geq\alpha} : (\textsc{e} \rightsquigarrow \textsc{f})} \qquad \frac{\alpha : (\textsc{e} \rightsquigarrow \textsc{f}) \rightsquigarrow \textsc{g}}{\begin{array}{c|c} \alpha : \textsc{e} & \alpha : \textsc{e} \rightsquigarrow \textsc{f} \\ \hat{0} : (\textsc{f} \rightsquigarrow \bot) & \hat{0} : \textsc{g} \end{array}}$$

$$\frac{\alpha : (\textsc{e} \otimes \textsc{f}) \rightsquigarrow \textsc{g}}{\alpha : \blacksquare s_{\geq\alpha} : ((\textsc{e} \lhd s) \rightsquigarrow (\partial \otimes s \rhd \textsc{f} \rightsquigarrow \textsc{g}))}$$

$$\frac{\alpha : ((\textsc{e} \times \textsc{f}) \rightsquigarrow \textsc{g})}{\alpha : \blacksquare s_{\geq\alpha} : ((\alpha \rhd \textsc{e} \lhd s) \rightsquigarrow (\partial \otimes \alpha \rhd \textsc{f} \lhd s \rightsquigarrow \textsc{g}))}$$

$\triangle$ Expl.: if $\textsc{e}$ implies $\textsc{f}$ then $\textsc{g}$ holds, if either $\textsc{e} \rightsquigarrow \textsc{f}$ does not hold – i.e, $\textsc{e}$ is detected and $\textsc{f}$ cannot be detected, or $\textsc{e} \rightsquigarrow \textsc{f}$ hold and $\textsc{g}$ can be detected.

Postfixed event formulas require additional attention: base cases can be resolved immediately, non-base cases have to be rewritten:

$$\frac{\alpha : (\textsc{e} \lhd \beta) \rightsquigarrow \textsc{f}\ ,\ \mathsf{len}(\textsc{e}) = \{0\}}{\begin{array}{c|c} \beta \not\equiv \alpha & \beta \equiv \alpha \\ & \alpha : \textsc{e} \rightsquigarrow \textsc{f} \end{array}} \qquad \frac{\alpha : (\partial \lhd \alpha) \rightsquigarrow \textsc{f}}{\hat{0} : \textsc{f}}$$

$$\frac{\begin{array}{c} \alpha : (\textsc{e} \lhd \beta) \rightsquigarrow \textsc{f}\ ,\ \mathsf{len}(\textsc{e}) = \{1\} \\ [..., \alpha, L, \gamma, ...] \end{array}}{\begin{array}{c|c|c} \beta \not\equiv \gamma & [..., \alpha, \mathsf{false}, \hat{\delta}(\mathsf{free}(T)), L, \gamma, ...] & [..., \alpha, \mathsf{false}, \gamma, ...] \\ & L[\hat{\delta}(\mathsf{free}(T))/\nabla] & \beta \equiv \gamma \\ & \beta \equiv \gamma & \alpha : \textsc{e} \rightsquigarrow \textsc{f} \end{array}}$$

Expl.: First case: $\gamma \not\equiv$ is the next known state – thus, β is later than $\alpha{+}1$, then $\textsc{e} \lhd \beta$ cannot be detected in α.
Second case: $\beta = \gamma$ is the next known state, but not the successor state.

$$\frac{\begin{array}{c} \alpha : (\partial \lhd \beta) \rightsquigarrow \textsc{f} \\ [..., \alpha, ..., \beta, ...] \end{array}}{\hat{0} : \textsc{f}} \qquad \frac{\begin{array}{c} \alpha : (\partial \lhd \beta) \rightsquigarrow \textsc{f} \\ [..., \beta, ..., \alpha, ...] \end{array}}{\mathsf{close}} \qquad \frac{\begin{array}{c} \alpha : (\textsc{e} \lhd \beta) \rightsquigarrow \textsc{f} \\ \mathsf{len}(\textsc{e}) \cap \{2, 3, ...\} \neq \emptyset \end{array}}{\alpha : \Theta(\textsc{e} \lhd \beta) \rightsquigarrow \textsc{f}}$$

Some constraints show a special behavior: Since $\textsc{e}^\star$ has trivial occurrences, $\textsc{e}^\star \rightsquigarrow \textsc{f}$ reduces to $0 : \textsc{f}$:

$$\frac{\alpha : \textsc{e}^\star \rightsquigarrow \textsc{f}}{\hat{0} : \textsc{f}}$$

Events of the form $\blacksquare s : \textsc{e}$ cannot be finitely detected, thus, implications containing them in the antecedent have to be resolved in a different way. $\alpha : (\blacksquare s : \textsc{e}) \rightsquigarrow \textsc{f}$ is equivalent to the fact that either there is a state β later than α such that $\alpha : \textsc{e}[\beta/s]$ does *not* occur, or $\boldsymbol{K}$ satisfies $\textsc{f}$:

$$\frac{\alpha : (\blacksquare s : \textsc{e}) \rightsquigarrow \textsc{f}}{\begin{array}{c|c} \alpha : \blacklozenge s : (\textsc{e} \rightsquigarrow \bot) & \hat{0} : \textsc{f} \end{array}}$$

Remark 3 (Properties of $\mathcal{TE}$). $\mathcal{TE}$ has some special properties which can easily be verified when carefully looking at the rules:

- every prefix symbol is introduced exactly once, with a given arity and the current free variables of the branch as arguments. Thus, for every prefix symbol, at every time, there is exactly one prefix which is built from it.
- every prefix which occurs as prefix or postfix in some formula (i.e., as $\gamma \rhd \mathrm{E}$ or $\mathrm{E} \lhd \gamma$) of a branch occurs also in the most recent path information formula of the branch.
- if a branch of the tableau contains a formula of the form $\alpha : \blacksquare_{\geq\beta}$ or $\alpha : \blacklozenge_{\geq\beta}:P$, $\alpha \neq \beta$, the most recent path information formula on this branch is of the form $[\ldots,\alpha,\ldots,\beta,\ldots]$.
- there are no free state variables in the calculus: Whereas first-order variables are replaced by free variables or Skolem terms, state variables are always replaced by prefixes (see the rules for $\blacksquare_{\geq\gamma}$ and $\blacklozenge_{\geq\gamma}$).

Theorem 1 (Correctness of $\mathcal{TE}$).

(a) If a tableau $\mathcal{T}$ is satisfiable and $\mathcal{T}'$ is created from $\mathcal{T}$ by an application of any of the rules mentioned above, then $\mathcal{T}'$ is also satisfiable.
(b) If there is any closed tableau for $\mathcal{F}$, then $\mathcal{F}$ is unsatisfiable.

The proof (see [May98]) of (a) is done by case-splitting separately for each of the rules. By assumption, there is a Kripke structure $\boldsymbol{K}$ with a transition oracle and a prefix interpretation Π such that for every variable assignment χ there is a branch T_χ in $\mathcal{T}$ with $(\boldsymbol{K},\Pi,\chi) \models T_\chi$. In all cases apart from the atomic closure rule, $\boldsymbol{K}$ and Π are extended such that they witness the satisfiability of $\mathcal{T}'$. In case of the atomic closure rule, a Substitution Lemma guarantees the existence of a satisfying branch for every variable assignment to $\mathsf{free}(\mathcal{T}')$. (b) follows directly from (a).

Since first-order ECL is not compact, no calculus for it can be complete. The calculus is complete modulo inductive properties. For such cases, induction rules for temporal properties and well-founded data structures have to be included. In this setting, the notion of completeness has to be relativized to that any proof done in a mathematical way can be completely redone formally.

6 Conclusion

This paper presents a tableau calculus for a linear time temporal logic which is based on *temporal connectives* instead of *modal operators*. The underlying Kripke semantics is explicitely encoded into the tableau, based on the ideas of the tableau representation of branching-time structures in [MS96,May97]. In both calculi, exactly those states which are required to construct a potential model of the given formula are named explicitly, all states in-between are characterized intensionally within the path information. From the conceptual point of view, the contribution of the paper is that this tableau representation is both suitable for *modal temporal logics* and for logics using *temporal connectives* and *explicit temporal quantification*.

Focusing on temporal connectives, the logic and the calculus show how temporal connections between events can be modeled explicitly by synchronization points. The calculus integrates this synchronization smoothly into the tableau semantics via the introduction of state variables. With the $\blacksquare$ and $\blacklozenge$ operators, the handling of state variables is closely related to the strategy for resolving the CTL modalities $\Box$ and $\Diamond$, making up one more connection between the calculus presented here and the CTL calculus.

Due to its path-orientation, the language is significantly different from CTL which is based on state-formulas. Although, the tableau semantics and calculus shares its basic ideas with the CTL calculus, allowing a potential integration of both calculi.

References

BK94. A. J. Bonner and M. Kifer. An Overview of Transaction Logic. *Theoretical Computer Science*, 133(2):205–265, 1994.

BT98. H. Bowman and S. Thompson. A Tableau Method for Interval Temporal Logic with Projection. In TABLEAUX'98, LNCS 1397, pp. 108–123, Springer, 1998.

CT98. J. Chomicki and D. Toman. Temporal Logic in Information Systems. In *Logics for Databases and Information Systems*, Ch. 3, pp. 31–70. Kluwer, 1998.

Fit90. M. Fitting. *First Order Logic and Automated Theorem Proving*. Springer,1990.

Jab94. S. Jablonski. Functional and Behavioural Aspects of Process Modelling in Workflow Management Systems. *Proc. CON '94: Workflow Management*,1994.

May97. W. May. Proving Correctness of Labeled Transition Systems by Semantic Tableaux. In TABLEAUX'97, LNCS 1227, pp. 261–275. Springer, 1997.

May98. W. May. *Integrated Static and Dynamic Modeling of Processes*. PhD thesis, Institut für Informatik, Universität Freiburg, Logos Verlag, 1998.

Mos86. B. Moszkowski. *Executing Temporal Logic Programs*. Cambridge University Press, 1986.

MP95. Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Safety*. Springer, 1995.

MS96. W. May and P. H. Schmitt. A Tableau Calculus for First-Order Branching Time Logic. In *Intl. Conf. on Formal and Applied Practical Reasoning, FAPR'96*, LNCS 1085, pp. 399–413. Springer, 1996.

Pra90. V. R. Pratt. Action Logic and Pure Induction. In *Logics in AI: Europ. Workshop Jelia 90*, LNCS 478, pp. 97–120, 1990.

SGL97. P. H. Schmitt and J. Goubault-Larrecq. A Tableau System for Linear-Time Temporal Logic. In *Tools and Algorithms for the Construction and Analysis of Systems*, LNCS 1217, Springer, pp. 130–144, 1997.

Sin95. M. P. Singh. Semantical Considerations on Workflows: An Algebra for Intertask Dependencies. In *Intl. Workshop on Database Programming Languages*, Electronic Workshops in Computing, Gubbio, Italy, 1995. Springer.

TN96. D. Toman and D. Niwinski. First-Order Queries over Temporal Databases Inexpressible in Temporal Logic. In *Proc. Int. Conf. on Extending Database Technology*, LNCS 1057, pp. 307–324. Springer, 1996.

Wol85. P. Wolper. The Tableau Method for Temporal Logic. *Logique et Analyse*, 28:110–111, 1985.