Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX'97), May 13-16, 1997, Pont-à-Mousson, France
Springer LNCS 1227, pp. 261-275.

Proving Correctness of Labeled Transition Systems by Semantic Tableaux

Wolfgang May*

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

Abstract. The paper presents a method for formally proving correctness of processes specified by transition systems which is based on a tableau calculus for an extended temporal logic. The model-theoretic semantics is given by *labeled* Kripke structures, incorporating information about the actions performed in transitions. Extending first-order CTL for handling action labels, the multi-modal logic MCTL is defined which is well-suited for specifying transition systems and their properties. For MCTL, a tableau semantics and -calulus is presented, allowing formal verification.

1 Introduction

Specifications of processes are often given by transition systems, describing the effects of actions on the state of the process. Proving correctness of such specifications involves an integrated reasoning about static (data) and dynamic (actions) aspects of processes.

Labeled transition systems focus on dynamic acpects, thereby lacking facilities for representing static, data-oriented aspects. In contrast, Kripke structures focus on data-oriented aspects, providing expressive concepts for describing states while no special facilities for expressing action-oriented aspects exist.

As a means to overcome this deficiency, *labeled* Kripke structures provide a suitable model-theoretic semantics for combined modeling of both static and dynamic aspects of processes: Actions are explicitly integrated into Kripke structures by labeling the accessibility relation with the actions executed in the respective transitions.

A suitable multi-modal logic MCTL for labeled Kripke structures is derived from first-order CTL (Computation Tree Logic) by defining more detailed modalities. It is taken into consideration that, when reasoning about processes, different relationships between actions and states have to be investigated: an action is performed *and* some property will hold, or *if* an action is performed, *then* some property will hold.

Dynamic Logic [Har79] is also based on labeled Kripke structures, although it is quite different: Both its modalities query only the action carried out in the next step, corresponding to the focus of DL on *program* verification.

A logic more similar to MCTL, ACTL, has been presented in [DV90]: ACTL is a modal logic which is interpreted over labeled transition systems with propositional interpretations as states, based on *propositional* CTL and propositional

* Supported by grant no. GRK 184/1-97 of the Deutsche Forschungsgemeinschaft.

action labels. In contrast, MCTL is designed for first-order logic and first-order actions, also providing more detailed modalities, consisting of a coupling mode and a modality in the classical sense.

In [BMP81] and [Wol85], a tableau method for propositional CTL is presented, closely related to the model checking procedure of [CES86]. The branches of the tableau represent paths in a fictive model. Cycles in the tableau are allowed. After termination, which is guaranteed, eventuality formulas have to be postprocessed. The method cannot be extended to first-order variants because the finite number of possible different states is crucial to their concept.

To overcome these problems, in [MS96] a tableau semantics and calculus for first-order CTL has been developed, making basic changes in the handling of eventualities by introducing the ability to abstract from finitely many states in-between. In the resulting tableau calculus we achieved a 1:1-correspondence of branches of the tableau to Kripke structures.

In the present work, this tableau semantics and calculus is extended to labeled Kripke structures and MCTL by integrating actions into this framework in an intuitive way, allowing a formal verification of transition systems.

The paper is structured as follows: In Section 2 the basic notions of first-order logic, Kripke structures, and temporal logics are reviewed. Section 3 introduces the integration of actions by labeled Kripke structures, and the logic MCTL is defined as an extension of CTL to labeled Kripke structures. Section 4 discusses the representation of labeled Kripke structures by semantical tableaux, identifying the relevant entities and developing a syntax of tableau formulas whereas the calculus is given in Section 5. The conclusion in Section 6 completes the paper.

2 Basic Notions

First Order Logic. Every language of first-order logic includes the symbols ")" and "(", the boolean connectives "$\neg$" and "$\vee$", the quantifier "$\exists$", and an infinite set of variables $\mathsf{Var} := \{x_1, x_2, \ldots\}$. A particular language is given by its *signature* Σ consisting of function symbols and predicate symbols with fixed arities $\mathsf{ord}(f)$ resp. $\mathsf{ord}(p)$. Terms, first-order formulas, and 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 interpretation* $\mathbf{I} = (I, \mathbf{U})$ over a signature Σ consists of a non-empty set $\mathbf{U}$ (*universe*) and a mapping I which maps every function symbol $f \in \Sigma$ to a function $I(f) : \mathbf{U}^{\mathsf{ord}(f)} \to \mathbf{U}$ and every predicate symbol $p \in \Sigma$ to a relation $I(p) \subseteq \mathbf{U}^{\mathsf{ord}(p)}$.

A *variable assignment* is a mapping $\chi : \mathsf{Var} \to \mathbf{U}$. For a variable assignment χ, a variable x, and $d \in \mathbf{U}$, 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 to an *evaluation* $\mathbf{I} : \mathsf{Term}_\Sigma \times \Xi \to \mathbf{U}$ s.t. $\mathbf{I}(x, \chi) := \chi(x)$ for $x \in \mathsf{Var}$, and $\mathbf{I}(f(t_1, \ldots, t_n), \chi) := (I(f))(\mathbf{I}(t_1, \chi), \ldots, \mathbf{I}(t_n, \chi))$

for $f \in \Sigma$, $\mathsf{ord}(f) = n$ and $t_1, \ldots, t_n \in \mathsf{Term}_\Sigma$. The truth of a formula F in a first-order interpretation $\mathbf{I}$ under a variable assignment χ, $(\mathbf{I}, \chi) \models_{\mathrm{FO}} F$ is defined as usual.

Kripke Structures.

Definition 1. A first-order *Kripke structure* over a signature Σ is a triple $\mathbf{K} = (\mathbf{G}, \mathbf{R}, \mathbf{M})$ where $\mathbf{G}$ is a set of states, $\mathbf{R} \subseteq \mathbf{G} \times \mathbf{G}$ an *accessibility relation*, and for every $g \in \mathbf{G}$, $\mathbf{M}(g) = (M(g), \mathbf{U}(g))$ is a first-order interpretation of Σ with universe $\mathbf{U}(g)$. $\mathbf{G}$ and $\mathbf{R}$ are called the *frame* of $\mathbf{K}$.

A *path* p in a Kripke structure $\mathbf{K} = (\mathbf{G}, \mathbf{R}, \mathbf{M})$ is a sequence $p = (g_0, g_1, g_2, \ldots)$, $g_i \in \mathbf{G}$ with $\mathbf{R}(g_i, g_{i+1})$ holding for all i. It induces a mapping $p : \mathbb{N} \to \mathbf{G}$ with $p(i) = g_i$. Let $p|_i := (g_i, g_{i+1}, \ldots)$.

In this paper, only Kripke structures with constant universe (i.e. $\mathbf{U}(g) = \mathbf{U}(g')$ for all $g, g' \in \mathbf{G}$) are considered. The notion of a *variable assignment* is then defined as in the first-order case.

Definition 2. For a Kripke structure $\mathbf{K}$ over a signature Σ, the state-independent portion $\Sigma^c \subseteq \Sigma$ consists of all function symbols f s.t. $(M(g))(f) = (M(g'))(f)$ for all $g, g' \in \mathbf{G}$ and all predicate symbols p s.t. $(M(g))(p) = (M(g'))(p)$ for all $g, g' \in \mathbf{G}$. This induces a state-independent evaluation $\mathbf{K}(t)$ for $t \in \mathsf{Term}_{\Sigma^c}$.

Temporal Logics. In [BMP81] and [EH83], the family CTL of temporal logics for branching time is defined in its propositional version for reasoning about Kripke structures in temporal contexts. It uses modal operators $\Box$ ("always"), $\Diamond$ ("sometimes"), $\circ$ ("nexttime"), and until, all interpreted as future modalities, and two path-quantifiers A and E. In CTL, two classes of formulas are distinguished: *state formulas* (SF) holding in states, and *path formulas* (PF) holding on paths.

Definition 3. The syntax of CTL-formulas is given as follows:
- Every first-order formula is in SF(CTL).
- $F, G \in \mathrm{SF(CTL)},\ x \in \mathsf{Var} \ \Rightarrow\ \neg F,\ F \vee G,\ \exists x\, F \in \mathrm{SF(CTL)}$,
- $F, G \in \mathrm{SF(CTL)} \ \Rightarrow\ \circ F,\ (F\ \mathsf{until}\ G) \in \mathrm{PF(CTL)}$,
- $P \in \mathrm{PF(CTL)} \ \Rightarrow\ \neg P \in \mathrm{PF(CTL)}$,
- $P \in \mathrm{PF(CTL)} \ \Rightarrow\ \mathsf{E}P \in \mathrm{SF(CTL)}$,
- Every CTL-state formula is a CTL-formula.

With the abbreviations $F \wedge G := \neg(\neg F \vee \neg G)$, $\forall x\, F := \neg\exists x\, \neg F$, $\Diamond F := \mathsf{true}\ \mathsf{until}\ F$, $\Box F := \neg\Diamond\neg F$, and $\mathsf{A}P := \neg\mathsf{E}\neg P$, those formulas are also state- resp. path formulas.

The definition shows that in CTL every modality (modal operators and negated modal operators) is immediately preceded by a path-quantifier. CTL* is obtained by weakening this requirement [EH83].

Definition 4. The truth of formulas in a first-order Kripke structure $\mathbf{K} = (\mathbf{G}, \mathbf{R}, \mathbf{M})$ is defined separately for state- and path formulas:

Let $g \in \mathbf{G}$ be a state, $p = (g_0, g_1, \ldots)$ a path in $\mathbf{K}$, A an atomic formula, F and G CTL-state formulas, P a CTL-path formula, and χ a variable assignment:

- $(g, \chi) \models A \quad :\Leftrightarrow\ (\mathbf{M}(g), \chi) \models_{\mathrm{FO}} A$.
- $(g, \chi) \models \neg F \ :\Leftrightarrow\ \text{not } (g, \chi) \models F$.

- $(g, \chi) \models F \vee G$ $:\Leftrightarrow$ $(g, \chi) \models F$ or $(g, \chi) \models G$.
- $(g, \chi) \models \exists x\, F$ $:\Leftrightarrow$ there is a $d \in \mathbf{U}$ with $(g, \chi_x^d) \models F$.
- $(p, \chi) \models \circ F$ $:\Leftrightarrow$ $(g_1, \chi) \models F$.
- $(p, \chi) \models F \,\mathsf{until}\, G$ $:\Leftrightarrow$ there is an $i \geq 0$ such that $(g_i, \chi) \models G$ and for all $j : 0 \leq j < i$ $(g_i, \chi) \models F$ holds.
- $(p, \chi) \models \neg P$ $:\Leftrightarrow$ not $(p, \chi) \models P$.
- $(g, \chi) \models \mathsf{E}P$ $:\Leftrightarrow$ there is a path $p = (g_0, g_1, \ldots)$ in $\mathbf{K}$ and an i such that $g_i = g$ and $(p|_i, \chi) \models P$.

3 Labeled Kripke Structures

A labeled transition system is a triple $(S, A, \rightarrow)$, where S is a set of states, A is a set of actions, and $\rightarrow$ is a relation over $S \times A \times S$ providing "labeled transitions" $\rightarrow^{\mathfrak{a}}$ for each action $\mathfrak{a}$. In most cases (cf. [Mil90]), $\rightarrow$ is given by a set of basic rules for elementary actions and a set of rules to derive transitions for composite actions. A multimodal logic for reasoning about labeled transition systems is given in [HM85], using labeled modalities $\langle\mathfrak{a}\rangle$ and $[\mathfrak{a}]$.
Labeled transition systems can even be simply regarded as Kripke structures where the accessibility relation is labeled with actions:

Definition 5. A *labeled Kripke structure* over a signature Σ is a quadruple $\mathbf{K} = (\mathbf{G}, \mathbf{A}, \mathbf{R}, \mathbf{M})$, where $\mathbf{G}$ is a set of elements (states or possible worlds), $\mathbf{A}$ is a set of elements (actions), $\mathbf{R} \subseteq \mathbf{G} \times 2^{\mathbf{A}} \times \mathbf{G}$ is a *labeled accessibility relation* between states, and $\mathbf{M}$ is a function which maps every $g \in \mathbf{G}$ to a first-order interpretation $\mathbf{M}(g)$ over Σ.

A *labeled path* p in a labeled Kripke structure $\mathbf{K} = (\mathbf{G}, \mathbf{A}, \mathbf{R}, \mathbf{M})$ is a sequence $(g_0, \mathfrak{a}_1, g_1, \mathfrak{a}_2, g_2, \ldots)$, $g_i \in \mathbf{G}$, $\mathfrak{a}_i \in 2^{\mathbf{A}}$ such that $(g_i, \mathfrak{a}_{i+1}, g_{i+1}) \in \mathbf{R}$.

Definition 6. The set $\mathbf{A}$ of actions is built from a signature Σ_A of action symbols with given arities (corresponding to action *schemata*, cf. function and predicate symbols) and the universe $\mathbf{U}$:

- For an n-ary action symbol a and $u_1, \ldots, u_n \in \mathbf{U}$, $a(u_1, \ldots, u_n)$ is an action.

Analogous to first-order logic for querying states, transitions are queried by *action formulas*. There, action symbols take the role of predicate symbols:

Definition 7.

- T is an action formula.
- For an n-ary action symbol $a \in \Sigma_A$ and terms $t_1, \ldots, t_n \in \mathsf{Term}_\Sigma$, $a(t_1, \ldots, t_n)$ is an action formula.
- Action formulas are constructed by the usual logical connectives, quantors and variables: For action formulas E and F, and a variable x, $\neg\mathrm{E}$, $\mathrm{E} \vee \mathrm{F}$, and $\exists x : \mathrm{E}$ are action formulas.

Since action formulas query transitions, the transition labels take the role of interpretations: If an elementary action $a(u_1, \ldots, u_n)$ is carried out in some transition, the "predicate" symbol a evaluates to true on $(u_1, \ldots, u_n)$:

Definition 8. Every transition label $\mathfrak{a} \in 2^{\mathbf{A}}$ defines an interpretation of action (predicate) symbols. Let $b \in \Sigma_A$.

$$\mathfrak{a}(b) = \{(u_1, \ldots, u_{\mathsf{ord}(a)}) \in \mathbf{U}^{\mathsf{ord}(b)} \mid b(u_1, \ldots, u_{\mathsf{ord}(b)}) \in \mathfrak{a}\}$$

Together with an interpretation $\mathbf{I}$ over Σ, every transition label $\mathfrak{a}$ induces an evaluation of action formulas:

Definition 9.

- $(\mathfrak{a}, \mathbf{I}, \chi) \models \mathrm{T}$.
- For every action symbol b, $t_1, \ldots, t_n \in \mathsf{Term}_\Sigma$ and $\mathfrak{a} \in 2^{\mathbf{A}}$:
$$(\mathfrak{a}, \mathbf{I}, \chi) \models b(t_1, \ldots, t_n) \;:\Leftrightarrow\; (\mathbf{I}(t_1, \chi), \ldots, \mathbf{I}(t_n, \chi)) \in \mathfrak{a}(b)\ .$$
- The evaluation of composite action formulas is defined in the same way as for first-order logic formulas.

3.1 MCTL: Extension of CTL to Labeled Kripke Structures

MCTL is an extension of first-order CTL to labeled Kripke structures, integrating the language of action formulas as defined above into first-order CTL: The modal operators are labeled – in the style of Dynamic Logic [Har79], Hennessy-Milner Logic $\mathcal{HML}$ [HM85], or ACTL [DV90] – with action formulas querying the respective transitions.

An examination of ACTL, which uses the same modalities as CTL, shows that its (labeled) "nexttime"-operator, X, is not its own dual, as it is in CTL: $\neg(\circ_{CTL}F) \;\Leftrightarrow\; \circ_{CTL}(\neg F)$, but $\neg(\mathsf{X}_\alpha)F \;\Leftrightarrow\; \mathsf{X}_\alpha(\neg F) \vee \neg(\mathsf{X}_\alpha \mathsf{true})$.

This leads to the reflection that there are two relevant coupling modes between actions and facts: One can state that the next transition satisfies α *and* the next state satisfies F, or, that *if* the next transition satisfies α, *then* the next state satisfies F. Thus, in MCTL, the modalities are combinations of a classical modal operator, an action formula, and a coupling mode.

Definition 10. The syntax of MCTL follows the distinction into state- and path formulas known from CTL: MCTL/MCTL* formulas are defined analogously to CTL/CTL* formulas (cf. Section 2) with the additional clause

- For MCTL-state formulas F, G and action formulas E, F, $!\textcircled{\tiny E}\, F$, $?\textcircled{\tiny E}\, F$, $!\boxed{\tiny E}\, F$, $?\boxed{\tiny E}\, F$, $!\Diamond\!\!\!\!\!{\tiny E}\;\, F$, $?\Diamond\!\!\!\!\!{\tiny E}\;\, F$, $!(\mathrm{E}, F)\mathsf{U}(\mathrm{F}, G)$, $?(\mathrm{E}, F)\mathsf{U}(\mathrm{F}, G)$ are MCTL-path formulas.

Notation: Since for complex action formulas the in-operator-notation becomes impossible, $(\!|\mathrm{E}|\!) := \textcircled{\tiny E}$, $[\mathrm{E}] := \boxed{\tiny E}$, and $\langle \mathrm{E} \rangle := \Diamond\!\!\!\!\!{\tiny E}\;$.

In the following, let E, F be action formulas and F, G MCTL-state formulas.

The semantics of the modalities is drawn from the CTL semantics, enriched by the coupling modes:

$\bigcirc$ ("Nexttime"):

$!\textcircled{\tiny E}\, F$: The transition to the next state satisfies E and in the next state F holds.

$?\textcircled{\tiny E}\, F$: If the transition to the next state satisfies E, then F holds in the next state.

$\Box$ ("Always"):

$!\boxed{\tiny E}\, F$: All subsequent transitions satisfy E and in all subsequent states F holds.

$?\boxed{\tiny E}\, F$: If some of the subsequent transitions satisfy E, then in the states reached by these transitions, F holds.

$\Diamond$ ("Sometimes"):

$!\Diamond\!\!\!\!\!{\tiny E}\;\, F$: There exists a transition in the future which satisfies E and in the state reached by this transition, F holds.

$?\Diamond_{\textsc{e}}F$**:** If there exists a first transition satisfying $\textsc{e}$, then in the state reached by this transition, F holds.

U **("Until"):**

$!(\textsc{e}, F)\mathsf{U}(\textsc{f}, G)$**:** There exists a first transition in the future which satisfies $\textsc{f}$ and in the state reached by it, G holds, and in all transitions resp. states before this transition, $\textsc{e}$ resp. F hold.

$?(\textsc{e}, F)\mathsf{U}(\textsc{f}, G)$**:** There exists a first transition in the future which satisfies $\textsc{f}$ and in the state reached by it, G holds, and for all transitions before this transition which satisfy $\textsc{e}$, in the state reached by them, F holds.

Formally there is the following definition of semantics:

Definition 11. Let $\mathbf{K} = (\mathbf{G}, \mathbf{A}, \mathbf{R}, \mathbf{M})$ be a labeled Kripke structure with a labeled path $p = (g_0, \mathfrak{a}_1, g_1, \mathfrak{a}_2, g_2, \ldots)$, and χ a variable assignment. Then, the semantics of the labeled modalities is

$$(p,\chi) \models !\textcircled{\textsc{e}}F \ :\Leftrightarrow (\mathfrak{a}_1, \mathbf{M}(g_0), \chi) \models \textsc{e} \text{ and } (g_1, \chi) \models F.$$
$$(p,\chi) \models ?\textcircled{\textsc{e}}F \ :\Leftrightarrow \text{if } (\mathfrak{a}_1, \mathbf{M}(g_0), \chi) \models \textsc{e}, \text{ then } (g_1, \chi) \models F.$$
$$(p,\chi) \models !\boxed{\textsc{e}}F \ :\Leftrightarrow \text{for all } i\text{: } (\mathfrak{a}_{i+1}, \mathbf{M}(g_i), \chi) \models \textsc{e} \text{ and } (g_{i+1}, \chi) \models F.$$
$$(p,\chi) \models ?\boxed{\textsc{e}}F \ :\Leftrightarrow \text{for all } i\text{: if } (\mathfrak{a}_{i+1}, \mathbf{M}(g_i), \chi) \models \textsc{e}, \text{ then } (g_{i+1}, \chi) \models F.$$
$$(p,\chi) \models !\Diamond_{\textsc{e}}F \ :\Leftrightarrow \text{there is an } i \text{ s.t. } (\mathfrak{a}_{i+1}, \mathbf{M}(g_i), \chi) \models \textsc{e} \text{ and } (g_{i+1}, \chi) \models F.$$
$$(p,\chi) \models ?\Diamond_{\textsc{e}}F \ :\Leftrightarrow \text{if there is a least } i \text{ s.t. } (\mathfrak{a}_{i+1}, \mathbf{M}(g_i), \chi) \models \textsc{e}, \text{ then } (g_{i+1}, \chi) \models F \text{ holds.}$$

$$(p,\chi) \models !(\textsc{e}, F)\mathsf{U}(\textsc{f}, G) \ :\Leftrightarrow$$
$$\text{there is a least } i \text{ s.t. } (\mathfrak{a}_{i+1}, \mathbf{M}(g_i), \chi) \models \textsc{f} \text{ and } (\mathbf{M}(g_{i+1}), \chi) \models G,$$
$$\text{and for all } 0 \leq j < i,\ (\mathfrak{a}_{j+1}, \mathbf{M}(g_j), \chi) \models \textsc{e} \text{ and } (\mathbf{M}(g_{j+1}), \chi) \models F.$$
$$(p,\chi) \models ?(\textsc{e}, F)\mathsf{U}(\textsc{f}, G) \ :\Leftrightarrow$$
$$\text{there is a least } i \text{ s.t. } (\mathfrak{a}_{i+1}, \mathbf{M}(g_i), \chi) \models \textsc{f} \text{ and } (\mathbf{M}(g_{i+1}), \chi) \models G,$$
$$\text{and for all } 0 \leq j < i, \text{ if } (\mathfrak{a}_{j+1}, \mathbf{M}(g_j), \chi) \models \textsc{e} \text{ then } (\mathbf{M}(g_{j+1}), \chi) \models F.$$

Thus, the coupling mode ! always denotes a conjunction of action and formula whereas coupling mode ? always denotes an implication "if action then formula".

Remark: Roughly, all modalities can also be expressed in CTL, with a state-splitted Kripke structure satisfying $\mathbf{M}(g) \models \textsc{e} \ \Leftrightarrow$ all transitions leading to g satisfy $\textsc{e}$, then e.g. $?\textcircled{\textsc{e}}F \ \Leftrightarrow \ \circ(\textsc{e} \to F)$, but in contrast to this, MCTL interprets terms occurring in action formulas according to the interpretation of the state where the transition started from, covering the intention of modeling actions even better. Another advantage of modeling transitions separately is that it can also serve as a base for integrating a process calculus, e.g. CCS [Mil90].

Theorem 12. *The modalities of ACTL can be formulated in MCTL:*
$\mathsf{X}_{\textsc{e}}F \ \Leftrightarrow\ !\textcircled{\textsc{e}}F$, $F\,{}_{\textsc{e}}\mathsf{U}_{\textsc{f}}G \ \Leftrightarrow\ !(\textsc{e}, F)\mathsf{U}(\textsc{f}, G)$, $\mathsf{F}F \ \Leftrightarrow\ !\Diamond_{\textsc{t}}F$ *and* $\mathsf{G}F \ \Leftrightarrow\ !\boxed{\textsc{t}}F$.

Theorem 13. *The semantics of the modal operators of CTL can be expressed by the labeled modal operators of MCTL with the trivial action predicate* $\textsc{t}$ *and coupling mode* !*:*

$$\circ F \Leftrightarrow !\textcircled{\textsc{t}}F \quad , \qquad \Diamond F \Leftrightarrow F \lor !\Diamond_{\textsc{t}}F \quad ,$$
$$\Box F \Leftrightarrow F \land !\boxed{\textsc{t}}F \quad , \qquad F \ \mathsf{until}\ G \Leftrightarrow G \lor (F \land !(\textsc{t}, F)\mathsf{U}(\textsc{t}, G)) \quad .$$

And, similar to CTL,

$$!\Diamond_{\textsc{e}}F \ \Leftrightarrow\ !(\textsc{t}, \mathsf{true})\mathsf{U}(\textsc{e}, F) \qquad \textit{and} \qquad ?\boxed{\textsc{e}}F \ \Leftrightarrow\ \neg(!\Diamond_{\textsc{e}}(\neg F)) \quad .$$

This shows that the versions with coupling mode ! are closely related to classical CTL whereas the coupling mode ? provides an additional aspect of each modality.

Theorem 14. *The duals in MCTL are as follows:*

$$
\begin{array}{lllll}
\neg !\textcircled{\scriptsize E} F & \Leftrightarrow\ ?\textcircled{\scriptsize E}(\neg F) & \textit{and} & \neg ?\textcircled{\scriptsize E} F & \Leftrightarrow\ !\textcircled{\scriptsize E}(\neg F) \quad , \\
\neg ?\boxed{\scriptstyle E} F & \Leftrightarrow\ !\Diamond_{\mathrm{E}}(\neg F) & \textit{and} & \neg !\Diamond_{\mathrm{E}} F & \Leftrightarrow\ ?\boxed{\scriptstyle E}(\neg F) \quad , \\
\neg ?\Diamond_{\mathrm{E}} F & \Leftrightarrow\ ?\Diamond_{\mathrm{E}}(\neg F) & \textit{and} & \neg !\boxed{\scriptstyle E} F & \Leftrightarrow\ !\langle\neg \mathrm{E}\rangle \mathsf{true} \vee !\Diamond_{\mathrm{T}}(\neg F) \quad .
\end{array}
$$

Example 1. $\mathbf{M}(g) \models \mathsf{E}!\ \llparenthesis a(t_1, \ldots, t_n) \rrparenthesis\ \mathsf{true}$ states that in g, the execution of action $a(\mathbf{M}(g)(t_1), \ldots, \mathbf{M}(g)(t_n))$ is possible, while $\mathbf{M}(g) \models \mathsf{A}?\ \llparenthesis a(t_1, \ldots, t_n) \rrparenthesis\ \mathsf{false}$ states that in g, this action cannot be executed.
$\mathbf{M}(g) \models \mathsf{A}?[a(t_1, \ldots, t_n)]\,\mathsf{false}$ states that from state g, in all future, this action can never be executed.
$\mathbf{M}(g) \models \mathsf{E}!\ \llparenthesis a(t_1, \ldots, t_n) \rrparenthesis\ F$ states that by executing $a(\mathbf{M}(g)(t_1), \ldots, \mathbf{M}(g)(t_n))$ (and perhaps some more actions) in g, a state satisfying F is reachable.
$\mathbf{M}(g) \models \mathsf{A}?\langle a(t_1, \ldots, t_n)\rangle\, F$ states that from state g, if $a(\mathbf{M}(g)(t_1), \ldots, \mathbf{M}(g)(t_n))$ is performed in some future transition, a state satisfying F is reached.
Fairness wrt. an action can be expressed in MCTL* by

$$\mathbf{M}(g) \models \mathsf{A}((\Box\Diamond \mathsf{E}!\ \llparenthesis a(t_1, \ldots, t_n) \rrparenthesis\ \mathsf{true}) \rightarrow (!\langle a(t_1, \ldots, t_n)\rangle\, \mathsf{true}))\ .$$

4 Semantical Tableaux Representation for Labeled Kripke Structures

To achieve a strict distinction between the two graph structures "Kripke structure" and "tableau", the terms "path" and "state" will be used for Kripke structures whereas the terms "branch" and "node" will be used for tableaux. The tableau semantics and -calculus extends the one presented for first-order CTL in [MS96]. Like in traditional tableau proving, for a proof of the validity of a formula F, the inconsistency of the formula $\neg F$ is proven by systematically showing that it is impossible to construct a model for $\neg F$. Thus, the situation from first-order theorem proving to find a model for a given set of formulas occurs multiply: Every state is such a first-order interpretation. For this purpose the well-known first-order tableau calculus is embedded in the temporal tableau calculus which is constructed. From these first-order interpretations, a labeled branching time temporal Kripke structure has to be built.

Therefore it is necessary to describe many individual states as well as the relations between them in the tableau. The latter include the ordering of states on a path together with the connections between different paths.

Thus, three kinds of entities have to be described: Elements of the universe inside states, states, and paths, including transitions. In the chosen semantics these will be explicitly named when their existence is stated by a formula:

- Elements of the universe: as in the first-order tableau calculus, a new constant resp. function symbol is introduced by a δ-rule when an $\exists$-quantor is processed.
- States: states are named when required by an existence formula (e.g. $\Diamond_{\mathrm{E}} F$ or $\textcircled{\scriptsize E} F$). In the chosen semantics a newly named state has to be positioned on an existing path, retaining the linear ordering of all states on this path.

- Paths: paths are named when required by an existence formula of the kind $\mathsf{E}P$. A newly introduced path is assumed to branch off in the state where its existence is claimed.

In general, between two known states there can be many other still unknown states. These can be named when needed. Thus, a straightforward dissolving of eventualities at any time is possible.

To allow the naming of states at any position of the model, the descriptions of paths contain, 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.

4.1 Representation

As a conceptional extension of first-order tableaux, every branch of the tableau corresponds to a complete labeled Kripke structure. Here, 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 from [Fit90] augmented with *prefixes* is employed: A state formula F, assumed to be true in a certain state, occurs in the tableau as *state prefixed formula* $\gamma{:}F$. The paths described in the tableau are named by *path descriptors. Path information formulas* contain the information about the prefixes situated on this path. Also, *transition prefixed formulas* will be used, representing information about transitions.

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 (frame part).

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 consists of a set $\hat{\Gamma}$ of *prefix symbols* and a set $\hat{\Lambda}$ of *path symbols*, each containing an infinite set of n-ary prefix- resp. path symbols for every $n \in \mathbb{N}$. The construction of prefixes and path descriptors corresponds to the use of skolem functions in the first-order tableau calculus. Here the prefix- and path symbols take the role of the skolem function symbols. With this, the free variables resulting from invocations of the γ-rule have to be considered. Thus prefixes γ and path descriptors λ are terms consisting of a prefix symbol $\hat{\gamma}$ resp. a path symbol $\hat{\lambda}$ of an arity n and an n-tuple of terms as arguments. Additionally, there is a 0-ary symbol $\hat{\infty}$ that is not a prefix symbol, but is used in a similar way.

Definition 15. Let $\hat{\Gamma}$ be the set of prefix symbols, $\hat{\Lambda}$ the set of path symbols, Σ^c the state-independent portion of Σ. Then the following sets Σ_L, Γ, and Λ are simultaneously recursively enumerable:

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

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 sets of *prefixes* resp. *path descriptors* are given as

$\Gamma := \{\hat{\gamma}(t_1, \ldots, t_n) \mid \hat{\gamma} \in \hat{\Gamma} \text{ is an } n\text{-ary prefix symbol}, t_1, \ldots, t_n \in \mathsf{Term}_{\Sigma_L^c}\}$, and

$\Lambda := \{\hat{\lambda}(t_1, \ldots, t_n) \mid \hat{\lambda} \in \hat{\Lambda} \text{ is an } n\text{-ary path symbol}, t_1, \ldots, t_n \in \mathsf{Term}_{\Sigma_L^c}\}$.

In both sets Γ, $\Lambda \subset \mathsf{Term}_{\Sigma_{\mathcal{T}}}$ of terms it is precisely the leading function symbol which is a prefix- resp. path symbol taken from Σ_F and all argument terms are in $\mathsf{Term}_{\Sigma_L^c}$. Those are interpreted state-independently by $\mathbf{K}$.

An interpretation of $\Sigma_{\mathcal{T}}$ – describing a labeled Kripke structure – is accordingly partitioned: The interpretation of Σ_L is taken over by the set $\{\mathbf{M}(g) \mid g \in \mathbf{G}\}$ of first-order interpretations, and the symbols of Σ_A are interpreted globally as action predicate symbols. Complementary to this, an "interpretation" of the prefix- and path symbols in Σ_F is defined. The corresponding evaluations map prefixes and path descriptors to the entities described by them:

Definition 16. A P&P-interpretation ("prefixes and paths interpretation") of the sets $\hat{\Gamma}$ and $\hat{\Lambda}$ to a labeled Kripke structure $(\mathbf{G}, \mathbf{A}, \mathbf{R}, \mathbf{M})$ with a constant universe $\mathbf{U}$ and a set $\mathbf{P}(\mathbf{K})$ of paths is a triple $\Omega = (\phi, \pi, \psi)$ where

$\phi : \hat{\Lambda} \to \mathbf{U}^n \to \mathbf{P}(\mathbf{K})$ maps every n-ary $\hat{\lambda} \in \hat{\Lambda}$ to a function $\phi(\hat{\lambda}) : \mathbf{U}^n \to \mathbf{P}(\mathbf{K})$ resp. $\phi(\hat{\lambda}) : \mathbf{U}^n \to \mathbb{N} \to (\mathbf{G} \times 2^{\mathbf{A}})$,

$\pi : \hat{\Lambda} \times (\hat{\Gamma} \cup \{\hat{\infty}\}) \to (\mathbf{U}^n \times \mathbf{U}^m) \to \mathbb{N} \cup \{\infty\}$ is an (in general not total) mapping of pairs of n-ary $\hat{\lambda} \in \hat{\Lambda}$ and m-ary $\hat{\gamma} \in \hat{\Gamma}$ to functions $\pi(\hat{\lambda}, \hat{\gamma}) : \mathbf{U}^n \times \mathbf{U}^m \to \mathbb{N} \cup \{\infty\}$ with $\pi(\lambda, \gamma) = \infty \;\Leftrightarrow\; \gamma = \hat{\infty}$, and

$\psi : \hat{\Gamma} \to \mathbf{U}^n \to \mathbf{G}$ maps every n-ary $\hat{\gamma} \in \hat{\Gamma}$ to a function $\psi(\hat{\gamma}) : \mathbf{U}^n \to \mathbf{G}$.

Ω is organized similarly to a first-order interpretation $\mathbf{I} = (I, \mathbf{U})$ if the corresponding mappings, "universes", and induced evaluations are considered:

$$\Phi = (\phi, \mathbf{P}(\mathbf{K})) \quad , \quad \Pi = (\pi, \mathbb{N} \cup \{\infty\}) \quad , \quad \Psi = (\psi, \mathbf{G}) \ .$$

Based on ϕ, π, and ψ, the evaluations $\Phi : \Lambda \times \Xi \to \mathbf{P}(\mathbf{K})$ of path descriptors, $\Pi : \Lambda \times (\Gamma \cup \{\hat{\infty}\}) \times \Xi \to \mathbb{N} \cup \{\infty\}$ of pairs of path descriptors and prefixes, and $\Psi : \Gamma \times \Xi \to \mathbf{G}$ of prefixes are defined as follows:
For $\lambda = \hat{\lambda}(t_1, \ldots, t_n) \in \Lambda$ and $\gamma = \hat{\gamma}(s_1, \ldots, s_m) \in \Gamma$ (thus $t_i, s_i \in \mathsf{Term}_{\Sigma_L^c}$),

$$\begin{aligned}
&\Phi(\lambda, \chi) := (\phi(\hat{\lambda}))(\mathbf{K}(t_1, \chi), \ldots, \mathbf{K}(t_n, \chi)) \quad , \\
&\Pi(\lambda, \gamma, \chi) := (\pi(\hat{\lambda}, \hat{\gamma}))(\mathbf{K}(t_1, \chi), \ldots, \mathbf{K}(t_n, \chi), \mathbf{K}(s_1, \chi), \ldots, \mathbf{K}(s_m, \chi)) \quad \text{, and} \\
&\Psi(\gamma, \chi) := (\psi(\hat{\gamma}))(\mathbf{K}(s_1, \chi), \ldots, \mathbf{K}(s_m, \chi)) \quad .
\end{aligned}$$

Finally, the interpretation of the derived function symbols f_γ is defined state-independently for all $g \in \mathbf{G}$ as

$$(\mathbf{M}(g))(f_\gamma(t_1, \ldots, t_n), \chi) := (\mathbf{M}(\Psi(\gamma, \chi)))(f(t_1, \ldots, t_n), \chi) \quad .$$

Tableau formulas.
On this foundation the syntax used in the tableaux can be worked out: Let $\mathcal{L}$ be the language of state formulas, i.e. SF(MCTL) or SF(MCTL*).
The frame of the Kripke structure is encoded in *path information formulas* of the form $\lambda : [\gamma_0, L_0, \gamma_1, L_1, \ldots, \gamma_n, L_n, \hat{\infty}]$ with $\lambda \in \Lambda$, $\gamma_i \in \Gamma$ and $L_i \in \mathcal{L} \cup \{\circ\}$.
Logical formulas occur in the tableau as *state prefixed formulas* of the form $\gamma{:}F$ with $\gamma \in \Gamma$, $F \in \mathcal{L}$, and the same branch of the tableau containing a path information formula $\lambda : [\ldots, \gamma, \ldots]$. Information about transitions is represented by *transition prefixed formulas* $\lambda[\gamma{\circ}\delta]{:}\mathrm{E}$ with $\lambda \in \Lambda$, $\gamma, \delta \in \Gamma$, E an action formula, and the same branch of the tableau containing a path information formula $\lambda : [\ldots, \gamma, \circ, \delta, \ldots]$.

Following the explicit naming of paths in the calculus, the formulas used internally to the tableau have a more detailed syntax than MCTL/MCTL*-formulas. A syntactic facility to use path descriptors in logical formulas is added: To state the validity of a path formula P on the suffix of a path p (described by a path descriptor λ) beginning in a fixed state g (described by a prefix γ) on that path, the symbol λ can syntactically take the role of a path quantifier. In this role, λ is a *path selector.* This results in the following syntax for node formulas in all tableaux tracing this concept:

Definition 17.
- Every atomic formula is a $\mathcal{TL}$-state formula.
- Every action formula is a $\mathcal{TL}$-action formula.
- With F und G $\mathcal{TL}$-state formulas, $\neg F$, $F \wedge G$, $F \vee G$ and $F \rightarrow G$ are $\mathcal{TL}$-state formulas.
- With F a $\mathcal{TL}$-state formula and x a variable, $\forall x\, F$ and $\exists x\, F$ are $\mathcal{TL}$-state formulas.
- With F and G $\mathcal{TL}$-state formulas, $\circ F$, $\Box F$, $\Diamond F$, and $(F\ \mathsf{until}\ G)$ are $\mathcal{TL}$-path formulas.
- With F, G $\mathcal{TL}$-state formulas and E, F $\mathcal{TL}$-action formulas, $!\textcircled{\scriptsize E}\, F$, $?\textcircled{\scriptsize E}\, F$, $!\boxed{\scriptsize E}\, F$, $?\boxed{\scriptsize E}\, F$, $!\Diamond\!\!\!\!\!\;{\scriptstyle E}\;F$, $?\Diamond\!\!\!\!\!\;{\scriptstyle E}\;F$, $!(\mathrm{E}, F)\mathsf{U}(\mathrm{F}, G)$, $?(\mathrm{E}, F)\mathsf{U}(\mathrm{F}, G)$ are $\mathcal{TL}$-path formulas.
- With P a $\mathcal{TL}$-path formula, $\neg P$ is a $\mathcal{TL}$-path formula.
- With P a $\mathcal{TL}$-path formula, $\mathsf{A}P$ and $\mathsf{E}P$ are $\mathcal{TL}$-state formulas.
- Every $\mathcal{TL}$-state formula is a $\mathcal{TL}$-pre-node formula.
- With P a $\mathcal{TL}$-path formula and $\lambda \in \Lambda$, λP is a $\mathcal{TL}$-pre-node formula.
- With F a $\mathcal{TL}$-pre-node formula and $\gamma \in \Gamma$ a prefix, $\gamma{:}F$ is a $\mathcal{TL}$-state prefixed formula.
- With E a $\mathcal{TL}$-action formula, $\lambda \in \Lambda$, and $\gamma, \delta \in \Gamma$, $\lambda[\gamma \circ \delta]{:}\mathrm{E}$ is a $\mathcal{TL}$-transition prefixed formula.
- All $\mathcal{TL}$-prefixed formulas are $\mathcal{TL}$-node formulas.
- Every path information formula is a $\mathcal{TL}$-node formula.

Semantics.

Definition 18. For a P&P-interpretation $\Omega = (\phi, \pi, \psi)$, a path information formula $I = \lambda : [\gamma_0, L_0, \gamma_1, L_1, \ldots, \gamma_n, L_n, \hat{\infty}]$ is *consistent with* Ω for a variable assignment χ, if every $\hat{\gamma}$ occurs in I at most once, and for all i

$$\Pi(\lambda, \gamma_0, \chi) = 0 \quad , \quad \Pi(\lambda, \gamma_i, \chi) < \Pi(\lambda, \gamma_{i+1}, \chi) \quad ,$$
$$\text{and} \quad \Psi(\gamma_i, \chi) = \Phi(\lambda, \chi, \Pi(\lambda, \gamma_i, \chi)) \quad .$$

This means that the path $\Phi(\lambda, \chi) = (g_0, \mathfrak{a}_1, g_1, \ldots)$ of $\mathbf{K}$ begins in state $g_0 = \Psi(\gamma_0, \chi)$ and passes through the other known states $g_{\Pi(\lambda,\gamma_1,\chi)} = \Psi(\gamma_1, \chi)$, …, $g_{\Pi(\lambda,\gamma_n,\chi)} = \Psi(\gamma_n, \chi)$ in the specified order.

Definition 19. The relation $\models\!\!\!\models$ of a labeled Kripke structure $\mathbf{K} = (\mathbf{G}, \mathbf{A}, \mathbf{R}, \mathbf{M})$ with a set $\mathbf{P}(\mathbf{K})$ of labeled paths, a P&P-interpretation Ω, a set $\mathcal{F}$ of formulas and a variable assignment χ to $\mathsf{free}(\mathcal{F})$ is defined as follows, based on the truth of formulas in labeled Kripke structures, $\models_{\mathrm{MCTL}}$ resp. $\models_{\mathrm{MCTL}^*}$:

- For every state prefixed formula $\gamma : F$, F not containing a path selector:
$$(\mathbf{K}, \Omega, \chi) \models\!\!\!\models \gamma : F \quad :\Leftrightarrow \quad (\Psi(\gamma, \chi), \chi) \models_{\mathrm{MCTL}} F \quad ,$$
i.e. in the state corresponding to the prefix γ under variable assignment χ, the (state) formula F holds.

- For every state prefixed formula $\gamma : F$, $F = \lambda P$ containing a (leading) path selector:
$$(\mathbf{K}, \Omega, \chi) \models\!\models \gamma : \lambda P \quad :\Leftrightarrow \quad (\Phi(\lambda, \chi)|_{\Pi(\lambda,\gamma,\chi)}, \chi) \models P \quad ,$$
i.e. on the suffix of the path $\Phi(\lambda, \chi)$ beginning in the $\Pi(\lambda, \gamma, \chi)$th state (which is $\Psi(\gamma, \chi)$ by consistency), the path formula P holds.
- For every transition prefixed formula $\lambda[\gamma\circ\delta]$:E :
$$(\mathbf{K}, \Omega, \chi) \models\!\models \lambda[\gamma\circ\delta] : \mathrm{E} \quad :\Leftrightarrow \quad (\Phi(\lambda, \chi)|_{\Pi(\lambda,\gamma,\chi)}, \chi) \models !\circledE\,\mathsf{true} \ ,$$
i.e. on the path $\Phi(\lambda, \chi) = (g_0, \mathfrak{a}_1, g_1, \ldots)$, the action $\mathfrak{a}_{\Pi(\lambda,\delta,\chi)}$ (which is performed in the transition from $\Psi(\gamma, \chi)$ to $\Psi(\delta, \chi)$) satisfies E.
- For all path information formulas $I = \lambda : [\gamma_0, L_0, \gamma_1, L_1, \ldots, \gamma_n, L_n, \hat{\infty}]$:
$$(\mathbf{K}, \Omega, \chi) \models\!\models \lambda : [\gamma_0, L_0, \gamma_1, L_1, \ldots, \gamma_n, L_n, \hat{\infty}]$$
iff I is consistent with Ω for the variable assignment χ, and for all $0 \le i \le n$:
$$L_i = \circ \;\Rightarrow\; \Pi(\lambda, \gamma_{i+1}, \chi) = \Pi(\lambda, \gamma_i, \chi){+}1 \quad ,$$
$$L_i \neq \circ \;\Rightarrow\; \text{for all } j \text{ with } \Pi(\lambda, \gamma_i, \chi) < j < \Pi(\lambda, \gamma_{i+1}, \chi) : (\Phi(\lambda, \chi, j), \chi) \models L_i \ ,$$
i.e. if $L_i = \circ$, then $\Pi(\lambda, \gamma_i, \chi)$ and $\Pi(\lambda, \gamma_{i+1}, \chi)$ are immediately successive indices, else for all (finitely, but arbitrary many) states g_j situated between $\Phi(\lambda, \chi, \Pi(\lambda, \gamma_i, \chi))$ and $\Phi(\lambda, \chi, \Pi(\lambda, \gamma_{i+1}, \chi))$ on path $\Phi(\lambda, \chi)$, the relation $(g_j, \chi) \models L_i$ holds.

A set $\mathcal{F}$ of path information formulas and prefixed formulas is *valid* in a labeled Kripke structure $\mathbf{K} = (\mathbf{G}, \mathbf{A}, \mathbf{R}, \mathbf{M})$ with a set $\mathbf{P}(\mathbf{K})$ of paths under a variable assignment χ to $\mathsf{free}(\mathcal{F})$ if there is a P&P-interpretation $\Omega = (\phi, \pi, \psi)$ such that $(\mathbf{K}, \Omega, \chi) \models\!\models \mathcal{F}$. Since a branch of a tableau is a set of formulas like this, validity is a relation on labeled Kripke structures and branches.

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

5 The Tableau Calculus $\mathcal{TL}$

For proving the validity of a formula F, the inconsistency of $\neg F$ is proven: it is shown that there is no labeled Kripke structure $\mathbf{K} = (\mathbf{G}, \mathbf{A}, \mathbf{R}, \mathbf{M})$ with any state $g_0 \in \mathbf{G}$ where F does not hold.
Thus the initialization of the tableau is $\boxed{\dfrac{}{\hat{0} : \neg F}}$.

The tableau calculus is based on the well-known first-order tableau calculus, consisting of α-, β-, γ- and δ-rules and the atomic closure rule [Fit90].

Let F and G be $\mathcal{TL}$-state formulas, A an atomic formula. In the sequel, $F[t/x]$ denotes the formula F with all occurences of x replaced by t.

S–α: $\dfrac{\gamma : F \wedge G}{\begin{array}{c}\gamma : F \\ \gamma : G\end{array}}$	S–γ: $\dfrac{\gamma : \forall x\, F}{\gamma : F[X/x]}$	with X a new variable.
S–β: $\dfrac{\gamma : F \vee G}{\gamma : F \;\big|\; \gamma : G}$	S–δ: $\dfrac{\gamma : \exists x\, F}{\gamma : F[f(\mathsf{free}(T))/x]}$	with f a new function symbol and T the current branch.

Analogous for $\neg(F \vee G)$ (α) , $\neg(F \wedge G)$ (β), $\neg\exists x\, F$ (γ), and $\neg\forall x\, F$ (δ).
Similar rules are given to decompose action formulas into their atomic units:

$T{-}\alpha$:	$\dfrac{\lambda[\alpha\circ\beta] : \mathrm{E} \wedge \mathrm{F}}{\begin{array}{c}\lambda[\alpha\circ\beta] : \mathrm{E} \\ \lambda[\alpha\circ\beta] : \mathrm{F}\end{array}}$	$T{-}\gamma$:	$\dfrac{\lambda[\alpha\circ\beta] : \forall x\, \mathrm{E}}{\lambda[\alpha\circ\beta] : \mathrm{E}[X/x]}$	with X a new variable.
$T{-}\beta$:	$\dfrac{\lambda[\alpha\circ\beta] : \mathrm{E} \vee \mathrm{F}}{\lambda[\alpha\circ\beta] : \mathrm{E} \mid \lambda[\alpha\circ\beta] : \mathrm{F}}$	$T{-}\delta$:	$\dfrac{\lambda[\alpha\circ\beta] : \exists x\, \mathrm{E}}{\lambda[\alpha\circ\beta] : \mathrm{E}[f(\mathsf{free}(T))/x]}$	f and T as for $S{-}\delta$.

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.
Let σ be a substitution, A, B atomic formulas and E, F atomic action formulas.

Closure by states, $S{-}\bot$:

$$\dfrac{\begin{array}{c}\gamma : A \\ \gamma : B \\ \sigma(A) = \neg\sigma(B)\end{array}}{\bot}$$

apply σ_γ to the whole tableau.

Closure by transitions: $T{-}\bot$:

$$\dfrac{\begin{array}{c}\lambda[\gamma\circ\delta] : \mathrm{E} \\ \lambda[\gamma\circ\delta] : \mathrm{F} \\ \sigma(\mathrm{E}) = \neg\sigma(\mathrm{F})\end{array}}{\bot}$$

apply σ_γ to the whole tableau.

For dissolving modalities, the information about the frame of the Kripke structure, which is encoded in the path information formulas, has to be considered. In a single step, a prefixed formula is dissolved "along" a path information formula, inducing the following form of tableau rules:

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

where the premise takes the latest path information formula on the current branch for the path symbol to be considered. The connection between the prefixed formula being dissolved and the path information formula is established by the prefix and, if it exists, the leading path selector of the prefixed formula. For dissolving prefixed formulas, a path quantifier resp. -selector is broken up together with the subsequent modality.

In the sequel, T denotes the current branch of the tableau, $\hat{\gamma}$ is a new prefix symbol and $\hat{\kappa}$ is a new path symbol. P is a path formula, F is a state formula, and E is an action formula.

- For dissolving a formula of the form $\gamma : \mathsf{E}P$, a path satisfying P is named and the path formula is bound to that path:

$$\dfrac{\gamma : \mathsf{E}P}{\begin{array}{c}\hat{\kappa}(\mathsf{free}(T)) : [\gamma, \mathsf{true}, \hat{\infty}] \\ \gamma : \hat{\kappa}(\mathsf{free}(T))P\end{array}}$$

- Formulas of the form $\gamma : \mathsf{A}P$ are dissolved once for every path information formula on this branch containing the prefix γ.
- Formulas of the form $\gamma : \lambda P$ are dissolved along the path information formula for λ.

In the latter cases, the claim that the state described by the current prefix satisfies some formula is decomposed in some less complex claims:
- Which formulas hold in the current state/transition ?
- Which state should be regarded as the "next relevant state/transition" on the path?
- Which formulas hold in this next relevant state/transition ?
- Which formulas hold in all states/transitions in-between ?

The tableau rules for CTL for formulas which are universally path-quantified or explicitly bound to named paths are as follows.

Transition prefixed formulas are only generated when modalities of type $\bigcirc$ are processed. The other modalities are reduced to these cases.

$$\dfrac{\begin{array}{c}\alpha : \mathsf{A}!\textcircled{\scriptsize E}F \\ \lambda : [\ldots,\alpha,L,\beta,\ldots],\ L \neq \circ\end{array}}{\begin{array}{c|c} \lambda : [\ldots,\alpha,\circ,\hat{\gamma}(\mathsf{free}(T)),L,\beta,\ldots] & \text{if } \beta \neq \hat{\infty}\text{:} \\ \hat{\gamma}(\mathsf{free}(T)) : L & \lambda : [\ldots,\alpha,\circ,\beta,\ldots] \\ \hat{\gamma}(\mathsf{free}(T)) : F & \beta : F \\ \lambda[\alpha\circ\hat{\gamma}(\mathsf{free}(T))] : \mathrm{E} & \lambda[\alpha\circ\beta] : \mathrm{E}\end{array}} \qquad \dfrac{\begin{array}{c}\alpha : \mathsf{A}!\textcircled{\scriptsize E}F \\ \lambda : [\ldots,\alpha,\circ,\beta,\ldots]\end{array}}{\begin{array}{c}\beta : F \\ \lambda[\alpha\circ\beta] : \mathrm{E}\end{array}}$$

$$\dfrac{\begin{array}{c}\alpha : \mathsf{A}?\textcircled{\scriptsize E}F \\ \lambda : [\ldots,\alpha,L,\beta,\ldots],\ L \neq \circ\end{array}}{\begin{array}{c|c} \lambda : [\ldots,\alpha,\circ,\hat{\gamma}(\mathsf{free}(T)),L,\beta,\ldots] & \text{if } \beta \neq \hat{\infty}\text{:} \\ \hat{\gamma}(\mathsf{free}(T)) : L & \lambda : [\ldots,\alpha,\circ,\beta,\ldots] \\ \lambda[\alpha\circ\hat{\gamma}(\mathsf{free}(T))] : \neg\mathrm{E} & \lambda[\alpha\circ\beta] : \neg\mathrm{E} \\ & \\ \hline \lambda : [\ldots,\alpha,\circ,\hat{\gamma}(\mathsf{free}(T)),L,\beta,\ldots] & \text{if } \beta \neq \hat{\infty}\text{:} \\ \hat{\gamma}(\mathsf{free}(T)) : L & \lambda : [\ldots,\alpha,\circ,\beta,\ldots] \\ \hat{\gamma}(\mathsf{free}(T)) : F & \beta : F \\ \lambda[\alpha\circ\hat{\gamma}(\mathsf{free}(T))] : \mathrm{E} & \lambda[\alpha\circ\beta] : \mathrm{E}\end{array}} \qquad \dfrac{\begin{array}{c}\alpha : \mathsf{A}?\textcircled{\scriptsize E}F \\ \lambda : [\ldots,\alpha,\circ,\beta,\ldots]\end{array}}{\begin{array}{c|c}\lambda[\alpha\circ\beta] : \neg\mathrm{E} & \beta : F \\ & \lambda[\alpha\circ\beta] : \mathrm{E}\end{array}}$$

Due to limited space, for $\boxed{\scriptstyle E}$, $\Diamond_{\mathrm{E}}$, and $(\mathrm{E},F)\mathsf{U}(\mathrm{F},G)$, only one coupling mode is presented. Also, the negated modalities and the versions where a path formula is explicitly bound to a path λ are omitted.

$$\dfrac{\begin{array}{c}\alpha : \mathsf{A}?\Diamond_{\mathrm{E}}F \\ \lambda : [\ldots,\alpha,L,\beta,\ldots],\ L \neq \circ\end{array}}{\begin{array}{c|c|c} \alpha : \lambda!\textcircled{\scriptsize E}F & \lambda : [\ldots,\alpha,L \cup & \text{if } \beta \neq \hat{\infty}\text{:} \\ & \{\mathsf{A}?\Diamond_{\mathrm{E}}F, \lambda!\,\llparenthesis\neg\mathrm{E}\rrparenthesis\,\mathsf{true}\}, & \lambda : [\ldots,\alpha,L \cup \\ & \hat{\gamma}(\mathsf{free}(T)),L,\beta,\ldots] & \{\mathsf{A}?\Diamond_{\mathrm{E}}F, \lambda!\,\llparenthesis\neg\mathrm{E}\rrparenthesis\,\mathsf{true}\},\beta,\ldots] \\ & \alpha : \lambda!\,\llparenthesis\neg\mathrm{E}\rrparenthesis\,\mathsf{true} & \alpha : \lambda!\,\llparenthesis\neg\mathrm{E}\rrparenthesis\,\mathsf{true} \\ & \hat{\gamma}(\mathsf{free}(T)) : L & \beta : \mathsf{A}?\Diamond_{\mathrm{E}}F \\ & \hat{\gamma}(\mathsf{free}(T)) : \lambda!\textcircled{\scriptsize E}F & \\ & & \text{if } \beta = \hat{\infty}\text{:} \\ & & \lambda : [\ldots,\alpha,L \cup \\ & & \{\mathsf{A}?\Diamond_{\mathrm{E}}F, \lambda!\,\llparenthesis\neg\mathrm{E}\rrparenthesis\,\mathsf{true}\},\beta,\ldots] \\ & & \alpha : \lambda!\,\llparenthesis\neg\mathrm{E}\rrparenthesis\,\mathsf{true}\end{array}}$$

$$
\dfrac{\begin{array}{c}\alpha : \mathsf{A}?\Diamond\!\!\!\!\!{\scriptstyle\mathrm{E}}\;F \\ \lambda : [\ldots,\alpha,\circ,\beta,\ldots]\end{array}}{\begin{array}{c|c}\alpha : \lambda!\text{\textcircled{\scriptsize E}}F & \alpha : \lambda!\,(\!|\neg\mathrm{E}|\!)\,\mathsf{true} \\ & \beta : \mathsf{A}?\Diamond\!\!\!\!\!{\scriptstyle\mathrm{E}}\;F\end{array}}
$$

$$
\dfrac{\begin{array}{c}\alpha : \mathsf{A}!\boxed{\scriptstyle\mathrm{E}}\,F \\ \lambda : [\ldots,\alpha,L,\beta,\ldots],\ L \neq \circ\end{array}}{\begin{array}{c}\lambda : [\ldots,\alpha,L \cup \{\mathsf{A}!\boxed{\scriptstyle\mathrm{E}}\,F, \lambda!\text{\textcircled{\scriptsize E}}F\},\beta,\ldots] \\ \alpha : \mathsf{A}!\text{\textcircled{\scriptsize E}}F \\ \beta : \mathsf{A}!\boxed{\scriptstyle\mathrm{E}}\,F \quad \text{if } \beta \neq \hat{\infty}\end{array}}
\qquad
\dfrac{\begin{array}{c}\alpha : \mathsf{A}!\boxed{\scriptstyle\mathrm{E}}\,F \\ \lambda : [\ldots,\alpha,\circ,\beta,\ldots]\end{array}}{\begin{array}{c}\alpha : \lambda!\text{\textcircled{\scriptsize E}}F \\ \beta : \mathsf{A}!\boxed{\scriptstyle\mathrm{E}}\,F\end{array}}
$$

$$
\dfrac{\begin{array}{c}\alpha : \mathsf{A}!(\mathrm{E},F)\mathsf{U}(\mathrm{F},G) \\ \lambda : [\ldots,\alpha,L,\beta,\ldots],\ L \neq \circ\end{array}}{\begin{array}{c|c|c}\alpha : \lambda!\text{\textcircled{\scriptsize F}}G & \begin{array}{c}\lambda : [\ldots,\alpha,L \cup \\ \{\mathsf{A}!(\mathrm{E},F)\mathsf{U}(\mathrm{F},G), \\ \lambda?\text{\textcircled{\scriptsize F}}(\neg G), \lambda!\text{\textcircled{\scriptsize E}}F\} \\ \hat{\gamma}(\mathsf{free}(T)),\circ,L,\beta,\ldots] \\ \alpha : \lambda?\text{\textcircled{\scriptsize F}}(\neg G) \\ \alpha : \lambda!\text{\textcircled{\scriptsize E}}F \\ \hat{\gamma}(\mathsf{free}(T)) : L \\ \hat{\gamma}(\mathsf{free}(T)) : \lambda!\text{\textcircled{\scriptsize F}}G\end{array} & \begin{array}{c}\text{if } \beta \neq \hat{\infty} : \\ \lambda : [\ldots,\alpha,L \cup \\ \{\mathsf{A}!(\mathrm{E},F)\mathsf{U}(\mathrm{F},G), \\ \lambda?\text{\textcircled{\scriptsize F}}(\neg G), \lambda!\text{\textcircled{\scriptsize E}}F\},\beta,\ldots] \\ \alpha : \lambda?\text{\textcircled{\scriptsize F}}(\neg G) \\ \alpha : \lambda!\text{\textcircled{\scriptsize E}}F \\ \beta : \mathsf{A}!(\mathrm{E},F)\mathsf{U}(\mathrm{F},G)\end{array}\end{array}}
$$

$$
\dfrac{\begin{array}{c}\alpha : \mathsf{A}!(\mathrm{E},F)\mathsf{U}(\mathrm{F},G) \\ \lambda : [\ldots,\alpha,\circ,\beta,\ldots]\end{array}}{\begin{array}{c|c}\alpha : \lambda!\text{\textcircled{\scriptsize F}}G & \begin{array}{c}\alpha : \lambda?\text{\textcircled{\scriptsize F}}(\neg G) \\ \alpha : \lambda!\text{\textcircled{\scriptsize E}}F \\ \beta : \mathsf{A}!(\mathrm{E},F)\mathsf{U}(\mathrm{F},G)\end{array}\end{array}}
$$

Correctness. As usual, for proving correctness, a *Substitution Lemma* has to be shown, stating that for a labeled Kripke structure $\mathbf{K}$, a formula F, a term s, a variable assigment χ, a variable X, and some properties $\Theta(F,\chi)$,

$$\Theta([s/X]F,\chi) \ \Leftrightarrow\ \Theta(F,\chi_X^{\mathbf{K}(s,\chi)})$$

holds. The proof of this shows the necessity of the substitutes s of σ_γ in the atomic closure rule being interpreted state-independently (i.e. $s \in \mathsf{Term}_{\Sigma^c}$): Then s has a global interpretation $\mathbf{K}(s,\chi) \in \mathbf{U}(\mathbf{K})$ needed for the modification of χ.

Theorem 20 (Correctness of $\mathcal{TL}$).

(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 of (a) is done by case-splitting separately for each of the rules. By assumption, there is a labeled Kripke structure $\mathbf{K}$ and a P&P-Interpretation $\Omega = (\phi,\pi,\psi)$ such that for every variable assignment χ there is a branch T_χ in $\mathcal{T}$ with $(\mathbf{K},\Omega,\chi) \models\!\models T_\chi$. In all cases apart from the atomic closure rules, $\mathbf{K}$ and Ω are extended such that they witness the satisfiability of $\mathcal{T}'$. In case of the atomic closure rule the Substitution Lemma guarantees the existence of a branch for every variable assignment to $\mathsf{free}(\mathcal{T}')$. (b) follows directly from (a).

It is well known that first-order CTL is not compact, thus, since MCTL extends CTL, MCTL is also not compact and any calculus for first-order CTL cannot 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. Fairness assumptions are treated in the same way as in [MS96].

6 Conclusion

The presented tableau semantics and -calculus integrates data-oriented and action-oriented aspects which occur when reasoning about transition systems, thus allowing a formal verification of processes given as transition systems. Due to the embedding of first-order tableaux, all recent techniques such as universal formulas, free variables, liberalized δ-rule, and equality-handling can fully be made use of. Because of the inherent size of the search space, including many occurrences of inductions, interactive theorem proving seems appropriate. Since transitions are modeled separately, this calculus can also serve as a base for integrating tableau rules for a process calculus, e.g. CCS [Mil90].

Acknowledgements.
The author thanks GEORG LAUSEN and BERTRAM LUDÄSCHER for many fruitful discussions and their help with improving the presentation of this paper.

References

[BMP81] M. Ben-Ari, Z. Manna, and A. Pnueli. The Temporal Logic of Branching Time. In *8th ACM Symp. on Principles of Programming Languages*, 1981.

[CES86] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.

[DV90] R. DeNicola and F. Vaandrager. Action versus State Based Logics for Transition Systems. In *Semantics of Systems of Concurrent Processes*, Springer LNCS 469, pp. 407 – 419, 1990.

[EH83] E. A. Emerson and J. Y. Halpern. "Sometimes" and "not never" revisited: On Branching Time versus Linear Time in Temporal Logic. In *10th ACM Symp. on Principles of Programming Languages*, 1983.

[Fit90] M. Fitting. *First Order Logic and Automated Theorem Proving.* Springer, New York, 1990.

[Har79] D. Harel. *First-Order Dynamic Logic*, volume 68 of *LNCS*. Springer, 1979.

[HM85] M. Hennessy and R. Milner. Algebraic Laws for Non-determinism and Concurrency. *Journal of the ACM*, 32:137–161, 1985.

[Mil90] R. Milner. *Operational and Algebraic Semantics of Concurrent Processes*, Ch. 19, pp. 1201–1242 of *Handbook of Theoretical Computer Science*, J. v. Leeuwen, ed., Volume B: Formal Models and Semantics, Elsevier, 1990.

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

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