A Tableau Calculus for First-Order Branching Time Logic

Wolfgang May[1] and Peter H. Schmitt[2]

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

[2] Institut für Logik, Komplexität und Deduktionssysteme, Universität Karlsruhe, Germany
pschmitt@ira.uka.de

Review: (first-order) CTL

- Computational Tree Logic, Temporal Logic for Branching Time

(S0) Every first-order formula is a CTL-state formula.

(S1) For F and G CTL-state formulas, $\neg F$, $F \wedge G$, and $F \vee G$ are CTL-state formulas.

Modalities:

(P1) For F and G CTL-state formulas, $\circ F$, $\Box F$, $\Diamond F$, and $(F \mathsf{until} G)$ are CTL-path formulas.

(P2) For P a CTL-path formula, $\neg P$ is a CTL-path formula.

Path quantifiers:

(S2) For P a CTL-path formula, $\mathsf{A}P$ and $\mathsf{E}P$ are CTL-state formulas.

Domain quantifiers:

(SQ) For F a CTL-state formula and x a variable, $\forall x : F$ and $\exists x : F$ are CTL-state formulas.

(F) Every CTL-state formula is a CTL-formula.

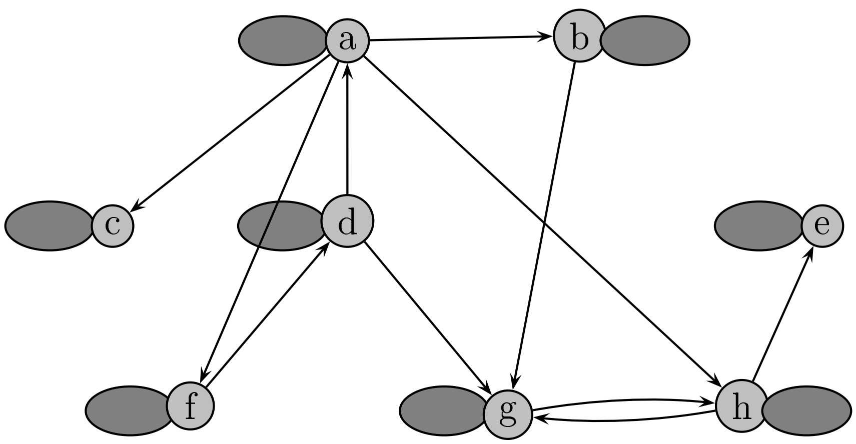
Review: Kripke-Structures

$$\mathbf{K} = (\mathbf{G}, \mathbf{R}, \mathbf{M})$$

G: A set of states (names)

$\mathbf{R} \subset \mathbf{G} \times \mathbf{G}$: An accessibility relation

M: $\mathbf{G} \rightarrow$ {First-order interpretations over a signature Σ}



Tableaux

For proving $\mathcal{F} \vdash F$, show $\mathcal{F} \cup \{F\} \vdash \bot$

Base: First-order tableau calculus: α-, β-, γ-, and δ-rules.

- 1:1-correspondence of branches of the tableau to Kripke-structures.

$\Rightarrow$ explicit description of paths and states by "names".

Entities to be described

- Elements of the universe – like in classical tableau calculus: γ and δ-rules.
- States: every state is a first-order interpretation, i.e. *in* every state the classical tableau calculus can be employed
- Paths: sequences of states and relationships between them

Representation

- Tableau calculus with prefixes, i.e. $\boxed{a : F}$ for "in the state denoted by a, F holds".
- Explicit description of paths: $\boxed{p : [a, \circ, b, \mathcal{F}_1, c, \ldots, \hat{\infty}]}$ for "p goes through a, a's successor is b, the the next named state is c, and in all states between b and c, $\mathcal{F}$ holds".

Strategy:

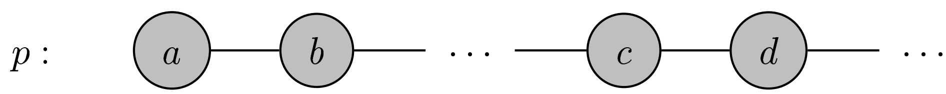
Every entity is "named" exactly when its existence is required by some formula – i.e. a formula which makes some statement about it:

$\exists x : F \Rightarrow$ name an element of the domain as a witness.

$\Diamond F$ (when considering a path p) $\Rightarrow$ name a state on p satisfying F.

$\mathsf{E}F$ (when considering a state a) $\Rightarrow$ name a path through a satisfying F.

$\Rightarrow$ in general, on paths only some states are explicitly named (even the amount of unnamed states is unknown):



States can be named at arbitrary places on a path.

Naming of Entities:

Elements of the universe:

δ-rule: introduces *Skolem function symbols*: $\dfrac{\exists x : F}{F[\hat{f}(\mathsf{free}(T))/x]}$; $\hat{f}$ new.

Elements of the Kripke-frame: states and paths:

Idea: *prefix symbols* ($\hat{\Gamma} := \{\hat{\alpha}, \hat{\beta}, \hat{\gamma}, \ldots\}$) and *path symbols* ($\hat{\Lambda} := \{\hat{\lambda}, \hat{\kappa}, \ldots\}$) to be used in the same way:

- prefixes (state descriptors): $\hat{\alpha}(t_1, \ldots, t_n)$, $t_1, \ldots, t_n \in \mathsf{Term}_\Sigma$, additionally $\hat{\infty}$.

  newly introduced states: $\hat{\alpha}(\mathsf{free}(T))$
- path descriptors: $\hat{\lambda}(t_1, \ldots, t_n)$, $t_1, \ldots, t_n \in \mathsf{Term}_\Sigma$.

  newly introduced paths: $\hat{\lambda}(\mathsf{free}(T))$
- Arguments of prefixes and path descriptors contain only function symbols which are interpreted state independent.

Technicalities

Interpretation of prefix- and path symbols by $\Omega = (\phi, \pi, \psi)$

Evaluation of prefixes and path terms: similarly to a first-order interpretation $\mathbf{I} = (I, \mathbf{U})$ with mapping and "universe":

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

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

$\pi : \hat{\Lambda} \times (\hat{\Gamma} \cup \{\hat{\infty}\}) \to (\mathbf{U}^n \times \mathbf{U}^m) \to \mathbb{N} \cup \{\infty\}$ 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 $\pi(\lambda, \gamma) = \infty \;\Leftrightarrow\; \gamma = \hat{\infty}$
(important if a path goes through some state twice), and

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

State independent terms are evaluated by $\mathbf{K} = (K, \mathbf{U})$

Let $\lambda = \hat{\lambda}(t_1, \ldots, t_n) \in \Lambda$ and $\gamma = \hat{\gamma}(s_1, \ldots, s_m) \in \Gamma$, $t_i, s_i \in \mathsf{Term}_\Sigma$ *interpreted state independently.* Then

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

Validity

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

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

Prefixed formulas:

$$(\mathbf{K}, \Omega, \chi) \models\!\!\models \gamma : F \quad :\Leftrightarrow \quad (\Psi(\gamma, \chi), \chi) \models F$$

Path formulas bound to named paths:

$$(\mathbf{K}, \Omega, \chi) \models\!\!\models \gamma : \lambda P \quad :\Leftrightarrow \ (\Phi(\lambda, \chi)|_{\Pi(\lambda,\gamma,\chi)}, \chi) \models P$$

Path information formulas

$$(\mathbf{K}, \Omega, \chi) \models\!\!\models \lambda : [\gamma_0, L_0, \gamma_1, L_1, \ldots, \gamma_n, L_n, \hat{\infty}]$$

iff I is consistent with Ω for the variable assignment χ, and

for all $0 \leq i \leq n$: $\quad L_i = \circ \Rightarrow \Pi(\lambda, \gamma_{i+1}, \chi) = \Pi(\lambda, \gamma_i, \chi) + 1 \quad ,$

$\quad L_i \neq \circ \Rightarrow$ for all j with $\Pi(\lambda, \gamma_i, \chi) < j < \Pi(\lambda, \gamma_{i+1}, \chi)$:

$$(\Phi(\lambda, \chi, j), \chi) \models L_i \ ,$$

Global Addressing of Entities by State Independent Terms

For substitutions – where elements of the universe have to be addressed in one state and substituted even to other states:

Since states are addressed by prefixes, functions can be bound to states by indexing them with prefixes:

Let f be a state-dependent function symbol and γ a prefix. Then f_γ is a state-independent function symbol: Let t_i be state-independent terms.

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

For a substitution σ, its *localization* to γ, σ_γ, is obtained by replacing each state-dependent function symbol f by f_γ.

The Tableau Calculus

Initialization:

$$\frac{}{\hat{0} : \neg F}$$

Rules:

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

- all rules of first-order tableau calculus (extended with prefixes)
- Closure rule: state-independent substitution

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

- introducing paths: T the current tableau branch

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

Fact:

in **CTL**, propagation of formulas along paths is simplified by the following observation:

For every CTL-path formula $\mathsf{A}P$ where P starts with $\Diamond$, $\Box$, or until, there are CTL-state formulas P_0, P_1 and P_2 s.t.

(1) $(g, \chi) \models P_0 \Rightarrow (g, \chi) \models \mathsf{A}P$,

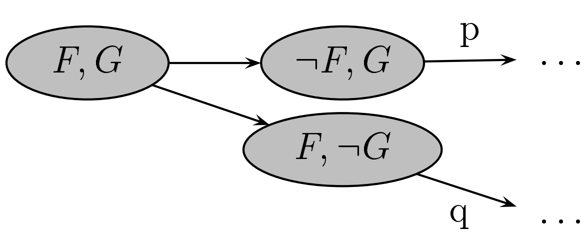
(2) $(g, \chi) \models (\mathsf{A}P \wedge P_1) \Rightarrow (g, \chi) \models \mathsf{A}\circ\mathsf{A}P$,

(3) If $(g, \chi) \models \mathsf{A}P$ and $(g, \chi) \not\models P_0$, then $(g, \chi) \models P_1$ and $(g, \chi) \models P_2$ and for all paths $p = (\ldots, g, \ldots)$ in all successor states h $(h, \chi) \models P_2 \wedge \mathsf{A}P$, until a state k is reached where $(k, \chi) \models P_0$ holds.

Example: F until G: $P_0 := G$, $P_1 := \neg G$, $P_2 := F \wedge \neg G$

This is *not* the case for CTL^*:

$g \models \mathsf{A}\Diamond(F \wedge \Box G)$



remainder on p: $\Box G \vee \Diamond(F \wedge \Box G)$

remainder on q: $\Diamond(F \wedge \Box G)$

$\Rightarrow$ In CTL, A-formulas have the *same* remainders on *all* outgoing paths.

$\Rightarrow$ The propagation of A-formulas on *all* outgoing paths is determined completely by the current state.

$\Rightarrow$ symbol "(A)" for considering *only* proper successor states.

Modalities

The modalities are dissolved in formulas of the form

$$(\mathsf{A}|(\mathsf{A})|\lambda)\ (\neg)^*\ ((\circ|\Diamond|\Box)F|F\ \mathsf{until}\ G)$$

Let T the current tableau branch.

$$\frac{\alpha : \mathsf{A}\Diamond F}{\alpha : F \;\Bigg|\; \begin{array}{c} \alpha : \neg F \\ \alpha : (\mathsf{A})\Diamond F \end{array}}$$

$$\frac{\begin{array}{c} \alpha : (\mathsf{A})\Diamond F \\ \lambda : [\ldots, \alpha, \circ, \beta, \ldots] \end{array}}{\beta : \mathsf{A}\Diamond F}$$

$$\frac{\begin{array}{c} \alpha : (\mathsf{A})\Diamond F \\ \lambda : [\ldots, \alpha, L, \beta, \ldots],\ \ L \neq \circ \end{array}}{\begin{array}{r} \lambda : [\ldots, \alpha, L \wedge \neg F \wedge (\mathsf{A})\Diamond F, \\ \hat{\gamma}(\mathsf{free}(T)), L, \beta, \ldots] \\ \hat{\gamma}(\mathsf{free}(T)) : L \\ \hat{\gamma}(\mathsf{free}(T)) : F \end{array} \;\Bigg|\; \begin{array}{c} \text{if } \beta \neq \hat{\infty}: \\ \lambda : [\ldots, \alpha, L \wedge \neg F \wedge (\mathsf{A})\Diamond F, \beta, \ldots] \\ \beta : \mathsf{A}\Diamond F \end{array}}$$

Create predecessor states:

$$\frac{\begin{array}{c} \lambda : [\ldots, \alpha, L, \beta, \ldots], L \neq \circ \\ \beta \neq \hat{\infty} \end{array}}{\begin{array}{c} \lambda : [\ldots, \alpha, L, \hat{\gamma}(\mathsf{free}(T)), \circ, \beta, \ldots] \\ \hat{\gamma}(\mathsf{free}(T)) : L \end{array} \;\Bigg|\; \lambda : [\ldots, \alpha, \circ, \beta, \ldots]}$$

Results for CTL

- The calculus is correct.
- First-order CTL is not compact.
- Any calculus for first-order CTL is *not* complete.
- This calculus is *not* complete.
- This calculus is complete modulo inductive properties.

Extension to Fairness

Fairness is not expressible in CTL.

Let a formula P of *linear* time temporal logic be of *type* ω iff for every Kripke-structure $\mathbf{K} = (\mathbf{G}, \mathbf{R}, \mathbf{M})$, every path $p \in \mathbf{P}(\mathbf{K})$, every variable assignment χ, and all $n \in \mathbb{N}$

$$(\text{for all } i < n : (p\,|_i, \chi) \models P) \;\Leftrightarrow\; p\,|_n \models P \quad .$$

meaning, that "P can be pushed to infinity".

Fairness, expressed by

$$(\Box\Diamond(\text{action enabled})) \rightarrow \Diamond(\text{action is carried out})$$

is of type ω.

$$\frac{\gamma_i : \mathsf{A}P, \quad P \text{ of type } \omega \qquad \lambda : [\gamma_0, L_0, \gamma_1, L_1, \ldots, \gamma_n, L_n, \hat{\infty}]}{\gamma_n : \lambda P \qquad \text{for all } j > i\text{:} \;\; \gamma_j : \mathsf{A}P}$$

Conclusion

Suitable for *interactive* verification of specifications of processes.