Local model checking in Park's μ-calculus

January 16, 1996

Alexander Kick*

Lehrstuhl Informatik für Ingenieure und Naturwissenschaftler,
Universität Karlsruhe, Am Fasanengarten 5,D-76128 Karlsruhe, Germany
Email: `kick@ira.uka.de`

Abstract

Temporal logic model checking is an automatic verification method for finite-state systems. In global model checking, the truth of a formula (and its subformulae) is determined for all the states in the model. Local model checking procedures are designed for proving that a specific state of the model satisfies a given formula. This may avoid the exhaustive traversal of a model. Also, the proof tree constructed during local model checking can serve as a witness (counterexample) which demonstrates the error in the design and can thus help locating errors.

In [SW91] it was shown how local model checking can be performed in the modal μ-calculus. In this paper, we introduce a tableau system and thus a local model checking method for the more expressive μ-calculus of Park [Par76] and prove its soundness and completeness.

1 Introduction

In the last twenty years many approaches to program verification have been developed. Hoare's partial correctness logic for simple while programs gave an early sound and relatively complete proof system. This approach was subsequently extended to total correctness and richer classes of programs. Dynamic logics offered a more abstract view of Hoare logics, especially in their propositional versions. Pnueli pioneered the use of propositional temporal logics as more general program logics, capable of describing crucial properties of perpetual concurrent systems. A variety of temporal logics have been studied, particularly branching and linear time.

An elegant generalization of propositional dynamic and temporal logics is the propositional modal μ-calculus, due to Pratt [Pra81] and Kozen [Koz83]. The modal μ-calculus has been shown to include Propositional Dynamic Logic, Process Logic, linear time temporal logic, the branching time computation tree logics CTL and CTL* [KP83] [EL86], past temporal operators as well as extended temporal operators like "at all even moments, P holds" [Wol83]. It also generalizes Hennessy-Milner logic [HM85] and thereby provides a natural temporal logic for process theory. Consequently, the modal μ-calculus can be viewed as a *general purpose* program logic.

A hallmark of modal and temporal logics is that their primary truth definition relates elements of a model (states, runs, or whatever) and formulae. But when these logics are

*Supported by DFG GRK 209/2-96

applied to reason about programs it is common to abstract from this relative truth. E.g., in [MP89], models are dispensed with by coding them in the logic as theories: verifying that elements of a model have a crucial property reduces to showing that it is formally derivable within the appropriate theory. This abstraction, however, is not adopted by the model checking method, as pioneered by Clarke, Emerson and Sistla [CES86]. This approach, extended to the modal μ-calculus in [EL86], hinges on constructing algorithms for computing *all* the states of the finite model which have the relevant property.

In contrast to this global model checking, local model checking ([SW91], [Cle90], [Win91], [BS92], etc.) focuses on establishing whether particular elements of a model have a property. This may avoid the exhaustive traversal of a model inherent in global model checking. Also, it permits the use of other techniques that may be specific to the program under consideration. Furthermore, the proof tree constructed during local model checking can serve as a witness for a formula (or counterexample for the negation of the formula). This proof tree can thus help locating errors in the design of a system.

Even more expressive than the modal μ-calculus is Park's μ-calculus [Par76]. E.g., bisimulation equivalence, transitive closure, the transition relation of the product of ω-automata or, more generally, defining new relations on the basis of given transition relations can not be expressed in the modal μ-calculus whereas it can be expressed in Park's μ-calculus [BCM$^+$92].

In [BCM$^+$92], a global model checking algorithm is advocated where the transition relation of a finite model is represented symbolically by BDDs [Bry86]. They derive efficient decision procedures for CTL model checking, satisfiability of linear time temporal logic formulae, strong and weak observational equivalence of finite transition systems and language containment for ω-automata. All the fixpoint computations can be concisely expressed as a formula in Park's μ-calculus. They thus provide a uniform framework based on Park's μ-calculus.

In this paper, we develop a local model checking method for Park's μ-calculus. This method also has the advantages over global model checking of Park's μ-calculus as stated above for the modal μ-calculus. The tableau calculus presented in this paper is mainly inspired by the tableau calculus in [SW91]. However, most of our proofs are totally different from theirs.

The rest of the paper is structured as follows. In Section 2, we summarize Park's μ-calculus. In Section 3, we modify the global model checking algorithm presented in [BCM$^+$92] and state some properties about the information saved during model checking. These properties are needed in Section 4, where we present a local tableau method for model checking Park's μ-calculus and prove its soundness and completeness. In Section 5, we give an example which shall enhance the understanding of the completeness proof of Section 4. In Section 6, we draw some conclusions and hint at further work.

2 Park's μ-calculus

In this section we remind the reader of the syntax and semantics of Park's μ-calculus. In this section, we mainly follow [BCM$^+$92].

2.1 Syntax

We assume we are given a finite signature $\mathcal{S}$. Each symbol in $\mathcal{S}$ is either an *individual variable* or a *relational variable* with some positive arity. There are two syntactic categories: *formulas* and *relational terms*. Formulas have the following form:

1. $R(z_1, z_2, \ldots, z_n)$, where R is an n-ary relational term and $z_1, z_2, \ldots, z_n$ are individual variables in $\mathcal{S}$ not free in R.
2. $\neg f, f \vee g, \exists z[f]$, where f and g are formulas and z is an individual variable in $\mathcal{S}$.

Relational terms of arity n have the following form:

1. P, where P is an n-ary relational variable in $\mathcal{S}$.
2. $\lambda z_1, z_2, \ldots, z_n[f]$, where f is a formula and $z_1, z_2, \ldots, z_n$ are distinct individual variables in $\mathcal{S}$.
3. $\mu P[R]$, where P is an n-ary relational variable in $\mathcal{S}$ and R is an n-ary relational term that is formally monotone with respect to P.

In the following, the word 'term' shall stand for either formula or relational term. We write $p \preceq q$ if p is a subterm of q, and $p \prec q$ if p is a proper subterm of q. A relational variable X is called a μ-variable or ν-variable if X occurs as $\mu X[R]$ or $\nu X[R]$ in a formula, respectively. $\sigma X[R]$ shall stand for either $\mu X[R]$ or $\nu X[R]$. $t\langle u \leftarrow v\rangle$ denotes the formula/relational term formed from substituting term v for the free instances of (individual or relational) variable u in t.

$\forall, \wedge$ are treated as abbreviations in the usual manner. We write $\neg R$ as an abbreviation for $\lambda z_1, \ldots, z_n[\neg R(z_1, \ldots, z_n)]$. $\nu P[R] = \neg\mu P[\neg R\langle P \leftarrow (\neg P)\rangle]$.

In the case of $\exists z[f]$ and $\forall z[f]$ we suppose that there is a free individual variable z occurring in f. This is not a restriction since if not, either of these terms could be rewritten to f. We also suppose (without loss of generality) that all σ-variables are named differently. It is clear that a formula can be transformed into a normal form where $\neg$ is only applied to relational variables. We assume in the rest of the paper that the formulae are in this normal form.

2.2 Semantics

The truth or falsity of a formula is determined with respect to a *model* $\mathcal{M} = (D, I_R, I_D)$, where D is a non-empty set called the *domain* of the model, I_R is the *relational variable interpretation*, and I_D is the *individual variable interpretation*. For a given domain, let $\mathcal{I}_D$ and $\mathcal{I}_R$ be the set of all possible individual variable interpretations and the set of all possible relational variable interpretations, respectively. In this paper, the domain of a model will always be finite.

The semantic function $\mathcal{D}$ maps formulas to elements of

$$(\mathcal{I}_R \rightarrow (\mathcal{I}_D \rightarrow \{true, false\}))$$

and n-ary relational terms to elements of

$$(\mathcal{I}_R \rightarrow (\mathcal{I}_D \rightarrow 2^{D^n}))$$

$$\mathcal{D}(R(z_1, z_2, \ldots, z_n))(I_R)(I_D) = (I_D(z_1), \ldots, I_D(z_n)) \in \mathcal{D}(R)(I_R)(I_D)$$

$$\mathcal{D}(\neg f)(I_R)(I_D) = \neg(\mathcal{D}(f)(I_R)(I_D))$$

$$\mathcal{D}(f \vee g)(I_R)(I_D) = \mathcal{D}(f)(I_R)(I_D) \vee \mathcal{D}(g)(I_R)(I_D)$$

$$\mathcal{D}(\exists z[f])(I_R)(I_D) = \exists e \in D\colon \mathcal{D}(f)(I_R)(I_D\langle z \leftarrow e\rangle)$$

$$\mathcal{D}(P)(I_R)(I_D) = I_R(P)$$

$$\mathcal{D}(\lambda z_1, \ldots, z_n[f])(I_R)(I_D) = \{(e_1, \ldots, e_n) \in D^n | \mathcal{D}(f)(I_R)(I_D\langle z_1 \leftarrow e_1, \ldots z_n \leftarrow e_n\rangle)\}$$

$$\mathcal{D}(\mu P[R])(I_R)(I_D) = Z$$

where Z is the subset of D^n that is the least fixed point (under the inclusion ordering) of the equation

$$Z = \mathcal{D}(R)(I_R\langle P \leftarrow Z\rangle)(I_D)$$

We write $\mathcal{M} \models f$ to indicate that f is true in $\mathcal{M}$ according to the above semantics.

3 Saving information during global model checking

The model checking problem is: given a model $\mathcal{M}$ and a formula f, does $\mathcal{M} \models f$. We give here a modified global model checking algorithm (originally given in [BCM$^+$92]) (Algorithm 1) where information needed for the later pseudo tableau construction in Section 3 is saved. We also define functions and state some properties needed in the next section. This information and these properties are necessary *only for the proof* of the completeness of the local model checking method presented in the next section.

Vectors shall denote the iteration numbers of the fixpoint iterations of subformulae of the form $\mu X.p$ in the model checking algorithm below. $\vec{x} = (x_1, \ldots, x_m) \in \mathbb{N}_0^m$ shall denote a vector of integers. The ordering on these vectors is defined by: $(x) < (y) \Leftrightarrow x < y$, $(x_1, \ldots, x_m) < (y_1, \ldots, y_m) \Leftrightarrow x_1 < y_1 \vee x_1 = y_1 \wedge (x_2, \ldots, x_m) < (y_2, \ldots, y_m)$. For vectors with different lengths we define

$$(x_1, \ldots, x_m) < (y_1, \ldots, y_l) \Leftrightarrow \begin{cases} (x_1, \ldots, x_m) < (y_1, \ldots, y_m) & m \leq l \\ (x_1, \ldots, x_l) < (y_1, \ldots, y_l) & \text{otherwise} \end{cases}$$

$$(x_1, \ldots, x_m) = (y_1, \ldots, y_l) \Leftrightarrow \begin{cases} \forall 1 \leq i \leq m \colon x_i = y_i & m \leq l \\ \forall 1 \leq i \leq l \colon x_i = y_i & m > l \end{cases}$$

Note that $=$ and $<$ on vectors is not transitive. We write $\vec{x} \leq \vec{y} \Leftrightarrow \vec{x} < \vec{y} \vee \vec{x} = \vec{y}$ and $\vec{x} \sqsubseteq \vec{y}$ if $\vec{x}$ is a prefix of $\vec{y}$. $()$ is the empty vector.

The global model checking method in [BCM$^+$92] is based on BDDs [Bry86]. A BDD is represented by place-holder variables $d_1, \ldots, d_n$. Thus, e.g., $\langle d_1 \leftarrow x_1, \ldots, d_n \leftarrow x_n\rangle$ stands for the substitution of the argument variables for the place-holder variables.

Algorithm 1 (Modified model checking algorithm)
For a given model $\mathcal{M}$ and a given relational term R which contains relational variables $P^1, \ldots, P^n$, where $P^1, \ldots P^m$ denote the μ-variables and $P^{m+1} \ldots P^n$ denote the ν-variables in f, $BDDR(R, ())$ determines the set of vectors of elements in D which are in the relation R.

```
function BDDf(f: formula, $I_R$: rel-interp, $(x_1, \ldots, x_k) : \mathbb{N}_0^*$):BDD
begin
  case $f$ of the form
    individual variable : b := BDDAtom(f);
    $f_1 \wedge f_2$ : b := BDDAnd(BDDf($f_1, I_R, (x_1, \ldots, x_k)$), BDDf($f_2, I_R, (x_1, \ldots, x_k)$));
    $f_1 \vee f_2$ : b := BDDOr(BDDf($f_1, I_R, (x_1, \ldots, x_k)$), BDDf($f_2, I_R, (x_1, \ldots, x_k)$));
    $\exists z[f_1]$ : b := BDDExists(z,BDDf($f_1, I_R, (x_1, \ldots, x_k)$));
    $\forall z[f_1]$ : b := BDDAll(z,BDDf($f_1, I_R, (x_1, \ldots, x_k)$));
    $R(z_1, \ldots, z_n)$ : $b := BDDR(R, I_R, (x_1, \ldots, x_k))\langle d_1 \leftarrow z_1, \ldots, d_n \leftarrow z_n\rangle$;
  esac
  $f_{(x_1,\ldots,x_k)} := b$;
  return $b$;
end

function $BDDR(R$: rel-term, $I_R$: rel-interp, $(x_1, \ldots, x_k) : \mathbb{N}_0^*)$:BDD
begin
  case $R$ of the form
    relational variable : $b := I_R(R)$;
    $\neg P$ ($P$ relational variable) : $b :=$ BDDNegate$(I_R(P))$;
    $\lambda z_1, \ldots, z_n[f]$ : $b := BDDf(f, I_R, (x_1, \ldots, x_k))\langle z_1 \leftarrow d_1, \ldots, z_n \leftarrow d_n\rangle$;
    $\mu P[R']$ : $b :=$ lfp$(P, R', I_R,$BDDFalse$,(x_1, \ldots, x_k, 0))$;
    $\nu P[R']$ : $b :=$ gfp$(P, R', I_R,$BDDTrue$,(x_1, \ldots, x_k))$;
  esac
  $R_{(x_1,\ldots,x_k)} := b$;
  return $b$;
end

function lfp($P$:rel-var,$R$:rel-term,$I_R$:rel-interp,$Z$:BDD, $(x_1, \ldots, x_k) : \mathbb{N}_0^*)$:BDD
begin
  $Z' :=$ BDDR$(R, I_R\langle P \leftarrow Z\rangle, (x_1, \ldots, x_k))$;
  if $Z' = Z$ then return $Z$
  else return lfp$(P, R', I_R, Z', (x_1, \ldots, x_{k-1}, x_k + 1))$;
end

function gfp($P$:rel-var,$R$:rel-term,$I_R$:rel-interp,$Z$:BDD, $\vec{x} : \mathbb{N}_0^*)$:BDD
begin
  for all $g \prec \nu P[R]$ for all $\vec{y}$ with $\vec{x} \sqsubseteq \vec{y}$ : $g_{\vec{y}} :=$ BDDFalse;
  $Z' :=$ BDDR$(R, I_R\langle P \leftarrow Z\rangle, \vec{x})$;
  if $Z' = Z$ then return $Z$
  else return gfp$(P, R', I_R, Z', \vec{x})$;
end
```

Remark 1

- *When model checking, we consider only closed terms. We define an ordering on individual variables in a term t: $x_i < x_j \Leftrightarrow$ there is a subterm of t of the form $\alpha \ldots x_i \ldots \beta \ldots x_j \ldots u$ where α, β is λ, $\exists$ or $\forall$.*

- *Within this remark and the following definition, individual variables bound by λ shall also count as free individual variables.*

- *The values calculated for a relational term R during the model checking algorithm is a function $D^n \rightarrow \{true, false\}$. In a similar way, for any subterm $u \preceq t$ a function u_f: $D^n \rightarrow \{true, false\}$ is computed where n is the number of free individual variables in u. $u_f(a_1, \ldots, a_n) = true \Leftrightarrow \mathcal{D}(u\langle x_1 \leftarrow a_1\rangle \ldots \langle x_n \leftarrow a_n\rangle)(I_R)(I_D) = true$ where $x_1, \ldots, x_n$ are all free individual variables in u and $\forall 1 \leq i < j \leq n$: $x_i < x_j$ and I_R and I_D are the respective interpretations during the model checking algorithm in the respective call of the procedure.*

- *In the following, we misuse notation. We drop the index f from u_f.*

- *As a consequence, we associate a set of tuples with a term u model checked, in fact the set of tuples which are in the relation u, i.e., we write $(a_1, \ldots, a_n) \in p$ instead of $p(a_1, \ldots, a_n) = true$. We write $\emptyset$ for the empty relation.*

- *Let Θ be a substitution of individual variables by elements in D. Let $x_1, \ldots, x_n$ be all free individual variables in a term p. If the substitution Θ contains the substitutions $\langle x_1 \leftarrow a_1 \rangle, \ldots, \langle x_n \leftarrow a_n \rangle$ we identify $p\Theta$ with $p(a_1, \ldots, a_n)$.*

- *Let $x_1, \ldots, x_n$ be all free individual variables in a term p. If the substitution Θ contains the substitutions $\langle x_1 \leftarrow a_1 \rangle, \ldots, \langle x_{n-1} \leftarrow a_{n-1} \rangle$ but not a substitution for x_n we write $a_n \in p\Theta \Leftrightarrow (a_1, \ldots, a_n) \in p$.*

- *$p_{\vec{x}}$ and $p^{\vec{x}}$ defined below are the functions computed in the corresponding iteration $\vec{x}$ in the model checking algorithm.*

- *The vectors $\vec{x}$ can be viewed as a kind of signature for the respective subterm as defined in [SE89]. In this paper, we have thus shown how to actually compute a similar type of signature and thus the there defined choice function.*

Note that $X_{(x_1,\ldots,x_{k-1},x_k,\ldots,x_l)} = X_{(x_1,\ldots,x_{k-1},x_k,\ldots,x_j)}$ if $\mu X.p(X)$ is labeled $(\mu X.p(X))_{(x_1,\ldots,x_{k-1})}$ in the model checking algorithm and therefore we define $X_{(x_1,\ldots,x_{k-1},x_k)} = X_{(x_1,\ldots,x_{k-1},x_k,\ldots,x_l)}$. Similarly, if $(\nu X.p(X))_{(x_1,\ldots,x_k)}$ we have $X_{(x_1,\ldots,x_k,\ldots,x_l)} = X_{(x_1,\ldots,x_k,\ldots,x_j)}$ and we define $X_{(x_1,\ldots,x_k)} = X_{(x_1,\ldots,x_k,\ldots,x_l)}$. In the rest of the paper we use these abbreviations.

Definition 1
In the following, let $p \preceq f$, $p_{\vec{x}}$ obtained by $BDDf(f, ())$, $\vec{x} = (x_1, \ldots, x_k) \in \mathbb{N}_0^k, \vec{y} \in \mathbb{N}_0^k$ and p shall contain n free individual variables. We define

- $(p^{(x_1,\ldots,x_k+1)} = p_{(x_1,\ldots,x_k+1)} \setminus p_{(x_1,\ldots,x_k)}) \wedge (p^{(x_1,\ldots,x_{k-1},0)} = p_{(x_1,\ldots,x_{k-1},0)})$

- $l(p(a_1, \ldots, a_n)) = \exists \vec{y} \colon (a_1, \ldots, a_n) \in p^{\vec{y}}$

- $min \colon L_\mu \to (L_\mu \times \mathbb{N}_0^*) \cup \{\bot\}$
$$min(p(a_1, \ldots, a_n)) = \begin{cases} (p(a_1, \ldots, a_n))^{min\{\vec{y} \mid (a_1,\ldots,a_n) \in p^{\vec{y}}\}} & l(p(a_1, \ldots, a_n)) = true \\ \bot & otherwise \end{cases}$$
Note that there can be several $\vec{y}$ with $(a_1, \ldots, a_n) \in p^{\vec{y}}$.

- $v \colon (L_\mu \times \mathbb{N}_0^*) \cup \{\bot\} \to (\mathbb{N}_0^* \cup \{\infty\})$
$$v(g) = \begin{cases} \vec{x} & g = p^{\vec{x}} \\ \infty & g = \bot \end{cases}$$
In the following, let $\forall \vec{x} \in \mathbb{N}_0^* \colon \vec{x} < \infty$.

Lemma 1 *Let $p \wedge q, p \vee q, \exists z[f], \forall z[f], \lambda z_1, \ldots, z_n[f], \nu P[R'], \mu P[R]$ be subterms of formula f model checked by the above algorithm. Let Θ be an arbitrary substitution of free individual variables by elements in D. If $l((p \wedge q)\Theta) = true, l((p \vee q)\Theta) = true, \ldots,$ respectively, in the items below then*

- $v(min(p\Theta)) \leq v(min((p \wedge q)\Theta)) \wedge v(min(q\Theta)) \leq v(min((p \wedge q)\Theta))$ *and*
$v(min((p \wedge q)\Theta)) = v(min(p\Theta)) \vee v(min((p \wedge q)\Theta)) = v(min(q\Theta))$

- $v(min(p\Theta)) \leq v(min((p \vee q)\Theta)) \vee v(min(q)) \leq v(min((p \vee q)\Theta))$ *and*
$v(min((p \vee q)\Theta)) = v(min(p\Theta)) \vee v(min((p \vee q)\Theta)) = v(min(q\Theta))$

- $\exists e \in D$: $v(min((f\langle z \leftarrow e\rangle)\Theta)) \leq v(min((\exists z[f])\Theta))$ *and*
  $\forall e \in D$: $v(min((f\langle z \leftarrow e\rangle)\Theta)) \leq v(min((\exists z[f])\Theta)) \rightarrow$
  $v(min((f\langle z \leftarrow e\rangle)\Theta)) = v(min((\exists z[f])\Theta))$

- $\forall e \in D$: $v((min(f\langle z \leftarrow e\rangle)\Theta)) \leq v(min((\forall z[f])\Theta))$ *and*
  $\exists e \in D$: $v((min(f\langle z \leftarrow e\rangle)\Theta)) \leq v(min((\forall z[f])\Theta)) \rightarrow$
  $\exists e \in D$: $v((min(f\langle z \leftarrow e\rangle)\Theta)) = v(min((\forall z[f])\Theta))$

- $v(min(\lambda z_1, \dots, z_n[f]\Theta)) = v(min(f\Theta))$

- $v(min(\nu P[R](d_1, \dots, d_n))) = v(min(P(d_1, \dots, d_n))) =$
  $v(min(R(d_1, \dots, d_n)))$

- $v(min(\mu P[R](a_1, \dots, a_n))) = v(min(P(a_1, \dots, a_n))) > v(min(R(a_1, \dots, a_n)))$
  In fact, if $v(min(P(a_1, \dots, a_n))) = (x_1, \dots, x_{k-1}, x_k + 1)$ *then*
  $v(min(R(a_1, \dots, a_n))) = (x_1, \dots, x_{k-1}, x_k)$.

Proof: from the way Algorithm 1 works ■

4 Local model checking

In this section we present a tableau system for Park's μ-calculus. We prove its soundness and completeness in the two subsections, respectively.

4.1 A tableau system

The syntax of the μ-calculus is extended to embrace a family of relational constant symbols. Associated with a constant U is a declaration of the form $U = A$ where A is a relational term of the form $\sigma P[R]$. A definition list is a sequence Δ of declarations $U_1 = A_1, \dots, U_n = A_n$ such that $U_i \neq U_j$ whenever $i \neq j$ and such that each constant occurring in A_i is one of $U_1, \dots, U_{i-1}$. Let $dom(\Delta) = \{U_1, \dots, U_n\}$ and $\Delta(U_i) = A_i$. $\Delta.(U = A)$ means appending $U = A$ to the definition list Δ. A definition list Δ is *admissible for B* if every constant occurring in B is declared in Δ. Definition lists are used to keep track of the "dynamically changing" subformulae as fixpoints are unrolled.

We further extend the syntax to incorporate constants of D: if R is an n-ary relational term and $x_1, \dots, x_n$ are elements of D or individual variables in $\mathcal{S}$ not free in R then $R(x_1, \dots, x_n)$ is a formula. The semantics is extended accordingly: $I_D(e) = e$ for $e \in D$, I_R operates on the set of relational variables and constants. L_μ shall denote the set of all formulae and relational terms - allowing the above extension.

Definition 2
If $\Delta.U = A$ is admissible for B then

$$\mathcal{D}(B_{\Delta.U=A})(I_R)(I_D) = \mathcal{D}((B\langle U \leftarrow A\rangle)_\Delta)(I_R)(I_D)$$

Definition 3
If $\Delta = U_1 = A_1, \dots, U_n = A_n$ then $\tilde{\Delta} = \langle U_n \leftarrow A_n\rangle \dots \langle U_1 \leftarrow A_1\rangle$.

When model checking we want to determine the truth of a formula. Therefore, in the local model checking procedure there will only be formulae appearing in the tableau rules.

Definition 4 (Tableau rules TR)

$$\frac{\vdash_\Delta f \vee g}{\vdash_\Delta f} \qquad \frac{\vdash_\Delta f \vee g}{\vdash_\Delta g}$$

$$\frac{\vdash_\Delta f \wedge g}{\vdash_\Delta f \quad \vdash_\Delta g}$$

$$\frac{\vdash_\Delta \exists z[f]}{\vdash_\Delta f\langle z \leftarrow e\rangle} \quad e \in D$$

$$\frac{\vdash_\Delta \forall z[f]}{\vdash_\Delta f\langle z \leftarrow e_1\rangle \ldots \vdash_\Delta f\langle z \leftarrow e_n\rangle} \quad \{e_1 \ldots e_n\} = D$$

$$\frac{\vdash_\Delta (\lambda z_1, \ldots, z_n[f])(d_1, \ldots, d_n)}{\vdash_\Delta f\langle z_1 \leftarrow d_1, \ldots, z_n \leftarrow d_n\rangle}$$

$$\frac{\vdash_\Delta \sigma P[R](d_1, \ldots, d_n)}{\vdash_{\Delta'} U(d_1, \ldots, d_n)} \quad \mathcal{B} \text{ and } \Delta' = \Delta.(U = \sigma P[R])$$

$$\frac{\vdash_\Delta U(d_1, \ldots, d_n)}{\vdash_\Delta R\langle P \leftarrow U\rangle(d_1, \ldots, d_n)} \quad \mathcal{C} \text{ and } \Delta(U) = \sigma P[R]$$

The condition $\mathcal{B}$ is that the new U must be different from any U' where there is a $\vdash_{\Delta''} U'(e_1, \ldots, e_n)$ for some Δ'', $(e_1, \ldots, e_n)$, appearing in the proof tree as a node above the current premise $\vdash_\Delta \sigma P[R](d_1, \ldots, d_n)$. The condition $\mathcal{C}$ is that (for fixed $(d_1, \ldots, d_n)$) no node above the current premise, $\vdash_\Delta U(d_1, \ldots, d_n)$, in the proof tree is labelled $\vdash_{\Delta'} U(d_1, \ldots, d_n)$ for some Δ'.

Lemma 2 *Let t be a formula or relational term. Then*

$$\mathcal{D}(t)(I_R)(I_D\langle y \leftarrow d\rangle) = \mathcal{D}(t\langle y \leftarrow d\rangle)(I_R)(I_D)$$

Proof: by induction on the structure of t:
Induction base:

$$\mathcal{D}(P)(I_R)(I_D\langle y \leftarrow d\rangle) = I_R(P)$$

$$\mathcal{D}(P\langle y \leftarrow d\rangle)(I_R)(I_D) = \mathcal{D}(P)(I_R)(I_D) = I_R(P)$$

Induction steps:

- $$\mathcal{D}(R(z_1, z_2, \ldots, z_n))(I_R)(I_D\langle y \leftarrow d\rangle) =$$

  $$((I_D\langle y \leftarrow d\rangle)(z_1), \ldots, (I_D\langle y \leftarrow d\rangle)(z_n)) \in \mathcal{D}(R)(I_R)(I_D\langle y \leftarrow d\rangle) =$$

  by definition of I_D and induction hypothesis

  $$(I_D(z_1\langle y \leftarrow d\rangle), \ldots, I_D(z_n\langle y \leftarrow d\rangle)) \in \mathcal{D}(R\langle y \leftarrow d\rangle)(I_R)(I_D) =$$

  by definition

  $$\mathcal{D}(R(z_1, z_2, \ldots, z_n)\langle y \leftarrow d\rangle)(I_R)(I_D)$$

- $$\mathcal{D}(\neg f)(I_R)(I_D\langle y \leftarrow d\rangle) = \neg(\mathcal{D}(f)(I_R)(I_D\langle y \leftarrow d\rangle)) =$$
$$\neg(\mathcal{D}(f\langle y \leftarrow d\rangle)(I_R)(I_D)) = \mathcal{D}((\neg f)\langle y \leftarrow d\rangle)(I_R)(I_D)$$
- The case for $\vee$ goes in the same way as the previous case.
- $$\mathcal{D}(\exists z[f])(I_R)(I_D\langle y \leftarrow d\rangle) =$$

  The case $y \equiv z$ is straight-forward. We show the case $y \not\equiv z$
$$\exists e \in D \colon \mathcal{D}(f)(I_R)(I_D\langle y \leftarrow d\rangle\langle z \leftarrow e\rangle) =$$
$$\exists e \in D \colon \mathcal{D}(f)(I_R)(I_D\langle z \leftarrow e\rangle\langle y \leftarrow d\rangle) =$$
$$\exists e \in D \colon \mathcal{D}(f\langle y \leftarrow d\rangle)(I_R)(I_D\langle z \leftarrow e\rangle) =$$
$$\mathcal{D}(\exists z[f\langle y \leftarrow d\rangle])(I_R)(I_D) = \mathcal{D}(\exists z[f]\langle y \leftarrow d\rangle)(I_R)(I_D)$$
- The case for $\mathcal{D}(\lambda z_1, \ldots, z_n[f])(I_R)(I_D)$ goes in the same way as the previous case.
- $$\mathcal{D}(\mu P[R])(I_R)(I_D\langle y \leftarrow d\rangle) = Z$$

  where Z is the subset of D^n that is the least fixed point (under the inclusion ordering) of the equation
$$Z = \mathcal{D}(R)(I_R\langle P \leftarrow Z\rangle)(I_D\langle y \leftarrow d\rangle) = \mathcal{D}(R\langle y \leftarrow d\rangle)(I_R\langle P \leftarrow Z\rangle)(I_D) =$$
$$\mathcal{D}(\mu P[R]\langle y \leftarrow d\rangle)(I_R)(I_D)$$

■

Lemma 3 *Let t be a formula or relational term, Q a relational variable or relational constant, and w a term which does not contain free relational variables or free individual variables. Then*
$$\mathcal{D}(t\langle Q \leftarrow w\rangle)(I_R)(I_D) = \mathcal{D}(t)(I_R\langle Q \leftarrow \mathcal{D}(w)(I_R)(I_D)\rangle)(I_D)$$

Proof: by induction on the structure of t:
Induction base:
if $P = Q$:
$$\mathcal{D}(P\langle Q \leftarrow w\rangle)(I_R)(I_D) = \mathcal{D}(w)(I_R)(I_D)$$
$$\mathcal{D}(P)(I_R\langle Q \leftarrow \mathcal{D}(w)(I_R)(I_D)\rangle)(I_D) = (I_R\langle Q \leftarrow \mathcal{D}(w)(I_R)(I_D)\rangle)(P) = \mathcal{D}(w)(I_R)(I_D)$$
if $P \neq Q$:
$$\mathcal{D}(P\langle Q \leftarrow w\rangle)(I_R)(I_D) = \mathcal{D}(P)(I_R)(I_D) = I_R(P)$$
$$\mathcal{D}(P)(I_R\langle Q \leftarrow \mathcal{D}(w)(I_R)(I_D)\rangle)(I_D) = (I_R\langle Q \leftarrow \mathcal{D}(w)(I_R)(I_D)\rangle)(P) = I_R(P)$$

Induction steps:

- $$\mathcal{D}(R(z_1, z_2, \ldots, z_n)\langle Q \leftarrow w\rangle)(I_R)(I_D) =$$
$$(I_D(z_1), \ldots, I_D(z_n)) \in \mathcal{D}(R\langle Q \leftarrow w\rangle)(I_R)(I_D) =$$
$$(I_D(z_1), \ldots, I_D(z_n)) \in \mathcal{D}(R)(I_R\langle Q \leftarrow \mathcal{D}(w)(I_R)(I_D)\rangle)(I_D) =$$
$$\mathcal{D}(R(z_1, z_2, \ldots, z_n))(I_R\langle Q \leftarrow \mathcal{D}(w)(I_R)(I_D)\rangle)(I_D)$$
- The cases $\vee$ and $\neg$ are straight-forward.
- $$\mathcal{D}(\exists z[f]\langle Q \leftarrow w\rangle)(I_R)(I_D) = \mathcal{D}(\exists z[f\langle Q \leftarrow w\rangle])(I_R)(I_D) =$$
$$\exists e \in D \colon \mathcal{D}(f\langle Q \leftarrow w\rangle)(I_R)(I_D\langle z \leftarrow e\rangle) =$$
$$\exists e \in D \colon \mathcal{D}(f\langle Q \leftarrow w\rangle\langle z \leftarrow e\rangle)(I_R)(I_D) =$$
by Lemma 2
$$\exists e \in D \colon \mathcal{D}(f\langle z \leftarrow e\rangle\langle Q \leftarrow w\rangle)(I_R)(I_D) =$$
since there are no free individual variables in w
$$\exists e \in D \colon \mathcal{D}(f\langle z \leftarrow e\rangle)(I_R\langle Q \leftarrow \mathcal{D}(w)(I_R)(I_D)\rangle)(I_D) =$$
by induction hypothesis
$$\exists e \in D \colon \mathcal{D}(f)(I_R\langle Q \leftarrow \mathcal{D}(w)(I_R)(I_D)\rangle)(I_D\langle z \leftarrow e\rangle) =$$
by Lemma 2
$$\mathcal{D}(\exists z[f])(I_R\langle Q \leftarrow \mathcal{D}(w)(I_R)(I_D)\rangle)(I_D)$$
- The case for $\mathcal{D}(\lambda z_1, \ldots, z_n[f])(I_R)(I_D)$ goes in the same way as the previous case.
- if $P = Q$:
$$\mathcal{D}(\mu P[R]\langle Q \leftarrow w\rangle)(I_R)(I_D) = \mathcal{D}(\mu P[R])(I_R)(I_D) = Z$$
where Z is the subset of D^n that is the least fixed point (under the inclusion ordering) of the equation
$$Z = \mathcal{D}(R)(I_R\langle P \leftarrow Z\rangle)(I_D)$$
$$\mathcal{D}(\mu P[R])(I_R\langle Q \leftarrow \mathcal{D}(w)(I_R)(I_D)\rangle)(I_D) = Y$$
where Y is the least fixpoint of
$$Y = \mathcal{D}(R)(I_R\langle Q \leftarrow \mathcal{D}(w)(I_R)(I_D)\rangle\langle P \leftarrow Y\rangle)(I_D) =$$
$$\mathcal{D}(R)(I_R\langle P \leftarrow Y\rangle)(I_D) = Z$$
if $P \neq Q$:
$$\mathcal{D}(\mu P[R]\langle Q \leftarrow w\rangle)(I_R)(I_D) = \mathcal{D}(\mu P[R\langle Q \leftarrow w\rangle])(I_R)(I_D) = Z$$
$$Z = \mathcal{D}(R\langle Q \leftarrow w\rangle)(I_R\langle P \leftarrow Z\rangle)(I_D) =$$
$$\mathcal{D}(R)(I_R\langle P \leftarrow Z\rangle\langle Q \leftarrow \mathcal{D}(w)(I_R\langle P \leftarrow Z\rangle)(I_D)\rangle)(I_D) =$$

by induction hypothesis

$$\mathcal{D}(R)(I_R\langle P \leftarrow Z\rangle\langle Q \leftarrow \mathcal{D}(w)(I_R)(I_D)\rangle)(I_D) =$$

since P does not occur free in w

$$\mathcal{D}(R)(I_R\langle Q \leftarrow \mathcal{D}(w)(I_R)(I_D)\rangle\langle P \leftarrow Z\rangle)(I_D) =$$

since $P \neq Q$

$$\mathcal{D}(\mu P[R])(I_R\langle Q \leftarrow \mathcal{D}(w)(I_R)(I_D)\rangle)(I_D)$$

■

For the rules to be backwards sound we need to restrict the syntax: given a term p then

- $\forall \sigma P[R] \preceq p$: $\sigma P[R]$ does not contain free individual variables (C1).
- $\forall \sigma P[R] \preceq p$: P does not occur free in p (C2).

The two restrictions are necessary to ensure that syntactic substitutions in the term as they occur in the tableau rules do not interfere with each other. These restrictions are not significant since all applications presented in [BCM$^+$92] conform to them. These restrictions shall hold from now onwards.

Lemma 4 *The tableau rules TR are backwards sound, i.e., if* $\dfrac{\vdash_\Delta u}{\vdash_{\Delta'} v \ldots \vdash_{\Delta''} w}$ *is a rule and* $\mathcal{D}(v_{\Delta'})(I_R)(I_D) = true$ *and ... and* $\mathcal{D}(w_{\Delta''})(I_R)(I_D) = true$ *then* $\mathcal{D}(u_\Delta)(I_R)(I_D) = true$*.*

Proof: It is easy to see that all rules preserve admissibility of Δ. The statement then follows from the definition of $\mathcal{D}$ using Lemmata 2 and 3 and relying on the above restriction.

- $$\mathcal{D}((f \vee g)_\Delta(I_R)(I_D) = \mathcal{D}((f \vee g)\tilde{\Delta})(I_R)(I_D) = \mathcal{D}(f\tilde{\Delta} \vee g\tilde{\Delta})(I_R)(I_D) =$$

  $$\mathcal{D}(f\tilde{\Delta})(I_R)(I_D) \vee \mathcal{D}(g\tilde{\Delta})(I_R)(I_D)$$

  The claim follows immediately.

- The case $f \wedge g$ goes in the same way.

- $$\mathcal{D}(f\langle z \leftarrow e\rangle\tilde{\Delta})(I_R)(I_D) = \mathcal{D}(f\tilde{\Delta}\langle z \leftarrow e\rangle)(I_R)(I_D) =$$

  because of C1, and by Lemma 2 we have

  $$\mathcal{D}(f\tilde{\Delta})(I_R)(I_D\langle z \leftarrow e\rangle)$$

  This implies:

  $$\exists e \in D\colon \mathcal{D}(f\tilde{\Delta})(I_R)(I_D\langle z \leftarrow e\rangle) = \mathcal{D}(\exists z[f\tilde{\Delta}])(I_R)(I_D) =$$

  $$\mathcal{D}(\exists z[f]\tilde{\Delta})(I_R)(I_D)$$

- The cases $\forall z[f]$ and $(\lambda z_1, \ldots, z_n[f])(d_1, \ldots, d_n)$ are similar to the previous case.

- $$\mathcal{D}(U(d_1,\ldots,d_n)\tilde{\Delta}')(I_R)(I_D) = \mathcal{D}(U(d_1,\ldots,d_n)\langle U \leftarrow \sigma P[R]\rangle\tilde{\Delta})(I_R)(I_D) =$$
$$\mathcal{D}(\sigma P[R](d_1,\ldots,d_n)\tilde{\Delta})(I_R)(I_D)$$

- Let $\Delta = \Delta_2.\Delta_3$ and $\Delta_2 = \Delta_1.(U = \sigma P[R])$.
$$\mathcal{D}(U\tilde{\Delta})(I_R)(I_D) =$$
since only variables in Δ_1 appear in $\sigma P[R]$
$$\mathcal{D}(U\tilde{\Delta_2})(I_R)(I_D) =$$
$$\mathcal{D}(\mu P[R]\tilde{\Delta_1})(I_R)(I_D) =$$
by condition C2
$$\mathcal{D}(\mu P[R\tilde{\Delta_1}])(I_R)(I_D) = Z$$
where Z is the least fixpoint of
$$Z = \mathcal{D}(R\tilde{\Delta_1})(I_R\langle P \leftarrow Z\rangle)(I_D)$$
On the other hand,
$$\mathcal{D}(R\langle P \leftarrow U\rangle_{\Delta})(I_R)(I_D) = \mathcal{D}(R\langle P \leftarrow U\rangle_{\Delta_2})(I_R)(I_D) =$$
because of the way the tableau rules are applied only declaration constants in Δ_1 can appear in R
$$\mathcal{D}(R\tilde{\Delta_1}\langle P \leftarrow U\tilde{\Delta_2}\rangle)(I_R)(I_D) =$$
by Lemma 3
$$\mathcal{D}(R\tilde{\Delta_1})(I_R\langle P \leftarrow \mathcal{D}(U\tilde{\Delta_2})(I_R)(I_D)\rangle)(I_D) =$$
$$\mathcal{D}(R\tilde{\Delta_1})(I_R\langle P \leftarrow Z\rangle)(I_D) = Z$$
which shows that the two expressions are equal.

■

A tableau for $\vdash f$, f a formula, is a maximal proof tree whose root is labelled with the sequent $\vdash f$. The sequents labelling the immediate successors of a node are determined by application of one of the rules. Maximality means that no rule applies to a sequent labelling a leaf of a tableau.

Theorem 1 *Every tableau for $\vdash f$ is finite (if D is finite).*

Proof: All rules of TR decrease the length of the formula except the last one. Let $\sigma X[R]$ be a top-level σ-subformula of f where R has arity n. Then the sequent $\vdash_\Delta U(d_1,\ldots,d_n)$ with $\Delta(U) = \sigma X[R]$ can occur at most $|D|^n + 1$ times. This is because there can be only $|D|^n$ different $(d_1,\ldots,d_n)$ and because no other relational variables can cause another sequent $\vdash \sigma X[R]$ (since it is top-level). This U can have spawned at most $|D|^n$ tableaux for proper top-level σ-subformulae of $\sigma X[R]$.

We can repeat this argument for these proper σ-subformulae of $\sigma X[R]$ and their σ-subformulae until a σ-subformula has been reached which does not contain any σ-subformulae. As a consequence, there can be only finitely many vertices in the tableau.

■

Definition 5 (Successful tableau)
Let $\mathcal{M} = (D, I_R, I_D)$ be a model. Then a successful tableau for $\vdash f$ is a finite tableau in which every leaf is labelled by a sequent $\vdash_\Delta p$ fulfilling one of the following requirements:

1. *$p = R(d_1, \ldots, d_n)$ and $\mathcal{D}(R(d_1, \ldots, d_n))(I_R)(I_D) = true$*
2. *$p = \neg R(d_1, \ldots, d_n)$ and $\mathcal{D}(R(d_1, \ldots, d_n))(I_R)(I_D) = false$*
3. *$p = U(d_1, \ldots, d_n)$ and $\Delta(U) = \nu P[R]$*

Theorem 2 *If $\vdash f$ has a successful tableau then $\mathcal{M} \models f$.*

Proof: In a similar way as the proof of a similar theorem for the modal μ-calculus in [SW91], relying on Lemma 4.

■

4.2 Constructing pseudo tableaux

We now show the completeness of the tableau method.

Definition 6 (Reverse substitution)
Let $\Delta = (U_1 = \ldots) \ldots (U_n = \ldots)$. Then $\Phi(U) = Z$ if $\Delta(U) = \sigma Z[R]$; $\overline{f} = ((f\langle U_n \leftarrow \Phi(U_n)\rangle) \ldots \langle U_1 \leftarrow \Phi(U_1)\rangle)$ ($\overline{f}$ is f where the declaration constants are substituted by the original variables in the formula); $\overline{\vdash f} = \overline{f}$; $\overline{f^{\vec{x}}} = \overline{f}^{\vec{x}}$.

We extend the definition of *min* to formulae p containing declaration constants:

$$min(p) = \begin{cases} p^{v(min(\overline{p}))} & l(\overline{p}) = \text{true} \\ \perp & \text{otherwise} \end{cases}$$

Definition 7 (Tableau rules PTR)

$$\frac{\vdash_\Delta (f \vee g)^{\vec{x}}}{choose(\vdash_\Delta (f \vee g)^{\vec{x}})}$$

$$\frac{\vdash_\Delta (f \wedge g)^{\vec{x}}}{\vdash_\Delta min(f) \quad \vdash_\Delta min(g)}$$

$$\frac{\vdash_\Delta (\exists z[f])^{\vec{x}}}{choose(\vdash_\Delta (\exists z[f])^{\vec{x}})}$$

$$\frac{\vdash_\Delta (\forall z[f])^{\vec{x}}}{\vdash_\Delta min(f\langle z \leftarrow e_1\rangle) \ldots \vdash_\Delta min(f\langle z \leftarrow e_n\rangle)} \quad \{e_1 \ldots e_n\} = D$$

$$\frac{\vdash_\Delta ((\lambda z_1, \ldots, z_n[f])(d_1, \ldots, d_n))^{\vec{x}}}{\vdash_\Delta (f\langle z_1 \leftarrow d_1, \ldots, z_n \leftarrow d_n\rangle)^{\vec{x}}}$$

$$\frac{\vdash_\Delta (\sigma P[R](d_1, \ldots, d_n))^{\vec{x}}}{\vdash_{\Delta'} min(U(d_1, \ldots, d_n))} \quad \mathcal{B} \text{ and } \Delta' = \Delta.(U = \sigma P[R])$$

$$\frac{\vdash_\Delta (U(d_1, \ldots, d_n))^{\vec{x}}}{\vdash_\Delta min(R\langle P \leftarrow U\rangle(d_1, \ldots, d_n))} \quad \mathcal{C} \text{ and } \Delta(U) = \sigma P[R]$$

The condition $\mathcal{B}$ is that the new U must be different from any U' where there is a $\vdash_{\Delta''} (U'(e_1, \ldots, e_n))^{\vec{z}}$ for some Δ'', $(e_1, \ldots, e_n)$, $\vec{z}$, appearing in the proof tree as a node above the current premise $\vdash_\Delta (\sigma P[R](d_1, \ldots, d_n))^{\vec{x}}$. The condition $\mathcal{C}$ is that (for fixed $(d_1, \ldots, d_n)$) no node above the current premise, $\vdash_\Delta (U(d_1, \ldots, d_n))^{\vec{x}}$, in the proof tree is labelled $\vdash_{\Delta'} (U(d_1, \ldots, d_n))^{\vec{x}}$ for some Δ'.

```
$choose(\vdash_\Delta (p \vee q)^{\vec{x}}) =$
    choose $u \in \{\vdash_\Delta min(p) | v(min(p)) \leq \vec{x}\} \cup \{\vdash_\Delta min(q) | v(min(q)) \leq \vec{x}\}$;
    return $u$;

$choose(\vdash_\Delta (\exists z[f])^{\vec{x}}) =$
    choose $e \in \{e' | (e') \in f^{\vec{z}} \wedge \vec{z} \leq \vec{x}\}$;
    return $\vdash_\Delta min(f\langle z \leftarrow e \rangle)$;
```

Definition 8 (Pseudo tableau, Successful pseudo tableau)
A pseudo tableau for $\vdash f$ is a tableau for $\vdash min(f)$ where the rules PTR are used instead of TR. A successful pseudo tableau for $\vdash f$ is a finite pseudo tableau for $\vdash f$ in which every leaf is labelled by a sequent $\vdash_\Delta p^{\vec{x}}$ fulfilling the same requirements as for the successful tableau.

Theorem 3 *Every pseudo tableau for $\vdash f$ is finite (if D is finite).*

Proof: In the same way as the proof for the finiteness of a tableau.

■

Theorem 4 *If $\mathcal{M} \models f$ then $\vdash f$ has a successful pseudo tableau.*

Proof: The tableau rules PTR guarantee that for the successors $\vdash g$ also $\mathcal{M} \models g$. Therefore, all nodes in the tableau are true since the tableau is started with a true root. It is clear from the semantics and the model checking algorithm that there are always such successors for nodes which fulfill the side conditions of the rules (e.g., $\mathcal{C}$). Consequently, the tableau construction stops when all current leaves are nodes which do not fulfill the side conditions.

The leaves of the maximal pseudo tableau will therefore be of the three types as in the definition of successful pseudo tableau. All that remains to be shown in order for the pseudo tableau to be successful is that if a leaf is of the form $\vdash U(d_1, \ldots, d_n)$ then $\Delta(U) = \nu P[R]$. This is done in the following argument.

All tableau rules PTR do not increase $\vec{x}$. This follows from Lemma 1. On a path of the proof tree (pseudo tableau), for a given U, $\Delta(U)$ will always be the same because of condition $\mathcal{B}$ and the vectors corresponding to this U will all have the same length. Lemma 1 implies that the last rule actually decreases $\vec{x}$ if $\Delta(U) = \mu P[R]$. Furthermore, the last rule has to be applied before any new $\vdash (U(e_1, \ldots, e_n))^{\vec{y}}$ can be reached. As a consequence, if $\Delta(U) = \mu P[R]$ then for $\vdash (U(d_1, \ldots, d_n))^{\vec{x}}$ and $\vdash (U(e_1, \ldots, e_n))^{\vec{y}}$ lying on a path in the pseudo tableau in this order it holds that $\vec{y} < \vec{x}$. Therefore, $(d_1, \ldots, d_n)$ and $(e_1, \ldots, e_n)$ must be different since there can be at most one $\vec{x}$ with $v(min(U(d_1, \ldots, d_n))) = \vec{x}$ (The unique minimum is always chosen.). It follows that there can not be a leaf $\vdash_\Delta U(d_1, \ldots, d_n)$ with $\Delta(U) = \mu P[R]$.

■

Theorem 5 *If $\mathcal{M} \models f$ then $\vdash f$ has a successful tableau.*

Proof: A successful tableau can be easily obtained from a successful pseudo tableau by stripping off the $\vec{x}$ from all formulae in the pseudo tableau.

■

5 Example

Let model M consist of the two states s_0, s_1 where the relational variable P has the interpretation $P(x) = \text{true} \Leftrightarrow x = s_1$ (We identify the variable name with its interpretation.) and the transition relation R has the interpretation $R(x, y) = \text{true} \Leftrightarrow x = s_0 \wedge y = s_1$. These two relations are represented as BDDs: $P(d) \equiv d = s_1$, $R(d, d') \equiv d = s_0 \wedge d' = s_1$.

Model checking the formula $\mu Q[\lambda s[P(s) \vee \exists t[R(s,t) \wedge Q(t)]]](s_0)$ yields the following trace.

```
$BDDf(\mu Q[\lambda s[P(s) \vee \exists t[R(s,t) \wedge Q(t)]]](s_0), I_R, ())$
    $BDDR(\mu Q[\lambda s[P(s) \vee \exists t[R(s,t) \wedge Q(t)]]]), I_R, ())\langle d \leftarrow s_0 \rangle$
        $lfp(Q, \lambda s[P(s) \vee \exists t[R(s,t) \wedge Q(t)]], I_R, BDDFalse, (0))$
            $BDDR(\lambda s[P(s) \vee \exists t[R(s,t) \wedge Q(t)]], I_R\langle Q \leftarrow BDDFalse \rangle, (0))$
                $BDDf(P(s) \vee \exists t[R(s,t) \wedge Q(t)], I_R\langle Q \leftarrow BDDFalse \rangle, (0))\langle s \leftarrow d \rangle$
                    $BDDf(P(s), I_R\langle Q \leftarrow BDDFalse \rangle, (0))$
                        $BDDR(P, I_R\langle Q \leftarrow BDDFalse \rangle, (0))\langle d \leftarrow s \rangle$
                            $P_{(0)} = (d = s_1)$
                        $P(s)_{(0)} = (s = s_1)$
                    $BDDExists \ldots$
                                    $Q_{(0)} = BDDFalse$
                                $Q(t)_{(0)} = BDDFalse$
                            $(R(s,t) \wedge Q(t))_{(0)} = BDDFalse$
                        $(\exists t[R(s,t) \wedge Q(t)])_{(0)} = BDDFalse$
                    $(P(s) \vee \exists t[R(s,t) \wedge Q(t)])_{(0)} = (s = s_1)$
                $(\lambda s[P(s) \vee \exists t[R(s,t) \wedge Q(t)]])_{(0)} = (d = s_1)$
        $lfp(Q, \lambda s[P(s) \vee \exists t[R(s,t) \wedge Q(t)]], I_R, (d = s_1), (1))$
            $BDDR(\lambda s[P(s) \vee \exists t[R(s,t) \wedge Q(t)]], I_R\langle Q \leftarrow (d = s_1) \rangle, (1))$
                $BDDf(P(s) \vee \exists t[R(s,t) \wedge Q(t)], I_R\langle Q \leftarrow (d = s_1) \rangle, (1))\langle s \leftarrow d \rangle$
                    $BDDf(P(s), I_R\langle Q \leftarrow (d = s_1) \rangle, (0))$
                        $\ldots$
                        $P(s)_{(1)} = (s = s_1)$
                    $BDDf(\exists t[R(s,t) \wedge Q(t)], I_R\langle Q \leftarrow (d = s_1) \rangle, (1))$
                        $BDDf(R(s,t) \wedge Q(t), I_R\langle Q \leftarrow (d = s_1) \rangle, (1))$
                            $BDDf(R(s,t), I_R\langle Q \leftarrow (d = s_1) \rangle, (1))$
                                $BDDR(R, I_R\langle Q \leftarrow (d = s_1) \rangle, (1))\langle d \leftarrow s \rangle\langle d' \leftarrow t \rangle$
                                    $R_{(1)} = (d = s_0 \wedge d' = s_1)$
                                $R(s,t)_{(1)} = (s = s_0 \wedge t = s_1)$
                            $BDDf(Q(t), I_R\langle Q \leftarrow (d = s_1) \rangle, (1))$
                                $BDDR(Q, I_R\langle Q \leftarrow (d = s_1) \rangle, (1))\langle d \leftarrow t \rangle$
                                    $Q_{(1)} = (d = s_1)$
                                $Q(t)_{(1)} = (t = s_1)$
                            $(R(s,t) \wedge Q(t))_{(1)} = (s = s_0 \wedge t = s_1)$
                        $(\exists t[R(s,t) \wedge Q(t)])_{(1)} = (s = s_0)$
                    $(P(s) \vee \exists t[R(s,t) \wedge Q(t)])_{(1)} = (s = s_1 \vee s = s_0)$
                $(\lambda s[P(s) \vee \exists t[R(s,t) \wedge Q(t)]])_{(1)} = (d = s_0 \vee d = s_1)$
        $lfp(Q, \lambda s[P(s) \vee \exists t[R(s,t) \wedge Q(t)]], I_R, (d = s_0 \vee d = s_1), (2))$
            $\ldots$
                                $Q(t)_{(2)} = (t = s_0 \vee t = s_1)$
            $\ldots$
                $(\lambda s[P(s) \vee \exists t[R(s,t) \wedge Q(t)]])_{(2)} = (d = s_0 \vee d = s_1)$
```

$$(\mu Q[\lambda s[P(s) \vee \exists t[R(s,t) \wedge Q(t)]]])_{()} = (d = s_0 \vee d = s_1)$$
$$(\mu Q[\lambda s[P(s) \vee \exists t[R(s,t) \wedge Q(t)]]](s_0))_{()} = (s_0 = s_0 \vee s_0 = s_1) = \text{true}$$

As a consequence, we have, e.g.:

- $Q^{(0)} = BDDFalse, Q^{(1)} = (d = s_1), Q^{(2)} = (d = s_0)$
- $P^{(0)} = (d = s_1), \forall i > 0\colon P^{(i)} = BDDFalse$
- $(\lambda s[P(s) \vee \exists t[R(s,t) \wedge Q(t)]])^{(0)} = (d = s_1)$
  $(\lambda s[P(s) \vee \exists t[R(s,t) \wedge Q(t)]])^{(1)} = (d = s_0)$
  $(\lambda s[P(s) \vee \exists t[R(s,t) \wedge Q(t)]])^{(2)} = BDDFalse$
- $(\exists t[R(s,t) \wedge Q(t)])^{(0)} = BDDFalse$
  $(\exists t[R(s,t) \wedge Q(t)])^{(1)} = (s = s_0)$
  $(\exists t[R(s,t) \wedge Q(t)])^{(2)} = BDDFalse$
- $(R(s,t) \wedge Q(t))^{(0)} = BDDFalse$
  $(R(s,t) \wedge Q(t))^{(1)} = (s = s_0 \wedge t = s_1)$
  $(R(s,t) \wedge Q(t))^{(2)} = BDDFalse$

The pseudo tableau looks as follows:

$$\begin{array}{ccc}
 & \vdash (\mu Q[\lambda s[P(s) \vee \exists t[R(s,t) \wedge Q(t)]]](s_0))^{()} & \\
 & \vdash (U(s_0))^{(2)} & \\
 & \vdash_\Delta ((\lambda s[P(s) \vee \exists t[R(s,t) \wedge U(t)]])(s_0))^{(1)} & \\
 & \vdash_\Delta (P(s_0) \vee \exists t[R(s_0,t) \wedge U(t)])^{(1)} & \\
 & \vdash_\Delta (\exists t[R(s_0,t) \wedge U(t)])^{(1)} & \\
 & \vdash_\Delta (R(s_0,s_1) \wedge U(s_1))^{(1)} & \\
\vdash_\Delta (R(s_0,s_1))^{(0)} & & \vdash_\Delta (U(s_1))^{(1)} \\
 & & \vdash_\Delta ((\lambda s[P(s) \vee \exists t[R(s,t) \wedge U(t)]])(s_1))^{(0)} \\
 & & \vdash_\Delta (P(s_1) \vee \exists t[R(s_1,t) \wedge U(t)])^{(0)} \\
 & & \vdash_\Delta (P(s_1))^{(0)}
\end{array}$$

Where $\Delta = (U = \mu Q[\lambda s[P(s) \vee \exists t[R(s,t) \wedge Q(t)]]])$

For the construction of the pseudo tableau we needed among others the following calculations:

- Calculation of $min(U(s_0))$:

$$min(Q(s_0)) = Q(s_0)^{min\{\vec{y} | s_0 \in Q^{\vec{y}}\}}$$

By Remark 1 we have
$$s_0 \in Q^{\vec{y}} \Leftrightarrow Q^{\vec{y}}(s_0) = \text{true}$$

The only vector for which this is the case is $\vec{y} = (2)$. Therefore

$$min(Q(s_0)) = Q(s_0)^{(2)}$$

and thus
$$min(U(s_0)) = U(s_0)^{v(min(Q(s_0)))} = U(s_0)^{(2)}$$

- Calculation of $choose(\vdash_\Delta (P(s_0) \vee \exists t[R(s_0,t) \wedge U(t)])^{(1)})$:

$$l(P(s_0)) = \text{false}$$

since there is no $\vec{y}$ with $s_0 \in P^{\vec{y}}$. As a consequence, we can immediately deduce that

$$min(\exists t[R(s_0,t) \wedge U(t)]) = (\exists t[R(s_0,t) \wedge U(t)])^{(1)}$$

since by Lemma 1 $v(min((p \vee q)\Theta)) = v(min(p\Theta))$ or $v(min((p \vee q)\Theta)) = v(min(q\Theta))$.

- Calculation of $choose(\vdash_\Delta (\exists t[R(s_0,t) \wedge U(t)])^{(1)})$:

$$(e) \in (R(s_0,t) \wedge Q(t))^{\vec{z}} \Leftrightarrow (e) \in (R(s,t) \wedge Q(t)\langle s \leftarrow s_0\rangle)^{\vec{z}} \Leftrightarrow$$

$$(s_0,e) \in (R(s,t) \wedge Q(t))^{\vec{z}}$$

The only e and $\vec{z}$ for which this expression is true is $e = s_1$ and $\vec{z} = (1)$. As a consequence

$$\{e|(e) \in (R(s_0,t) \wedge Q(t))^{\vec{z}} \wedge \vec{z} \leq (1)\} = \{s_1\}$$

6 Conclusion and further work

In this paper, we have presented a local model checking method for Park's μ-calculus. Since Park's μ-calculus is even more expressive than the modal μ-calculus (and of course also more expressive than CTL, fair CTL, CTL*, etc.) we have thus developed a powerful model checking technique which, in contrast to global model checking, may avoid the exhaustive traversal of a model. Furthermore, a tableau itself can be viewed as a witness showing that a certain property holds in the model. If a property does not hold in a model then the tableau for the negation of the property can be viewed as a counterexample which shows where the error in the model occurs.

The feature of counterexample construction is an important advantage of global fair CTL model checking [CGL93] over other verification techniques. Although it is a very powerful technique, the global model checking procedure for Park's μ-calculus in [BCM$^+$92] lacks this important feature. In further work, we will show how to modify the technique presented in this paper to construct counterexamples in the case of global model checking.

References

[BCM$^+$92] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98(2):142–170, June 1992.

[Bry86] R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, C-35(8), 1986.

[BS92] Bradfield and Stirling. Local model checking for infinite state spaces. *Theoretical Computer Science*, 96, 1992.

[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, April 1986.

[CGL93] E. Clarke, O. Grumberg, and D. Long. Verification tools for finite-state concurrent systems. In de Bakker, editor, *A Decade of Concurrency, REX School/Symposium*, volume 803 of *LNCS*, pages 124 – 175. Springer, 1993.

[Cle90] Rance Cleaveland. Tableau-based model checking in the propositional mu-calculus. *Acta Inf.*, 27:725–747, 1990.

[EL86] E. A. Emerson and C.-L. Lei. Efficient model checking in fragments of the propositional mu-calculus. In *IEEE Symposium on Logic in Computer Science*, pages 267–278, 1986.

[HM85] M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *J. ACM*, 1985.

[Koz83] D. Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27:333–354, 1983.

[KP83] D. Kozen and R. Parikh. A decision procedure for the propositional μ-calculus. In E. Clarke and Kozen D., editors, *Proc. Workshop on Logics of Programs 1983, LNCS 164*, pages 313–325. Springer-Verlag, 1983.

[MP89] Z. Manna and A. Pnueli. The anchored version of the temporal framework. In J. W. de Bakker, W.-P. de Roever, and G. Rozenberg, editors, *Proc. Workshop on Linear time, Branchin Time and Partial Order in Logics for Concurrency*, volume 354 of *LNCS*, pages 201 – 284. Springer, 1989.

[Par76] D. Park. Finiteness is mu-ineffable. *Theoretical Computer Science*, 3(2):173–181, 1976.

[Pra81] Pratt. A decidable mu-calculus (preliminary report). In *IEEE Symposium on Foundations of Computer Science (FOCS)*, 1981.

[SE89] Robert S. Streett and E. Allen Emerson. An automata theoretic decision procedure for the propositional mu-calculus. *Information and Computation*, 81(3):249–264, June 1989.

[SW91] Stirling and Walker. Local model checking in the modal mu-calculus. *Theoretical Computer Science*, 89, 1991.

[Win91] Winskel. A note on model checking the modal nu-calculus. *Theoretical Computer Science*, 83:157 – 167, 1991.

[Wol83] P. Wolper. Temporal logic can be more expressive. *Information and Control*, 56:185 – 194, 1983.