Reduced witnesses for the μ-calculus

November 3, 1995

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

Abstract

Symbolic temporal logic model checking is an automatic verification method. One of its main features is that a counterexample can be constructed when a temporal formula does not hold for the model. Most model checkers so far have restricted the type of formulae that can be checked and for which counterexamples can be constructed to fair CTL formulae. In a previous paper, we have presented an algorithm which constructs counterexamples and witnesses for the whole μ-calculus. The witnesses constructed by this algorithm can be huge, however. In this paper, we show how to construct reduced witnesses.

1 Introduction

Complex state-transition systems occur frequently in the design of sequential circuits and protocols. Symbolic temporal logic model checking [CGL93] has shown in practice to be an extremely useful automatic verification method. In this approach, the state-transition systems are checked with respect to a propositional temporal logic specification.

If the model satisfies the specification the model checker returns true. Otherwise, a counterexample can be constructed, which helps finding the error in the design. The latter facility is one of the most important advantages of model checking over other verification approaches.

*Supported by DFG Vo 287/5-2

The symbolic model checker SMV developed at Carnegie Mellon University ([McM93]) can check fair CTL (FCTL) ([CGL93]) formulae and construct counterexamples for these formulae. Model checkers which can check μ-calculus formulae [Koz83] have greater expressive power, since arbitrary μ-calculus formulae can be checked in contrast to the small subclass FCTL of the μ-calculus, and are more general since many problems can be translated into the μ-calculus.

In [CGMZ94], it is described how to construct counterexamples for FCTL formulae. To our knowledge, noone has yet investigated how to construct counterexamples for arbitrary μ-calculus formulae. To be able to construct counterexamples for μ-calculus formulae, however, is necessary to make a μ-calculus model checker as useful as a CTL model checker. In this paper, we therefore investigate how counterexamples for μ-calculus formulae can be computed.

The rest of the paper is structured as follows. Section 2 consists of preliminaries where the μ-calculus is repeated and some terminology is introduced. In Section 3, we repeat the definition of abstract and concrete witness as defined in [Kic95]. In Section 4, we define reduced witnesses and in Section 5, we present an algorithm to construct such reduced witnesses. Note that we will not care about counterexamples for a formula f in the rest of the paper since counterexamples are simply witnesses for the negation of formula f. In Section 6, we draw some conclusions and compare our reduced witness construction for the whole μ-calculus to the witness construction in [CGMZ94].

2 The modal μ-calculus

In this section we remind the reader of the syntax and semantics of the modal μ-calculus, we introduce some notation and give a slightly modified model checking algorithm which suits our purposes of witness construction. We mainly follow [EL86]

2.1 Syntax and semantics

There are the following syntactic classes:

- *PropCon*, the class of propositional constants $P, Q, R, \ldots$
- *PropVar*, the class of propositional variables $X, Y, Z, \ldots$
- *ProgAt*, the class of program atoms or basic actions $A, B, C, \ldots$
- *Form*, the class of formulae L_μ of the propositional μ-calculus $p, q \ldots$, defined by

  $p ::= P | X | p \wedge q | \neg p | \mu X.p | \langle A \rangle p$

where in $\mu X.p$, p is any formula syntactically monotone in the propositional variable X, i.e., all free occurrences of X in p fall under an even number of negations.

The other connectives are introduced as abbreviations in the usual way: $p \vee q$ abbreviates $\neg(\neg p \wedge \neg q)$, [A] p abbreviates $\neg\langle A\rangle\neg p$ and $\nu X.p(X)$ abbreviates $\neg\mu X.\neg p(\neg X)$.

The semantics of the μ-calculus is defined with respect to a model. A model is a triple $M = (S, R, L)$ where S is a set of states, $R\colon \mathit{ProgAt} \to \mathcal{P}(S \times S)$ is a mapping from program atoms A to a set of state transitions involving A, and $L\colon S \to \mathcal{P}(\mathit{PropCon})$ labels each state with a set of atomic propositions true in that state.

In the rest of the paper, we rarely need the program atoms. Therefore, we introduce the abbreviation $R := \bigcup\{(s,t)|(s,t) \in R(A) \wedge A \in \mathit{ProgAt}\}$. A path in M is a sequence of states: $\pi = s_0 s_1 \ldots$ such that $\forall i \geq 0 : (s_i, s_{i+1}) \in R$. We assume that the models we deal with in the following are finite (i.e., S and ProgAt are finite). The semantics for the modal μ-calculus is given via least and greatest fixpoints. For the details, the reader is referred to [EL86].

The meanings of formulae is defined relative to valuations $\rho\colon \mathit{PropVar} \to \mathcal{P}(S)$. The variant valution $\rho[X/T]$ is defined by

$$\rho[X/T](Y) = \begin{cases} T & Y \equiv X \\ \rho(Y) & \text{otherwise} \end{cases}$$

The set of states satisfying a formula f in a model M with valuation ρ is inductively defined as

$$[\![P]\!]\rho = \{s|P \in L(s)\}$$

$$[\![X]\!]\rho = \rho(X)$$

$$[\![p \wedge q]\!]\rho = [\![p]\!] \cap [\![q]\!]$$

$$[\![\neg p]\!]\rho = S \setminus [\![p]\!]\rho$$

$$[\![\langle A\rangle p]\!]\rho = \{s|\exists t \in S\colon (s,t) \in R(A) \wedge t \in [\![p]\!]\rho\}$$

$$[\![\mu X.p]\!]\rho = \bigcap\{S' \subseteq S|[\![p]\!]\rho[X/S'] \subseteq S'\}$$

We define

$$s, \rho \models p \Leftrightarrow s \in [\![p]\!]\rho$$

2.2 Some terminology

$\langle\rangle$ shall stand for any $\langle A\rangle$, $[\,]$ for any $[A]$. The function $occ(X, f)$ shall return *true* if X occurs in formula f and *false* otherwise.

The terms *subformula, closed formula, bound* and *free variables* are used as usual. We write $p \preceq q$ if p is a subformula of q. A *μ-, ν-subformula* is a subformula whose main connective is μ and ν, respectively.

Alternation depth $\mathcal{A}(f)$ of a formula f is defined in [EL86]. L_{μ_i} shall denote the sublanguage of L_μ with alternation depth i. The formulae $\mu X.p(X)$ and $\nu X.p(X)$ have *iteration depth 1*, denoted by $\mathcal{I}(\mu X.p(X)) = 1, \mathcal{I}(\nu X.p(X)) = 1$, if all proper μ- and ν-subformulae do not contain variable X (supposing that no variables are quantified twice).

$\sigma X.p(X)$ shall stand for either $\mu X.p(X)$ or $\nu X.p(X)$, $\Box$ shall stand for either $[\,]$ or $\langle\rangle$. Let $b(X) = \sigma X.p(X)$ if the latter formula appears as a subformula of an original formula f. We say that X is *in the scope of* $[\,]$, $\langle\rangle$ *in formula* f if X is a subformula of a subformula of f of the form $[\,]q$ and $\langle\rangle q$, respectively.

The length of a formula f ($|f|$) is defined as follows: $|P| = 1$, $|X| = 1$, $|p \wedge q| = |p| + |q| + 1$, $|p \vee q| = |p| + |q| + 1$, $|\langle\rangle p| = |p| + 1$, $|[\,]p| = |p| + 1$, $|\sigma X.p| = |p| + 1$.

A formula is said to be in *propositional normal form (PNF)* provided that no variable is quantified twice and all the negations are applied to atomic propositions only. Note that every formula can be put in PNF. It can be shown by induction on the number of fixpoint iterations that each $\sigma X.p(X)$ can be transformed into a formula without σ or into $\sigma X.p(X)$, where X occurs in $p(X)$ and all occurrences of X in $p(X)$ are in the scope of $\langle\rangle$ or $[\,]$. In the rest of the paper we suppose (without loss of generality) that all μ-calculus formulae are in PNF and closed and all subformulae $\sigma X.p(X)$ fulfill the above constraint.

2.3 Model checking the modal μ-calculus

The model checking problem is: given a model M, a formula f and a state s in M, is $s \in [\![f]\!]\rho$? We do not need to care about ρ, since it can be arbitrary in the case of closed formulae which we consider only. For this reason, we also write $s \models f$ instead of $s, \rho \models f$. To enhance understanding, we give here a model checking algorithm which is a straightforward implementation of the semantics of the μ-calculus.

Algorithm 1 (Model checking algorithm)
input: f, M = (S,R,L)
output: labeling of all states with $l(s,p)$ equal to $s \models p$ for all subformulae p of f

```
procedure mc(f : $L_\mu$)
begin
  for all $\mu X.p \preceq f$ do
    for all $s \in S$ do begin $l(s,X) := false$; $l(s,\mu X.p) := false$ end;
  for all $\nu X.p \preceq f$ do
    for all $s \in S$ do begin $l(s,X) := true$; $l(s,\nu X.p) := true$ end;
  for $i := 2$ to $|f|$ do
  for $p \preceq f \wedge |p| = i$ do
    case $p$ of the form
      $p \wedge q$ : for all $s \in S$ do $l(s,f) := l(s,p) \wedge l(s,q)$
      $p \vee q$ : for all $s \in S$ do $l(s,f) := l(s,p) \vee l(s,q)$
      $\neg p$ : for all $s \in S$ do $l(s,f) := \neg l(s,p)$
      $\langle\rangle p$ : for all $s \in S$ do $l(s,f) := \exists s' \in S\colon (s,s') \in R \wedge l(s,p)$
      $[\,]p$ : for all $s \in S$ do $l(s,f) := \forall s' \in S\colon (s,s') \in R \rightarrow l(s,p)$
      $\mu X.p$ :
        repeat
          changed := false;
          for all $s \in S$ do if $l(s,p) \wedge \neg l(s,X)$ then
            begin $l(s,X)$ := true; $l(s,\mu X.p)$ := true; changed := true; end;
          if changed then mc(p);
        until $\neg$changed
      $\nu X.p$ :
        repeat
          changed := false;
          for all $s \in S$ do if $\neg l(s,p) \wedge l(s,X)$ then
            begin $l(s,X)$ := false; $l(s,\nu X.p)$ := false; changed := true; end;
          if changed then mc(p);
        until $\neg$changed
    esac
end

for all $P \preceq f$ do for all $s \in S$ do $l(s,P) := P \in L(s)$
mc(f);
```

In [EL86] an improved algorithm for model checking is presented on which the following theorem is based.

Theorem 1 (Emerson,Lei) *Model checking can be done in time $O((|M|\cdot|f|)^{\mathcal{A}(f)+1})$ where $|M| = |S| + |R|$ and $|f|$ is the length of formula f.*

3 Witnesses for the μ-calculus

The definition of witnesses for the μ-calculus was motivated and defined in [Kic95]. We repeat it here since we need parts of it for other definitions in the rest of this paper.

Definition 1 (Abstract witness) *An abstract witness $W_{s,f}$ for $s \models_M f$, where $M = (S,R,L)$, $s \in S, f \in L_\mu$, is a triple (V,E,m) where $V \subseteq S$, $E \subseteq R$ and*

$m\colon V \to \mathcal{P}(L_\mu)$. For given s, f and M the components V, E and m of the abstract witness are inductively defined as follows:

1. *$s \in V$, $f \in m(s)$*
2. (a) *$p \wedge q \in m(s)$ implies $p \in m(s)$ and $q \in m(s)$*

   (b) *$p \vee q \in m(s)$ implies:*
   if $s \models p$ and $s \models q$ then $p \in m(s)$ or $q \in m(s)$
   if $s \models p$ and $s \not\models q$ then $p \in m(s)$
   if $s \not\models p$ and $s \models q$ then $q \in m(s)$

   (c) *$\langle\rangle p$ implies: for an arbitrary $s' \in \{s'' | (s, s'') \in R \wedge s'' \models p\}$: $s' \in V$, $p \in m(s')$, $(s, s') \in E$*

   (d) *$[\,]p$ implies: for all $s' \in \{s'' | (s, s'') \in R\}$: $s' \in V$, $p \in m(s')$, $(s, s') \in E$*

   (e) *$\sigma X.p(X) \in m(s)$ implies $p(X) \in m(s)$*

   (f) *$X \in m(s)$ implies $b(X) \in m(s)$*
3. *No other states, edges and formulae belong to V, E and $m(s)$, s arbitrary, respectively.*

Definition 1 is motivated by the premise that we need to demonstrate $s \models p$ for a formula p and a fixed state s just once.

Definition 1 does not ensure that the μ-paths - paths produced by subsequent unwinding of $\mu X.p$ (Definition 3) - are dealt with properly. For this reason, we define 'concrete witnesses' in Definition 4.

In the following definition, we define the intermediate paths between subsequent states which model $\mu X.p$ on a μ-path.

Definition 2 (Xpath) *For a witness $W = (V, E, m)$ and model M, a finite sequence of states π is an Xpath for a subformula g of a formula f and a propositional variable X if $Xp(\pi, g, X)$ where*

$$Xp\colon S^* \times L_\mu \times PropVar \to \{true, false\}$$

$$Xp(s\pi, p \wedge q, X) \Leftrightarrow (p \wedge q) \in m(s) \wedge (Xp(s\pi, p, X) \vee Xp(s\pi, q, X))$$

$$Xp(s\pi, p \vee q, X) \Leftrightarrow (p \vee q) \in m(s) \wedge (p \in m(s) \wedge Xp(s\pi, p, X) \vee q \in m(s) \wedge Xp(s\pi, q, X))$$

$$Xp(ss'\pi, \Box p, X) \Leftrightarrow \Box p \in m(s) \wedge (s, s') \in E \wedge Xp(s'\pi, p, X)$$

$$Xp(s\pi, \sigma Y.p, X) \Leftrightarrow \sigma Y.p \in m(s) \wedge Xp(s\pi, p, X)$$

$$Xp(s\pi, Y, X) \Leftrightarrow Y \in m(s) \wedge Xp(s\pi, b(Y), X) \wedge (b(Y) \prec \sigma X.p)$$

$$Xp(s, X, X) \Leftrightarrow X \in m(s)$$

where $(Y \not\equiv X)$.

Definition 3 (μ-path in a witness) *A μ-path π in a witness $W = (V, E, m)$ for a formula $\mu X.p(X)$ is a finite sequence of states $s_0 s_1 \ldots s_m$ with $(\forall 0 \le i \le m\colon s_i \in V) \wedge \mu X.p(X) \in m(s_0) \wedge (\forall 0 \le i < m\colon \exists \rho = \rho_0 \ldots \rho_n\colon Xp(\rho, \mu X.p(X), X) \wedge \rho_0 = s_i \wedge \rho_n = s_{i+1})$. π_i shall denote the ith state in the μ-path. The set of all μ-paths in a witness W for a formula $\mu X.p(X)$ is denoted by $Mp(\mu X.p(X))$.*

Definition 4 (Concrete witness) *A concrete witness for $s \models_M f$, $M = (S, R, L)$, $s \in S$, $f \in L_\mu$ is an abstract witness $W = (V, E, m)$ for $s \models_M f$ with the additional constraint*

$$\begin{aligned}\forall \mu X.p(X) \preceq f\colon \forall s \in V\colon \mu X.p(X) \in m(s) \to \\ \exists \pi = \pi_0 \ldots \pi_n \in Mp(\mu X.p(X))\colon \pi_0 = s \wedge p(false) \in m(\pi_n) \quad (1)\end{aligned}$$

4 Reduced witnesses

4.1 Example

We use an example to give an intuition for the construction of a witness for a formula in the μ-calculus. What is the witness for

$$\mu X.\nu Y.(P \vee ((\mu Z.(X \vee \langle A \rangle Z)) \wedge \langle B \rangle Y)) \quad (2)$$

Let X_n be the set of states fulfilling this formula. The set of states labeled with this formula by the model checking algorithm (Algorithm 1) is then:

$$\nu Y.(P \vee (K \wedge \langle B \rangle Y)) \quad \text{where} \quad K = \mu Z.(X_n \vee \langle A \rangle Z) \quad (3)$$

Figure 1 shows what kind of states ψ fulfill this formula. A state ψ fulfills this formula $\nu Y.(P \vee (K \wedge \langle B \rangle Y))$ if there is a main path with only B-transitions where at each state there is a side path according to K, i.e., a sidepath with only A-transitions which ends in a state, e.g., δ where again a structure similar to the one starting at ψ begins. The main path can either end at a state γ where P holds or end up in a loop.

4.2 Main path witnesses

The main purpose of constructing witnesses when model checking is to allow the user of a verification tool to find errors in the model. Huge witnesses, however, are difficult to understand. The witnesses constructed by the algorithm given in [Kic95] can still be huge although they were already defined as reduced tableaux. In this subsection we therefore consider the reduction of witnesses.

It does not make sense to return the whole bushy tree in Figure 1 as a witness for Formula 2. Since there are a vast amount of side paths (along $\langle A \rangle$-arcs), we should refrain from constructing these. Indeed, the user of a model checker only

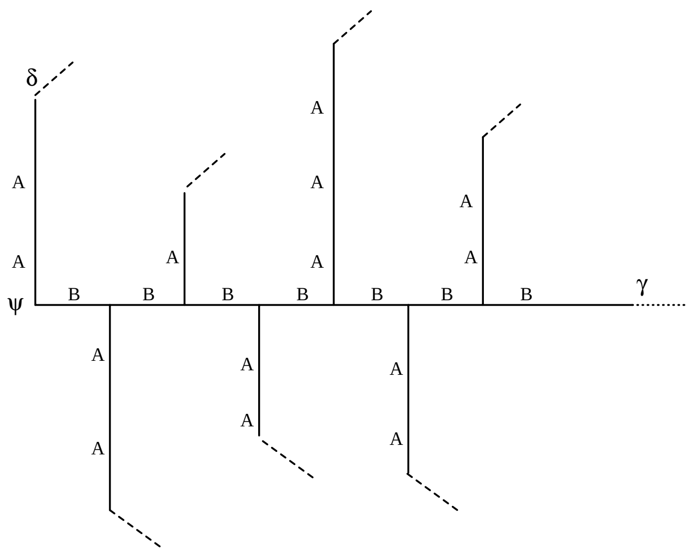

Figure 1: Witnesses: main path and side paths

wants to see the main path - the path along the B-arcs in Figure 1 - and not all side conditions the states on the path have to fulfill. We therefore suggest returning just the main path as a witness, e.g., the witness for the much simpler Formula 3, treating K as a propositional constant, as a witness for Formula 2.

Definition 5 (Main paths) *The main paths of a formula f ($M(f)$) are defined as*

$$M(\sigma X.p(X)) = \begin{cases} \{\sigma X.p(X)\} & \mathcal{I}(\sigma X.p(X)) = 1 \\ M(p(X)) & otherwise \end{cases}$$

$$M(p \wedge q) = M(p) \cup M(q) \qquad M(p \vee q) = M(p) \cup M(q)$$

$$M(\langle\rangle p) = M(p) \qquad M([\,]p) = M(p)$$

$$M(P) = \emptyset \qquad M(Y) = \emptyset \qquad M(\neg P) = \emptyset$$

$$VM(f) = \{X | \sigma X.p(X) \in M(f)\}$$

The motivation behind this definition is that the fixpoint iterations of μ- and ν-expressions with iteration depth 1 are the last ones when model checking a formula f.

Even if the side paths are not developed in the witness, the witness for the main paths can still be huge. Therefore, within $p(X)$ of a main path $\sigma X.p(X)$ we develop only one conjunct q or r as a witness to demonstrate $q \wedge r$, and the witness for one successor to demonstrate $[\,]q$. We also consider all other propositional variables $Y \notin VM(f)$ as propositional constants.

Consider, e.g., the CTL formula $AFEFf$. It makes sense to construct just a linear path to a state where EFf holds instead of constructing a huge witness for AF.

Definition 6 (Reduced witness path for $\sigma Y.q(Y)$) *A reduced witness path for $\sigma Y.q(Y)$ is defined in the same way as the abstract witness except that (a),(d),(e) and (f) are replaced by*

(a) $p \wedge q \in m(s)$ *implies* $p \in m(s)$ *or* $q \in m(s)$

(d) $[\ ]p$ *implies: for an arbitrary* $s' \in \{s'' | sRs''\} : s' \in V$, $p \in m(s')$, $(s, s') \in E$

(e) $\sigma Y.q(Y) \in m(s)$ *implies* $p(Y) \in m(s)$

(f) $Y \in m(s)$ *implies* $\sigma Y.q(Y) \in m(s)$

In (e) and (f) the Y is not an arbitrary propositional variable but the fixed Y in $\sigma Y.q(Y)$.

Definition 7 (Main path witness) *A main path witness is defined in the same way as the concrete witness except that $\forall \mu X.p(X)$ is replaced by $\forall \mu X.p(X) \in M(f)$ in condition (1), (f) is deleted and (e) is replaced by*

(e1) $\sigma X.p(x) \in m(s) \wedge \sigma X.p(X) \in M(f)$ *implies: let* (V_X, E_X, m_X) *be a reduced witness path for* $\sigma X.p(X)$, *then* $V_X \subseteq V, E_X \subseteq E, \forall s \in S \colon m_X(s) \subseteq m(s)$

(e2) $\sigma X.p(x) \in m(s) \wedge \sigma X.p(X) \notin M(f)$ *implies:* $p(X) \in m(s)$

Witnesses should also be as small as possible since small witnesses make it easier to find an error in a design.

Definition 8 (Minimal witness) *A main path witness $W = (V,E,m)$ for $s \models_M f$, $M = (S, R, L)$, $s \in S$, $f \in L_\mu$ is minimal if there is no other main path witness $W' = (V',E',m')$ with $|E'| < |E|$.*

5 Constructing main path witnesses

The following three definitions help us in constructing minimal main path witnesses.

Definition 9 (Possible Xpath) *π is a possible Xpath for a formula f and variable X if $pXp(\pi, f, X)$. $pXp(\pi, f, X)$ is defined in the same way as $Xp(\pi, f, X)$ except that $f \in m(s)$ is always replaced by $l(s, f)$.*

Definition 10 (Possible σ-path) *A sequence of states $\pi = s_0 \dots s_m$ is called a possible σ-path for $\sigma X.p(X)$ if $psp(\pi, \sigma X.p(X))$ where*

$$psp(s_0 \dots s_m, \sigma X.p(X)) \Leftrightarrow \forall 0 \le i \le m \colon l(s_i, \mu X.p(X)) \wedge \forall 0 \le i < m \colon \\ \exists \rho = \rho_0 \dots \rho_n \colon pXp(\rho, p(X), X) \wedge \rho_0 = s_i \wedge \rho_n = s_{i+1}$$

It is straightforward to see that an appropriate labeling m of the states in one of the possible σ-paths for $\sigma X.p(X)$ is a reduced witness path for $\sigma X.p(X)$. To find the shortest σ-path for $\sigma X.p(X)$ we have to know the length of the possible Xpaths.

Definition 11 (Distance of a possible Xpath) $T_X\colon S \times S \to \mathbb{N}_0$ *denotes the distance of a possible Xpath for* $\sigma X.p(X)$ *and variable* X*.*

$$T_X(s,s') = \begin{cases} d & (s,d,s') \in U(p(X),X) \\ \infty & otherwise \end{cases}$$

where

$$U\colon L_\mu \times PropVar \to S \times \mathbb{N}_0 \times S$$

$$U(Y,X) = \begin{cases} \{(s,0,s) | l(s,X) = true\} & Y \equiv X \\ \emptyset & otherwise \end{cases}$$

$$U(P,X) = \emptyset \qquad U(\neg P,X) = \emptyset \qquad U(\sigma Y.p,X) = \emptyset$$

$$U(p \wedge q,X) = \{(s,d,s') | l(s,p \wedge q) \wedge (s,d,s') \in U(p,X) \cup U(q,X)\}$$

$$U(p \vee q,X) = \{(s,d,s') | l(s,p) \wedge (s,d,s') \in U(p,X) \vee l(s,q) \wedge (s,d,s') \in U(q,X)\}$$

$$U(\Box p,X) = \{(s,d+1,s') | l(s,\Box p) \wedge \exists s''\colon sRs'' \wedge (s'',d,s') \in U(p,X)\}$$

This distance T_X is used in the following algorithm to compute the shortest reduced witness paths for $\sigma X.p(X) \in M(f)$. These have the form of a linear path ending up in $p(false)$ in the case of μ and ending up in $p(false)$ or a loop in the case of ν. $U_{X,1}[s]$ is the minimal distance of the σ-paths for $\sigma X.p(X)$ of this form from a state s. The minimal main path witness can be constructed by computing the minimal distance of a main path witness for all subformulae and for all states $s \in S$, using $U_{X,1}[s]$. Information saved during this process and $U_{X,2}[t]$, the final state t' (which is either the end state of the σ-path or the state from which a loop starts) of a shortest σ-path from a state t, are later used for the actual construction of the minimal main path witness for f.

Definition 12 *For an original formula* f *we define for* $p \preceq f$*:*

$$ml(P) = 1 \qquad ml(\neg P) = 1 \qquad ml(X) = 1$$

$$ml(p \wedge q) = ml(p) + ml(q) + 1 \qquad ml(p \vee q) = ml(p) + ml(q) + 1$$

$$ml(\Box p) = ml(p) + 1$$

$$ml(\sigma X.p) = \begin{cases} 1 & \sigma X.p \in M(f) \\ ml(p) + 1 & otherwise \end{cases}$$

Definition 13 *For a formula $\sigma X.p$ we define for $q \preceq \sigma X.p$ (in the following $Y \not\equiv X$):*

$$pl(P) = 1 \qquad pl(\neg P) = 1 \qquad pl(X) = 1$$

$$pl(Y) = 1 \qquad pl(\sigma Y.r) = 1$$

$$pl(p \wedge q) = pl(p) + pl(q) + 1 \qquad pl(p \vee q) = pl(p) + pl(q) + 1$$

$$pl(\Box p) = pl(p) + 1$$

Algorithm 2
input: f, M(f), M = (S,R,L), from model checking: l(s,p) for all states s and subformulae p of f
output: W = (V,E,m) minimal main path witness

```
/* computes a matrix $T_X$ with row and column elements in $S$ where $T_X[s, s']$
denotes the distance of a path according to $p(X)$ between the states $s, s'$
with $s \models \sigma X.p(X)$ and $s' \models \sigma X.p(X)$ */
procedure ctx($\sigma X.p : L_\mu$)
begin
  for $i = 1$ to $pl(u)$ do
    for all $r \preceq u \wedge pl(r) = i$ do
      for all $s \in S$ do
        if $l(s, r)$ then
          case $r$ of the form
            $X : e_X(s, r) := \{(s, 0, s)\}$
            $P, \neg P, \sigma Y.q, Y(\not\equiv X) : e_X(s, r) := \emptyset$
            $p \wedge q, p \vee q : e_X(s, r) := e_X(s, p) \cup e_X(s, q)$
            $\Box p : e_X(s, r) := \{(s, d + 1, s') | l(s, \Box p) \wedge \exists s'' : (s, s'') \in R$
                                $\wedge (s'', d, s') \in e_X(s, p)\}$
          esac
  for all $s, s' \in S : T_X(s, s') := \infty$;
  for all $s \in S$ for all $(s, d, s') \in e_X(s, \sigma X.p) : T_X(s, s') := d$;
end

/* finds the shortest witness for a formula $u$
the following information for the later construction is saved:
d(s,r): the length of the shortest witness starting at s
n(s,r): the next state of s on the shortest witness starting at s
min(s,r): the subformula for which the witness is shortest */
procedure find-shortest-witness($u$:$L_\mu$)
begin
  for $i = 1$ to $ml(u)$ do
    for all $r \preceq u \wedge ml(r) = i$ do
      for all $s \in S$ do
```

```
            if $l(s,r)$ then
               case $r$ of the form
                  $X$ : $d(s,r) := 0$;
                  $P, \neg P$ : $d(s,r) := 0$;
                  $p \wedge q$ : $d(s,r) := d(s,p) + d(s,q)$;
                  $p \vee q$ : if $l(s,p) \wedge l(s,q)$ then $d(s,r) := min\{d(s,p), d(s,q)\}$
                          else if $l(s,p)$ then $d(s,r) := d(s,p)$
                          else $d(s,r) := d(s,q)$;
                          if $d(s,r) = d(s,p)$ then $min(s,r) := p$ else $min(s,r) := q$;
                  $\langle\rangle p$ : $d(s,r) := min\{d(s',p) | sRs' \wedge l(s',p)\} + 1$
                        $n(s,r) := s'$ where $d(s',p) = d(s,r) \Leftrightarrow 1$
                  $[\,]p$ : $d(s,r) := \sum_{(s,s') \in R} (d(s',p) + 1)$
                  $\sigma X.p(X)$ : if $\sigma X.p(X) \in M(f)$ then $d(s,r) := U_{X,1}[s]$
                          else $d(s,r) := d(p(X))$
               esac
end

procedure $is(s{:}S, s'{:}S, p{:}L_\mu, X{:}PropVar)$
begin
   $m(s) := m(s) \cup \{p\}$;
   $(s,d,s') := (s, min\{d' | (s,d',s') \in e_X(s,p)\}, s')$
   case $p$ of the form
      $X$ : return;
      $p \wedge q, p \vee q$ : if $(s,d,s') \in e_X(s,p)$ then $is(s,s',p,X)$ else $is(s,s',q,X)$;
      $\Box p$ : choose $s''$ with $(s,s'') \in R \wedge (s'', d \Leftrightarrow 1, s') \in e_X(s'',p)$;
          $V := V \cup \{s''\}$; $E := E \cup \{(s,s'')\}$; $is(s'',s',p,X)$;
   esac
end

procedure construct-$\mu$path$(s{:}S, s'{:}S, p{:}L_\mu, X{:}PropVar)$
begin
   repeat
      choose $t \in S$ such that $T_X[s,t] + A_X[t,s'] = A_X[s,s']$;
      $is(s,t,p,X)$;
      $s := t$;
   until $s = s'$
end

procedure construct-$\nu$path$(s{:}S, s'{:}S, p{:}L_\mu, X{:}PropVar)$
begin
   $\mu path(s,s',p,X)$;
   $\mu path(s',s',p,X)$;
```

```
end

/* constructs the shortest path for $\sigma X.p(X)$ from s
along states $s'' \models \sigma X.p(X)$ */
procedure construct-mpath(s:S,$\sigma X.p : L_\mu$)
begin
  if $U_{X,1}[s] \neq 0$ then
    if $\sigma = \mu$ then construct-$\mu$path(s,$U_{X,2}[s]$,p,X)
    else construct-$\nu$path(s,$U_{X,2}[s]$,p,X);
end

procedure csw($s : S, f : L_\mu$)
begin
  $m(s) := m(s) \cup \{f\}$;
  case f of the form
    X : return;
    $P, \neg P$ : return;
    $p \wedge q$ : csw(s,p); csw(s,q);
    $p \vee q$ : csw(s,min(f));
    $\langle\rangle p$ : $V := V \cup \{n(s)\}; E := E \cup \{(s, n(s))\}; csw(n(s), p)$;
    $[\,]p$ : for all $s' \in sR$ do begin $V := V \cup \{s'\}; E := E \cup \{(s, s')\}; csw(s', p)$; end;
    $\sigma X.p(X)$ : if $\sigma X.p(X) \in M(f)$ then construct-mpath(s,$\sigma X.p(X)$) else csw(s,$p(X)$);
  esac
end


for all $\sigma X.p(X) \in M(f)$ do begin
  ctx($\sigma X.p(X)$);
  /* $A_X[s, s']$ is the distance of the shortest $\sigma$-path for $\sigma X.p(X)$ between s and s' */
  $A_X$ := all-pairs-shortest-path($T_X$);
  if $\sigma = \mu$ then
    for all $t \in S$ do
      begin
        $U_{X,1}[t] := min\{A_X[t, t'] | t' \in S \wedge l(t', p(false))\}$
        $U_{X,2}[t] := t'$ where $U_1[t] = A_X[t, t'] \wedge l(t', p(false))$
      end;
  if $\sigma = \nu$ then
    for all $t \in S$ do
      begin
        $U_{X,1}[t] := min\{\{A_X[t, t'] + A_X[t', t'] | t' \in S\} \cup \{A_X[t, t'] | t' \in S \wedge l(t', p(false))\}\}$
        $U_{X,2}[t] := t'$ where $U_1[t] = A_X[t, t'] \wedge l(t', p(false))$
                      or $U_{X,1}[t] = A_X[t, t'] + A_X[t', t']$
      end;
```

```
end;
find-shortest-witness(f);
let s such that $d(s,f) = min\{d(s',f) | s' \in S\}$
$V := \{s\}; E := \emptyset;$ for all $s \in S$ do $m(s) := \emptyset$
csw(s,f);
```

Theorem 2 *Algorithm 2 constructs a minimal main path witness for $s \models f$.*

Proof: The proof is straightforward but tedious. Some inductions on the length of the subformulae of f have to be performed.

■

Theorem 3 *Algorithm 2 has time complexity $O(|M(f)| \cdot (|S|^3 + |S| \cdot |M| \cdot |f|))$.*

Proof: The following table shows the time complexity of the different parts of the algorithm:

$ctx(\sigma X.p(X))$	$O(\|M\| \cdot \|p\|)$
all-pairs-shortest-path(T_X)	$O(\|S\|^3)$
computing $U_{X,i}$	$O(\|S\|^2)$
find-shortest-witness(f)	$O(\|M\| \cdot \|f\|)$
is(s,s',p,X)	$O(\|M\| \cdot \|p\|)$
construct-mpath(s,$\sigma X.p(X)$)	$O(\|S\| \cdot (\|S\| + \|M\| \cdot \|p\|))$
csw(s,f)	$O(\|M\| \cdot \|f\| + \|M(f)\| \cdot (\|S\| \cdot (\|S\| + \|M\| \cdot \|f\|)))$

As a consequence, the total complexity of the algorithm is:

$$O(|M(f)| \cdot (|M| \cdot |f| + |S|^3 + |S|^2) + |S| + |M| \cdot |f| + |M(f)| \cdot (|S| \cdot (|S| + |M| \cdot |f|))) =$$

$$O(|M(f)| \cdot (|S|^3 + |S| \cdot |M| \cdot |f|))$$

■

6 Conclusion and comparision

We have shown how to construct reduced counterexamples and reduced witnesses for the whole μ-calculus. This eliminates the most important disadvantage of μ-calculus model checkers and allows a much more general approach to model checking than usual CTL model checkers.

The construction of minimal main path witnesses for μ-calculus expressions is polynomial in $|f|$ and $|M|$ in contrast to model checking f which is exponential in the alternation depth of f. As a consequence, the construction of witnesses is only a minor factor in the verification of reactive systems.

FCTL is a subclass of L_{μ_2}. The counterexamples for FCTL in [CGMZ94] are for the special type of L_{μ_2} formulae of the form $\nu Z.[f \wedge \bigwedge_k \langle\rangle [\mu X.Z \wedge h_k \vee (f \wedge \langle\rangle X)]]$.

The algorithm in [CGMZ94] would construct a single path with a cycle in which all fairness constraints h_k are contained in contrast to Algorithm 2 which would construct a path for each separate conjunct $\langle\rangle[\mu X.Z \wedge h_k \vee (f \wedge \langle\rangle X)]]$ where the final state fulfills Z. In the construction of the witness in [CGMZ94] the special meaning of the FCTL formula is exploited. Therefore, their counterexample construction does not extend to the whole μ-calculus.

References

[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.

[CGMZ94] E. Clarke, O. Grumberg, K. McMillan, and X. Zhao. Efficient generation of counterexamples and witnesses in symbolic model checking. Technical Report CMU-CS-94-204, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, October 1994.

[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.

[Kic95] A. Kick. Tableaux and witnesses for the μ-calculus. Technical Report 44/95, Faculty of Computer Science, University of Karlsruhe, D-76128 Karlsruhe, Germany, October 1995.

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

[McM93] K. L. McMillan. *Symbolic Model Checking.* Kluwer Academic Publishers, Boston,USA, 1993.