Available online at www.sciencedirect.com

SCIENCE DIRECT®

Information Processing Letters 97 (2006) 225–230

Information Processing Letters

www.elsevier.com/locate/ipl

Mu-calculus path checking

Nicolas Markey *, Philippe Schnoebelen

Lab. Spécification & Vérification, CNRS & ENS de Cachan, France

Received 11 August 2005; received in revised form 31 October 2005

Available online 19 December 2005

Communicated by D. Basin

Abstract

We investigate the path model checking problem for the μ-calculus. Surprisingly, restricting to deterministic structures does not allow for more efficient model checking algorithm, as we prove that it can encode any instance of the standard model checking problem for the μ-calculus.

Keywords: Program correctness; Formal methods; Mu-calculus; Model checking

1. Introduction

Model checking is a fundamental problem, originally motivated by concerns with the automatic verification of systems, but now more broadly associated with several different fields ranging from Bio-Informatics to Databases to Automated Deduction. In verification settings, model checking problems usually ask whether S, a given model of a system, satisfies ϕ, a given formal property, denoted "$S \models \phi$". In [8] we introduced the *path model checking* problem (see also Open Problem 4.1 in [4]). This problem is unusual since it is a *restriction* of the classical model checking problem, not an extension as is usually considered. The restriction is that one only considers models having the form of a *finite path* (or a finite loop, or more generally an ultimately periodic infinite path). These are models *without choice*, or *without nondeterminism*. Checking finite paths or loops occurs naturally in many applications: run-time verification [5], analysis of machine-generated scenarios or debugger traces [1], analysis of log files [11], Monte Carlo methods for verification [6], etc.

In [8] we consider path model checking for several temporal logics. Our findings can be summarized as follows:

- checking a deterministic path is usually much easier than checking a nondeterministic structure,
- checking a finite path and checking a loop are usually equivalent (inter-reducible).

In this note, we consider path model checking for the modal μ-calculus. It is known that checking whether a Kripke structure S satisfies a μ-calculus formula (called the *branching-time*, or B_μ, model-checking problem) is PTIME-hard, and is in UP $\cap$ coUP [7]. Additionally, checking whether all paths of S satisfy a μ-calculus formula (called the *linear-time*, or L_μ, model-checking problem) is PSPACE-complete [12].

* Corresponding author.

E-mail addresses: markey@lsv.ens-cachan.fr (N. Markey), phs@lsv.ens-cachan.fr (Ph. Schnoebelen).

0020-0190/$ – see front matter
doi:10.1016/j.ipl.2005.11.010

For path model checking, our findings are surprising:

1. General B_μ model checking reduces to path model checking. Hence B_μ model checking does not become easier when it is restricted to structures without choice. This does not fit the pattern observed in [8] for other logics like CTL or CTL*.
2. The above reduction uses loops. We were not able to reduce checking of finite loops to checking of finite paths. Again this does not fit the pattern observed in [8] for other logics.

The paper contains some additional results, e.g., that model checking of finite paths is PTIME-complete (hence the above discrepancies would disappear if it turns out that μ-calculus model checking is in PTIME, a conjecture believed true by several researchers), or relating loops and finite paths in a μ-calculus extended with backwards (sometimes called "past-time") modalities.

2. Preliminaries

We refer to [3]. μ-calculus formulae are given by the following grammar:

$$B_\mu \ni \varphi, \psi ::= p \mid \neg p \mid Z \mid \varphi \wedge \psi \mid \varphi \vee \psi \\ \mid \Diamond\varphi \mid \Box\varphi \mid \mu Z.\varphi \mid \nu Z.\varphi,$$

where p ranges over a set AP of *atomic propositions*, and Z over a set $\mathcal{V}$ of *variable names*. Our definition only allows negations on propositions, but negation of arbitrary formulae can be defined in the standard way, and similarly for classical shorthands such as $\Rightarrow$, etc. We define the CTL-modalities EF and AG with:

$$\mathrm{EF}\,\varphi \stackrel{\text{def}}{=} \mu Z.(\varphi \vee \Diamond Z) \quad \text{and} \quad \mathrm{AG}\,\varphi \stackrel{\text{def}}{=} \nu Z.(\varphi \wedge \Box Z),$$

where Z is any variable not free in φ.

Formulae in B_μ are interpreted over finite Kripke structures (KS), i.e., labeled finite-state systems of the general form $K = (Q, R, l)$ where $R \subseteq Q \times Q$ is the set of transitions and $l : Q \to 2^{\mathrm{AP}}$ is the state labeling. As usual, and when R is understood, we write $x \to y$ rather than $(x, y) \in R$, and we say y is *a successor* of x. Given $S \subseteq Q$, we write $\mathrm{Pre}(S)$ for the set $\{x \in Q \mid \exists y \in S.\ x \to y\}$, and $\overline{S}$ for $Q \setminus S$. Then $x \in \overline{\mathrm{Pre}(\overline{S})}$ iff all the successors of x (if any) are in S.

Formally, for a KS $K = (Q, R, l)$ and a context $v : \mathcal{V} \to 2^Q$, the set $[\![\varphi]\!]_v^K$ of states where φ holds is defined inductively:

$$\begin{aligned}
[\![p]\!]_v^K &\stackrel{\text{def}}{=} \{x \in Q \mid p \in l(x)\},\\
[\![\neg p]\!]_v^K &\stackrel{\text{def}}{=} \{x \in Q \mid p \notin l(x)\},\\
[\![\varphi \vee \psi]\!]_v^K &\stackrel{\text{def}}{=} [\![\varphi]\!]_v^K \cup [\![\psi]\!]_v^K,\\
[\![\varphi \wedge \psi]\!]_v^K &\stackrel{\text{def}}{=} [\![\varphi]\!]_v^K \cap [\![\psi]\!]_v^K,\\
[\![Z]\!]_v^K &\stackrel{\text{def}}{=} v(Z),\\
[\![\Diamond\varphi]\!]_v^K &\stackrel{\text{def}}{=} \mathrm{Pre}([\![\varphi]\!]_v^K),\\
[\![\Box\varphi]\!]_v^K &\stackrel{\text{def}}{=} \overline{\mathrm{Pre}(\overline{[\![\varphi]\!]_v^K})},\\
[\![\mu Z.\varphi]\!]_v^K &\stackrel{\text{def}}{=} \bigcap\{U \subseteq Q \mid [\![\varphi]\!]_{v\,[Z\mapsto U]}^K \subseteq U\},\\
[\![\nu Z.\varphi]\!]_v^K &\stackrel{\text{def}}{=} \bigcup\{U \subseteq Q \mid U \subseteq [\![\varphi]\!]_{v\,[Z\mapsto U]}^K\}.
\end{aligned}$$

We sometimes omit the "K" and "v" subscripts when no ambiguity arises (or for closed formulae where "v" is irrelevant) and write $x \models_v^K \varphi$ when $x \in [\![\varphi]\!]_v^K$. The above definition entails the following standard *fixed-point equalities*:

$$[\![\mu Z.\varphi]\!]_v = [\![\varphi]\!]_{v[Z\mapsto [\![\mu Z.\varphi]\!]_v]},$$
$$[\![\nu Z.\varphi]\!]_v = [\![\varphi]\!]_{v[Z\mapsto [\![\nu Z.\varphi]\!]_v]}.$$

For $\alpha \in \mathbb{N}$, the *approximant* $[\![\mu Z^\alpha.\varphi]\!]_v^K$ is defined inductively by

$$[\![\mu Z^0.\varphi]\!]_v \stackrel{\text{def}}{=} \emptyset \quad \text{and}$$
$$[\![\mu Z^{\alpha+1}.\varphi]\!]_v \stackrel{\text{def}}{=} [\![\varphi]\!]_{v[Z\mapsto [\![\mu Z^\alpha.\varphi]\!]_v]}.$$

Set $[\![\nu Z^\alpha.\varphi]\!]_v$ is defined dually. It is well known that, since K is finite, the sequences $([\![\mu Z^\alpha.\varphi]\!]_v)_{\alpha\in\mathbb{N}}$ and $([\![\nu Z^\alpha.\varphi]\!]_v)_{\alpha\in\mathbb{N}}$ eventually reach $[\![\mu Z.\varphi]\!]_v$ and $[\![\nu Z.\varphi]\!]_v$, respectively.

A KS is *deterministic* if every state has at most one successor. For such KS's, $\Diamond\varphi$ and $\Box\varphi$ have very close meanings: $\Diamond\varphi$ means that φ holds in the successor state, while $\Box\varphi$ means that, *if* there is a successor state, *then* φ holds in that state. We consider below deterministic KS's having the form of a finite *path* (isomorphic to an initial segment of $\mathbb{N}$, with a last state having no successors), or a finite *loop* (where there is a single strongly connected component). On loops, the meanings of $\Diamond\varphi$ and $\Box\varphi$ coincide exactly.

3. Main result

Theorem 3.1. *B_μ model checking logspace-reduces to model checking of loops.*

Hence μ-calculus model checking of loops and general B_μ model checking are equivalent (inter-reducible).

Considering deterministic KS's does not simplify the problem:

Corollary 3.2. *B_μ model checking of loops is* PTIME-*hard, and in* UP ∩ coUP.

The rest of this section describes our reduction. We transform an instance "$x \models^K \varphi$?" into an equivalent "$x' \models^L \tilde{\varphi}$?" where L is a loop. We observe that $|L| = \mathrm{O}(|K|)$, and $|\tilde{\varphi}| = \mathrm{O}(|K| \cdot |\varphi|)$. Furthermore, the transformation from φ to $\tilde{\varphi}$ does not increase the alternation depth (Proposition 3.8).

Let $K = (Q, R, l)$ be a KS. For this reduction we assume that AP and Q coincide, and that l is the identity.[1] L has labels from $\mathrm{AP}' \stackrel{\text{def}}{=} \mathrm{AP} \cup \{\mathbf{s}, \mathbf{d}\}$ where **s** (for *source*) and **d** (for *destination*) are two new atomic propositions. Assume $R = \{r_1, \ldots, r_n\}$ contains n transitions: then $L = (Q', R', l')$ has $Q' \stackrel{\text{def}}{=} \{s_1, d_1, s_2, d_2, \ldots, s_n, d_n\}$. R' has transitions $s_i \to d_i$ and $d_i \to s_{(i \bmod n)+1}$ for $1 \leqslant i \leqslant n$, arranging Q' into a loop. Finally, the labeling l' is defined as follows: if $r_i = (x, y)$ then $l'(s_i) = \{x, \mathbf{s}\}$ and $l'(d_i) = \{y, \mathbf{d}\}$.

In summary, L lists the transitions of K. The states of L maps to original states via the mapping $h : Q' \to Q$ given by $h(x') = x \Leftrightarrow x \in l'(x')$. Fig. 1 illustrates this construction on a schematic example.

In the sequel we use $h(x')$ either as a state or as an element of AP', depending on the context. For any $S \subseteq Q$, $h(x') \in S$ iff $x' \in h^{-1}(S)$.

Lemma 3.3. *Let $S \subseteq Q$. Then* $\mathrm{Pre}_K(S) = h([\![\mathbf{s}]\!]^L \cap \mathrm{Pre}_L(h^{-1}(S)))$.

Proof. Assume $x \in \mathrm{Pre}_K(S)$ because of a transition r_i of the form $x \to y$ with $y \in S$. In L, $s_i \to d_i$ has $d_i \in h^{-1}(y) \subseteq h^{-1}(S)$ and $s_i \in [\![\mathbf{s}]\!]^L$. Hence $x = h(s_i) \in h([\![\mathbf{s}]\!]^L \cap \mathrm{Pre}_L(h^{-1}(S)))$. Conversely, if $x \in h([\![\mathbf{s}]\!]^L \cap \mathrm{Pre}_L(h^{-1}(S)))$, then $x = h(s_i)$ for some i such that $h(d_i) \in S$. Therefore r_i shows that $x \in \mathrm{Pre}_K(S)$. □

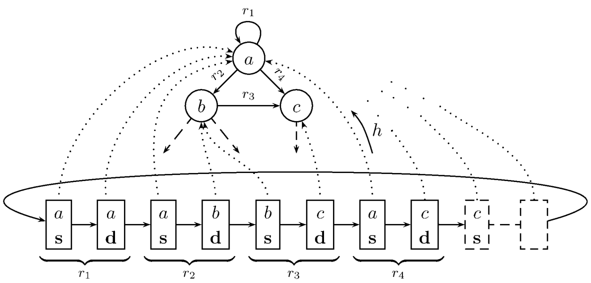

Fig. 1. From non-deterministic to deterministic Kripke structure.

Now, define

$$\Theta(Z) \stackrel{\text{def}}{=} \bigvee_{x \in Q} \big[x \wedge \mathrm{EF}(x \wedge Z)\big]$$

and

$$\Xi(Z) \stackrel{\text{def}}{=} \bigwedge_{x \in Q} \big[x \Rightarrow \mathrm{AG}(x \Rightarrow Z)\big].$$

Lemma 3.4. *For all v, $[\![\Theta(Z)]\!]^L_v = h^{-1}(h([\![Z]\!]^L_v))$ and $[\![\Xi(Z)]\!]^L_v = \overline{h^{-1}(h\overline{[\![Z]\!]^L_v})}$.*

Proof. $[\![\Theta(Z)]\!]_v$ is $\bigcup_{x \in Q} [\![x \wedge \mathrm{EF}(x \wedge Z)]\!]_v$. Since L is strongly connected, this is $\{x' \mid \exists y' \in [\![Z]\!]_v, h(x') = h(y')\}$ by definition of l'. We end up with $h^{-1}(h([\![Z]\!]_v))$. The second result follows by duality. □

Lemma 3.5. *Assume Y and Z are distinct variables. Then for all v, we have*

$$[\![\mu Z.\big(Y \vee \Theta(Z)\big)]\!]^L_v = \Theta(Y) = h^{-1}\big(h\big([\![Y]\!]^L_v\big)\big),$$
$$[\![\mu Z.\big(Y \vee \Xi(Z)\big)]\!]^L_v = \Xi(Y) = \overline{h^{-1}\big(h\big(\overline{[\![Y]\!]^L_v}\big)\big)}.$$

Proof. We only prove the first result, the second one being dual.

(⊆) Write U for $h^{-1}(h([\![Y]\!]_v))$. Then

$$\begin{aligned}[\![Y \vee \Theta(Z)]\!]_{v[Z \mapsto U]} &= [\![Y]\!]_v \cup [\![\Theta(Z)]\!]_{v[Z \mapsto U]} \\ &= [\![Y]\!]_v \cup h^{-1}\big(h(U)\big) \\ &\qquad \text{(by Lemma 3.4)} \\ &= U.\end{aligned}$$

Hence U is a fixed point and $[\![\mu Z.(Y \vee \Theta(Z))]\!]_v \subseteq U$.

(⊇) Write S for $[\![\mu Z.(Y \vee \Theta(Z))]\!]_v$. From the fixed-point property, we have

$$\begin{aligned}S &= [\![Y \vee \Theta(Z)]\!]_{v[Z \mapsto S]} = [\![Y]\!]_v \cup [\![\Theta(S)]\!]_v \\ &= [\![Y]\!]_v \cup h^{-1}\big(h(S)\big) \quad \text{(by Lemma 3.4).}\end{aligned}$$

Hence $S \supseteq h^1(h([\![Y]\!]_v))$. □

Thus $\Theta(\psi)$ and $\mu Z.(\psi \vee \Theta(Z))$ are equivalent on L (when Z does not occur free in ψ). The important difference between them is size: $|\Theta(\psi)|$ is in $\mathrm{O}(|Q| \cdot |\psi|)$ while $|\mu Z.(\psi \vee \Theta(Z))|$ is in $\mathrm{O}(|Q| + |\psi|)$.

[1] This assumption is no loss of generality. Any general KS can be relabeled in such a way. This requires replacing any proposition used in the original labeling with a disjuction of (the propositions denoting) the states where it holds. This transformation is logspace.

We now translate each formula φ into a $\tilde{\varphi}$ in such a way that if φ holds in $x \in Q$, then $\tilde{\varphi}$ holds in all $x' \in h^{-1}(x)$. Formally, $\tilde{\varphi}$ is defined inductively by:

$$\begin{aligned}
\tilde{p} &\stackrel{\text{def}}{=} p,\\
\widetilde{\neg p} &\stackrel{\text{def}}{=} \neg p,\\
\widetilde{\varphi \vee \psi} &\stackrel{\text{def}}{=} \tilde{\varphi} \vee \tilde{\psi},\\
\widetilde{\varphi \wedge \psi} &\stackrel{\text{def}}{=} \tilde{\varphi} \wedge \tilde{\psi},\\
\tilde{Z} &\stackrel{\text{def}}{=} Z,\\
\widetilde{\Diamond\varphi} &\stackrel{\text{def}}{=} \mu Z\big[(\mathbf{s} \wedge \Diamond\tilde{\varphi}) \vee \Theta(Z)\big],\\
\widetilde{\Box\varphi} &\stackrel{\text{def}}{=} \nu Z.\big[(\mathbf{s} \Rightarrow \Box\tilde{\varphi}) \wedge \Xi(Z)\big],\\
\widetilde{\mu Z.\varphi} &\stackrel{\text{def}}{=} \mu Z.\tilde{\varphi},\\
\widetilde{\nu Z.\varphi} &\stackrel{\text{def}}{=} \nu Z.\tilde{\varphi}.
\end{aligned}$$

Lemma 3.6. *For any formula φ involving atomic propositions in* AP, *and any context $v : \mathcal{V} \to 2^Q$, and writing v' for $h^{-1} \circ v$:*

$$h^{-1}\big(\llbracket\varphi\rrbracket_v^K\big) = \llbracket\tilde{\varphi}\rrbracket_{v'}^L. \tag{1}$$

In other words, $x' \in \llbracket\tilde{\varphi}\rrbracket_{v'}^L$ iff $h(x') \in \llbracket\varphi\rrbracket_v^K$.

Proof. By induction on the structure of φ.

Case $\varphi = p \in \mathrm{AP}$: Since $\mathrm{AP} = Q$, and by definition of l', $h^{-1}(\llbracket p\rrbracket^K) = \llbracket p\rrbracket^L$.

Case $\varphi = Z \in \mathcal{V}$: $h^{-1}(\llbracket Z\rrbracket_v) = h^{-1} \circ v(Z) = \llbracket Z\rrbracket_{v'}$ by definition of v'.

Case $\varphi = \mu Z.\psi$: It is sufficient to show that, for all integers α, $h^{-1}(\llbracket\mu Z^\alpha.\psi\rrbracket_v) = \llbracket\mu Z^\alpha.\tilde{\psi}\rrbracket_{v'}$. We proceed by induction on α. The base case where $\alpha = 0$ holds trivially, and the inductive step relies on

$$\begin{aligned}
h^{-1}\big(\llbracket\mu Z^{\alpha+1}.\psi\rrbracket_v\big) &= h^{-1}\big(\llbracket\psi\rrbracket_{v[Z\mapsto\llbracket\mu Z^\alpha.\psi\rrbracket_v]}\big)\\
&= \llbracket\tilde{\psi}\rrbracket_{h^{-1}\circ v[Z\mapsto\llbracket\mu Z^\alpha.\psi\rrbracket_v]}\\
&\qquad\text{by ind. hyp. (Lemma 3.6 on } \psi).
\end{aligned}$$

This is $\llbracket\tilde{\psi}\rrbracket_{v'[Z\mapsto h^{-1}(\llbracket\mu Z^\alpha.\psi\rrbracket_v)]} = \llbracket\tilde{\psi}\rrbracket_{v'[Z\mapsto\llbracket\mu Z^\alpha.\tilde{\psi}\rrbracket_{v'}]}$ (by ind. hyp. on α), hence equals $\llbracket\mu Z^{\alpha+1}.\tilde{\psi}\rrbracket_{v'}$.

Case $\varphi = \Diamond\psi$:

$$\begin{aligned}
h^{-1}\big(\llbracket\Diamond\psi\rrbracket_v\big) &= h^{-1}\big(\mathrm{Pre}\big(\llbracket\psi\rrbracket_v\big)\big)\\
&= h^{-1}\big(h\big(\llbracket\mathbf{s}\rrbracket \cap \mathrm{Pre}\big(h^{-1}(\llbracket\psi\rrbracket_v)\big)\big)\big)\\
&\qquad\text{(Lemma 3.3)}\\
&= h^{-1}\big(h\big(\llbracket\mathbf{s}\rrbracket \cap \mathrm{Pre}(\llbracket\tilde{\psi}\rrbracket_{v'})\big)\big) \quad \text{by ind. hyp.}
\end{aligned}$$

This is $h^{-1}(h(\llbracket\mathbf{s} \wedge \Diamond\tilde{\psi}\rrbracket_{v'}))$, or $\llbracket\widetilde{\Diamond\psi}\rrbracket_{v'}$ (Lemma 3.5).

Remaining cases: The case where φ is some $\varphi_1 \wedge \varphi_2$ is obvious and the remaining cases are obtained by duality. □

Corollary 3.7. *For $x' \in h^{-1}(x)$ and φ a closed formula, $x \models_K \varphi$ iff $x' \models_L \tilde{\varphi}$.*

Proof. Lemma 3.6 provides the "⇒" direction, and the "⇐" direction too once we observe that $h \circ h^{-1} = Id_Q$. □

Regarding alternation depth, we refer to [10,2]. A μ-calculus formula is in Σ_0 ($= \Pi_0$) iff it contains not fixpoint operation. Then, for $n \in \mathbb{N}$, Σ_{n+1} is defined as the smallest class of formulae that contains $\Sigma_n \cup \Pi_n$ and is closed under conjunctions and disjunctions, $\Diamond$- and $\Box$-modalities, least fixed points $\mu Z.\varphi$ with $\varphi \in \Sigma_{n+1}$, and substitution of $\varphi' \in \Sigma_{n+1}$ for a free variable of a formula $\varphi \in \Sigma_{n+1}$, provided that no free variable of φ' is captured by φ. Π_{n+1} is defined dually.

Proposition 3.8. *If $\varphi \in \Sigma_n$ (or dually, Π_n), then $\tilde{\varphi}$ is in $\Sigma_{\max(n,2)}$ (resp. $\Pi_{\max(n,2)}$).*

Proof. By induction on the structure of φ. The only difficult cases are $\Diamond$- and $\Box$-formulae. If $\varphi = \Diamond\psi$, with $\psi \in \Sigma_n$, the induction hypothesis yields that $\tilde{\psi} \in \Sigma_{\max(n,1)}$. Then $\tilde{\varphi}$ is obtained from $\mu Z.[(\mathbf{s} \wedge \Diamond W) \vee \Theta(Z)]$, a Σ_1-formula, by substituting $\tilde{\psi}$ for W. If $\varphi = \Box\psi$, we substitute in a Π_1 (hence Σ_2) formula. □

4. Finite paths and acyclic structures

It is well known that, for acyclic KS's, B_μ model checking can be done in polynomial-time (hence is PTIME-complete), see, e.g., [9]. Thus model checking finite paths is in polynomial-time and it is not surprising that we could not reduce model checking of loops to model checking of paths: with Theorem 3.1, this would have solved the general B_μ model-checking problem.

However, even if finite paths seem easier than finite loops, they are not easier than arbitrary acyclic KS's as we now show.

Theorem 4.1. *B_μ model checking of finite paths is* PTIME-*complete.*

For this result, it turns out that the reduction from the previous section adapts very easily. If we omit the step $d_n \to s_1$ that closed the loop, we obtain a finite path where, assuming that the transitions $R = \{r_1, \ldots, r_n\}$ of the acyclic K are given in some topological order, for every vertex of K, the *destination* copies (if any) occur before the *source* copies. That way, we get:

Lemma 4.2. *Given $x', y' \in Q'$ s.t. $h(x') = h(y')$ and x' occurs before y', for any formula $\varphi \in B_\mu$ and any context $v : \mathcal{V} \to 2^Q$, writing $v' = h^{-1} \circ v$, we have: if $y' \in [\![\tilde{\varphi}]\!]^{K'}_{v'}$, then $x' \in [\![\tilde{\varphi}]\!]^{K'}_{v'}$.*

That result can easily be shown by induction. We then obtain weaker versions of Lemmas 3.4–3.6:

Lemma 4.3. *Assuming Y and Z are distinct variables, for any context v', we have*

$$h\big([\![\Theta(Y)]\!]^{K'}_{v'}\big) = h\big([\![Y]\!]^{K'}_{v'}\big) = h\big([\![\mu Z.(Y \vee \Theta(Z))]\!]^{K'}_{v'}\big).$$

Lemma 4.4. *For any formula φ of B_μ involving atomic propositions in* AP, *context $v : \mathcal{V} \to 2^Q$, and writing v' for $h^{-1} \circ v$:*

$$[\![\varphi]\!]^K_v = h\big([\![\tilde{\varphi}]\!]^{K'}_{v'} \cap [\![\mathbf{s}]\!]\big),$$
$$h^{-1}\big([\![\varphi]\!]^K_v\big) \cap [\![\mathbf{d}]\!] = [\![\tilde{\varphi}]\!]^{K'}_{v'} \cap [\![\mathbf{d}]\!].$$

Now, clearly, a state in K satisfies formula φ iff its first *source* copy in L satisfies $\tilde{\varphi}$.

5. Paths, loops, and backwards modalities

Model checking of loops reduces to finite paths when one considers $2B_\mu$, or "2-way B_μ", the extension of B_μ with backwards modalities $\Diamond^{-1}$ and $\Box^{-1}$. One lets $x \in [\![\Diamond^{-1}\varphi]\!]$ iff there is some $y \in [\![\varphi]\!]$ with $y \to x$, and dually for $\Box^{-1}$ [13].

Theorem 5.1. *The following three problems are log-space inter-reducible*:

(a) *B_μ model checking of loops,*
(b) *$2B_\mu$ model checking of loops,*
(c) *$2B_\mu$ model checking of finite paths.*

Corollary 5.2. *These three problems are equivalent to B_μ model checking on arbitrary KS's. They are thus* PTIME-*hard, and in* UP ∩ coUP.

Proof of Theorem 5.1. Since (a) is a special case of (b), we only need two reductions.

((b) *reduces to* (c)) Let L be a loop $x_1 \to x_2 \to \cdots \to x_n$ $(\to x_1)$. With L, the reduction associates a finite path F of the form $x_0 \to x_1 \to x_2 \to \cdots \to x_n \to x_{n+1}$. The labeling of F is inherited from L (and irrelevant for x_0 and x_{n+1}). The reduction translates a formula φ to a φ' such that $[\![\varphi']\!]^F \setminus \{x_0, x_{n+1}\} = [\![\varphi]\!]^L$. The translation is obtained with

$$(\Diamond\psi)' \stackrel{\text{def}}{=} \mu Z.\big((\Diamond\psi' \wedge \Diamond\Diamond\top) \vee (\Diamond^{-1})^n Z\big),$$
$$(\Diamond^{-1}\psi)' \stackrel{\text{def}}{=} \mu Z.\big((\Diamond^{-1}\psi' \wedge \Diamond^{-1}\Diamond^{-1}\top) \vee (\Diamond)^n Z\big).$$

One adds dual clauses for $(\Box\psi)'$ and $(\Box^{-1}\psi)'$, and obvious clauses, like $(\mu Z.\psi)' \stackrel{\text{def}}{=} \mu Z.(\psi')$, for the other constructs. Then $|\varphi'|$ is in $O(|\varphi| \cdot |L|)$.

((c) *reduces to* (a)) Let F be a finite path $x_1 \to x_2 \to \cdots \to x_n$. A loop L is obtained from F by adding a transition $x_n \to x_1$ and labeling x_1 with a new additional proposition $\mathbf{i}$. The reduction then translates a formula φ to a φ' without backwards modalities, and such that $[\![\varphi']\!]^L = [\![\varphi]\!]^F$. We use

$$(\Diamond\psi)' \stackrel{\text{def}}{=} \Diamond(\psi' \wedge \neg\mathbf{i}) \quad \text{and}$$
$$(\Diamond^{-1}\psi)' \stackrel{\text{def}}{=} \neg\mathbf{i} \wedge \Diamond^{n-1}\psi'$$

and obvious remaining clauses. Again, $|\varphi'|$ is in $O(|\varphi| \cdot |L|)$. □

6. Conclusion

We proved that μ-calculus model checking is not easier when restricting to deterministic Kripke structures having the form of a single loop. On the other hand, we could not reduce model checking of finite loops to model checking of finite paths, a PTIME-complete problem. These results help understand what makes μ-calculus model checking difficult.

It comes as a surprise that none of these two results fits the pattern we exhibited for several other logics [8], where checking nondeterministic KS's is harder than checking deterministic loops, and where finite loops are no harder than finite paths. A possible explanation for the first discrepancy is the expressive power of the μ-calculus, that allows the reduction we developed in Section 3. The second discrepancy is harder to justify, but would disappear if μ-calculus model checking were proved to be in PTIME.

Acknowledgements

We thank Misa Keinänen for drawing our attention to the μ-calculus path model-checking problem.

References

[1] C. Artho, H. Barringer, A. Goldberg, K. Havelund, S. Khurshid, M. Lowry, C. Pasareanu, G. Roşu, K. Sen, W. Visser, R. Washington, Combining test case generation and runtime verification, Theoret. Comput. Sci. 336 (2–3) (2005) 209–234.

[2] J.C. Bradfield, The modal mu-calculus alternation hierarchy is strict, Theoret. Comput. Sci. 195 (2) (1998) 133–153.
[3] J.C. Bradfield, C. Stirling, Modal logics and mu-calculi: an introduction, in: Handbook of Process Algebra, Elsevier, Amsterdam, 2001, pp. 293–330 (Chapter 4).
[4] S. Demri, Ph. Schnoebelen, The complexity of propositional linear temporal logics in simple cases, Inform. and Comput. 174 (1) (2002) 84–103.
[5] K. Havelund, G. Roşu, An overview of the runtime verification tool Java PathExplorer, Formal Methods in System Design 24 (2) (2004) 189–215.
[6] T. Hérault, R. Lassaigne, F. Magniette, S. Peyronnet, Approximate probabilistic model checking, in: Proc. 5th Internat. Conf. Verification, Model Checking, and Abstract Interpretation (VMCAI'04), Venice, Italy, January 2004, in: Lecture Notes in Comput. Sci., vol. 2937, Springer, Berlin, 2004, pp. 73–84.
[7] M. Jurdziński, Deciding the winner in parity games is in UP $\cap$ coUP, Inform. Process. Lett. 68 (3) (1998) 119–124.
[8] N. Markey, Ph. Schnoebelen, Model checking a path (preliminary report), in: Proc. 14th Internat. Conf. Concurrency Theory (CONCUR'03), Marseille, France, August 2003, in: Lecture Notes in Comput. Sci., vol. 2761, Springer, Berlin, 2003, pp. 251–265.
[9] R. Mateescu, Local model-checking of modal mu-calculus on acyclic labeled transition systems, in: Proc. 8th Internat. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'02), Grenoble, France, April 2002, in: Lecture Notes in Comput. Sci., vol. 2280, Springer, Berlin, 2002, pp. 281–295.
[10] D. Niwiński, On fixed point clones, in: Proc. 13th Internat. Coll. Automata, Languages and Programming (ICALP'86), Rennes, France, July 1986, in: Lecture Notes in Comput. Sci., vol. 226, Springer, Berlin, 1986, pp. 464–473.
[11] M. Roger, J. Goubault-Larrecq, Log auditing through model checking, in: Proc. 14th IEEE Computer Security Foundations Workshop (CSFW'01), Cape Breton, Nova Scotia, Canada, June, IEEE Comp. Soc. Press, 2001, pp. 220–236.
[12] M.Y. Vardi, A temporal fixpoint calculus, in: Proc. 15th ACM Symp. Principles of Programming Languages (POPL'88), San Diego, CA, USA, January 1988, pp. 250–259.
[13] M.Y. Vardi, Reasoning about the past with two-way automata, in: Proc. 25th Internat. Coll. Automata, Languages, and Programming (ICALP'98), Aalborg, Denmark, July 1998, in: Lecture Notes in Comput. Sci., vol. 1443, Springer, Berlin, 1998, pp. 628–641.