Mu-calculus path checking

Nicolas Markey and Philippe Schnoebelen

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

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.

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).

Email addresses: `markey@lsv.ens-cachan.fr` (Nicolas Markey), `phs@lsv.ens-cachan.fr` (Philippe Schnoebelen).

Preprint submitted to Elsevier Science

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].

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: $\mathbf{EF}\varphi \stackrel{\text{def}}{=} \mu Z.(\varphi \vee \Diamond Z)$ and $\mathbf{AG}\varphi \stackrel{\text{def}}{=} \nu Z.(\varphi \wedge \Box Z)$ 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 $v \colon \mathcal{V} \to 2^Q$, the set $[\![\varphi]\!]^K_v$ of states where φ holds is defined inductively:

$$\llbracket p \rrbracket_v^K \stackrel{\text{def}}{=} \{x \in Q \mid p \in l(x)\} \qquad \llbracket \neg p \rrbracket_v^K \stackrel{\text{def}}{=} \{x \in Q \mid p \notin l(x)\}$$
$$\llbracket \varphi \vee \psi \rrbracket_v^K \stackrel{\text{def}}{=} \llbracket \varphi \rrbracket_v^K \cup \llbracket \psi \rrbracket_v^K \qquad \llbracket \varphi \wedge \psi \rrbracket_v^K \stackrel{\text{def}}{=} \llbracket \varphi \rrbracket_v^K \cap \llbracket \psi \rrbracket_v^K$$
$$\llbracket Z \rrbracket_v^K \stackrel{\text{def}}{=} v(Z)$$
$$\llbracket \Diamond \varphi \rrbracket_v^K \stackrel{\text{def}}{=} \mathrm{Pre}\left(\llbracket \varphi \rrbracket_v^K\right) \qquad \llbracket \Box \varphi \rrbracket_v^K \stackrel{\text{def}}{=} \overline{\mathrm{Pre}\left(\overline{\llbracket \varphi \rrbracket_v^K}\right)}$$
$$\llbracket \mu Z.\varphi \rrbracket_v^K \stackrel{\text{def}}{=} \bigcap \{U \subseteq Q \mid \llbracket \varphi \rrbracket_{v[Z \mapsto U]}^K \subseteq U\}$$
$$\llbracket \nu Z.\varphi \rrbracket_v^K \stackrel{\text{def}}{=} \bigcup \{U \subseteq Q \mid U \subseteq \llbracket \varphi \rrbracket_{v[Z \mapsto U]}^K\}$$

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 $x \in \llbracket \varphi \rrbracket_v^K$. The above definition entails the following standard *fixed-point equalities*:

$$\llbracket \mu Z.\varphi \rrbracket_v = \llbracket \varphi \rrbracket_{v\left[Z \mapsto \llbracket \mu Z.\varphi \rrbracket_v\right]} \qquad \llbracket \nu Z.\varphi \rrbracket_v = \llbracket \varphi \rrbracket_{v\left[Z \mapsto \llbracket \nu Z.\varphi \rrbracket_v\right]}.$$

For $\alpha \in \mathbb{N}$, the *approximant* $\llbracket \mu Z^\alpha.\varphi \rrbracket_v^K$ is defined inductively by

$$\llbracket \mu Z^0.\varphi \rrbracket_v \stackrel{\text{def}}{=} \emptyset \qquad \text{and} \qquad \llbracket \mu Z^{\alpha+1}.\varphi \rrbracket_v \stackrel{\text{def}}{=} \llbracket \varphi \rrbracket_{v\left[Z \mapsto \llbracket \mu Z^\alpha \varphi \rrbracket_v\right]}.$$

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

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** $\cap$ **coUP**.

The rest of this section describes our reduction. We transform an instance "$x \models^K \varphi$?" into an equivalent "$x' \models^L \widetilde{\varphi}$?" where L is a loop. We observe that $|L| = O(|K|)$,

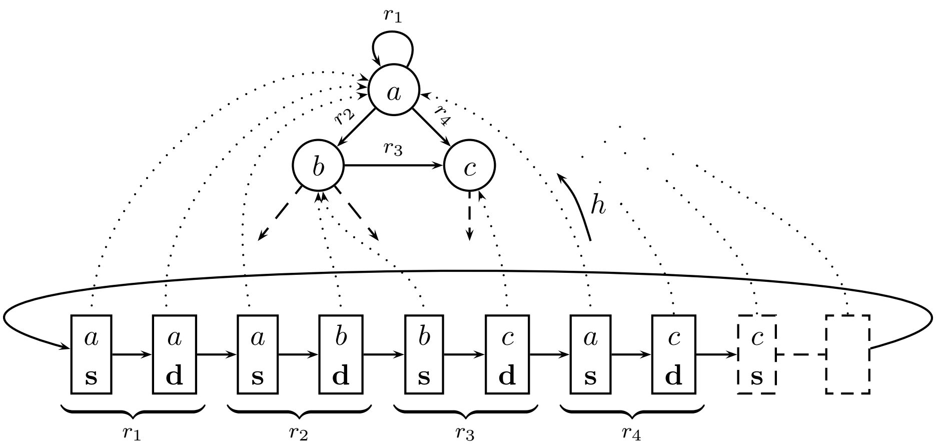

Fig. 1. From non-deterministic to deterministic Kripke structure

and $|\widetilde{\varphi}| = O(|K|\cdot|\varphi|)$. Furthermore, the transformation from φ to $\widetilde{\varphi}$ does not increase the alternation depth (Prop. 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 $\mathbf{s}$ (for *source*) and $\mathbf{d}$ (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 $1 \le i \le n$, 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 $h\colon Q' \to Q$ 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* $\mathrm{Pre}_K(S) = h\left([\![\mathbf{s}]\!]^L \cap \mathrm{Pre}_L\left(h^{-1}(S)\right)\right)$.

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 $x \in \mathrm{Pre}_K(S)$. $\square$

Now, define $\Theta(Z) \stackrel{\text{def}}{=} \bigvee_{x\in Q}[x \wedge \mathbf{EF}(x \wedge Z)]$ and $\Xi(Z) \stackrel{\text{def}}{=} \bigwedge_{x\in Q}[x \Rightarrow \mathbf{AG}(x \Rightarrow Z)]$.

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

[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.

PROOF. $[\![\Theta(Z)]\!]_v$ is $\bigcup_{x\in Q}[\![x \wedge \mathbf{EF}(x \wedge Z)]\!]_v$. 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. $\square$

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

$$[\![\mu Z.(Y \vee \Theta(Z))]\!]^L_v = \Theta(Y) = h^{-1}\left(h\left([\![Y]\!]^L_v\right)\right)$$
$$[\![\nu Z.(Y \wedge \Xi(Z))]\!]^L_v = \Xi(Y) = \overline{h^{-1}\left(h\left(\overline{[\![Y]\!]^L_v}\right)\right)}.$$

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

($\subseteq$): Write U for $h^{-1}(h([\![Y]\!]_v))$. Then $[\![Y \vee \Theta(Z)]\!]_{v[Z\mapsto U]} = [\![Y]\!]_v \cup [\![\Theta(Z)]\!]_{v[Z\mapsto U]} = [\![Y]\!]_v \cup h^{-1}(h(U))$ (by Lemma 3.4) $= U$. Hence U is a fixed point and $[\![\mu Z.(Y \vee \Theta(Z))]\!]_v \subseteq U$.

($\supseteq$): Write S for $[\![\mu Z.(Y \vee \Theta(Z))]\!]_v$. From the fixed-point property, we have $S = [\![Y \vee \Theta(Z)]\!]_{v[Z\mapsto S]} = [\![Y]\!]_v \cup [\![\Theta(S)]\!]_v = [\![Y]\!]_v \cup h^{-1}(h(S))$ (by Lemma 3.4). Hence $S \supseteq h^{-1}(h([\![Y]\!]_v))$. $\square$

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 $O(|Q| \cdot |\psi|)$ while $|\mu Z.(\psi \vee \Theta(Z))|$ is in $O(|Q| + |\psi|)$.

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

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

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

$$h^{-1}\left([\![\varphi]\!]^K_v\right) = [\![\widetilde{\varphi}]\!]^L_{v'} \tag{1}$$

In other words, $x' \in [\![\widetilde{\varphi}]\!]^L_{v'}$ iff $h(x') \in [\![\varphi]\!]^K_v$.

PROOF. By induction on the structure of φ.

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

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, $h^{-1}(\llbracket \mu Z^\alpha.\psi \rrbracket_v) = \llbracket \mu Z^\alpha.\widetilde{\psi} \rrbracket_{v'}$. We proceed by induction on α. The base case where $\alpha = 0$ holds trivially, and the inductive step relies on $h^{-1}(\llbracket \mu Z^{\alpha+1}.\psi \rrbracket_v) = h^{-1}(\llbracket \psi \rrbracket_{v[Z \mapsto \llbracket \mu Z^\alpha.\psi \rrbracket_v]}) = \llbracket \widetilde{\psi} \rrbracket_{h^{-1} \circ v[Z \mapsto \llbracket \mu Z^\alpha.\psi \rrbracket_v]}$ by ind. hyp. (Lemma 3.6 on ψ). This is $\llbracket \widetilde{\psi} \rrbracket_{v'[Z \mapsto h^{-1}(\llbracket \mu Z^\alpha.\psi \rrbracket_v)]} = \llbracket \widetilde{\psi} \rrbracket_{v'[Z \mapsto \llbracket \mu Z^\alpha.\widetilde{\psi} \rrbracket_{v'}]}$ (by ind. hyp. on α), hence equals $\llbracket \mu Z^{\alpha+1}.\widetilde{\psi} \rrbracket_{v'}$.

Case $\varphi = \Diamond\psi$: $h^{-1}(\llbracket \Diamond\psi \rrbracket_v) = h^{-1}(\mathrm{Pre}(\llbracket \psi \rrbracket_v)) = h^{-1}(h(\llbracket \mathbf{s} \rrbracket \cap \mathrm{Pre}(h^{-1}(\llbracket \psi \rrbracket_v))))$ (Lemma 3.3) $= h^{-1}(h(\llbracket \mathbf{s} \rrbracket \cap \mathrm{Pre}(\llbracket \widetilde{\psi} \rrbracket_{v'})))$ by ind. hyp. This is $h^{-1}(h(\llbracket \mathbf{s} \wedge \Diamond\widetilde{\psi} \rrbracket_{v'}))$, 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 $x' \models_L \widetilde{\varphi}$.*

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 $\widetilde{\varphi}$ 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 $\widetilde{\psi} \in \Sigma_{\max(n,1)}$. Then $\widetilde{\varphi}$ is obtained from $\mu Z.\,[(\mathbf{s} \wedge \Diamond W) \vee \Theta(Z)]$, a Σ_1-formula, by substituting $\widetilde{\psi}$ 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 $v\colon \mathcal{V} \to 2^Q$, writing $v' = h^{-1} \circ v$, we have: if $y' \in [\![\widetilde{\varphi}]\!]^{K'}_{v'}$, then $x' \in [\![\widetilde{\varphi}]\!]^{K'}_{v'}$.*

That result can easily be shown by induction. We then obtain weaker versions of Lemmas 3.4, 3.5 and 3.6:

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

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

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

$$[\![\varphi]\!]^{K}_{v} = h\left([\![\widetilde{\varphi}]\!]^{K'}_{v'} \cap [\![\mathbf{s}]\!]\right) \qquad h^{-1}\left([\![\varphi]\!]^{K}_{v}\right) \cap [\![\mathbf{d}]\!] = [\![\widetilde{\varphi}]\!]^{K'}_{v'} \cap [\![\mathbf{d}]\!]$$

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

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 logspace 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* $\mathbf{UP} \cap \mathbf{coUP}$.

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 $x_1 \to x_2 \to \cdots x_n (\to x_1)$. With L, the reduction associates a finite path F of the form $x_0 \to x_1 \to x_2 \to \cdots x_n \to x_{n+1}$. 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 $x_1 \to x_2 \to \cdots x_n$. 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

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

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

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**.

Acknowledgments. 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, and R. Washington. Combining

test case generation and runtime verification. *Theoretical Computer Science*, 336(2-3):209–234, 2005.

[2] J. C. Bradfield. The modal mu-calculus alternation hierarchy is strict. *Theoretical Computer Science*, 195(2):133–153, 1998.

[3] J. C. Bradfield and C. Stirling. Modal logics and mu-calculi: an introduction. In *Handbook of Process Algebra*, chapter 4, pages 293–330. Elsevier, 2001.

[4] S. Demri and Ph. Schnoebelen. The complexity of propositional linear temporal logics in simple cases. *Information and Computation*, 174(1):84–103, 2002.

[5] K. Havelund and G. Roşu. An overview of the runtime verification tool Java PathExplorer. *Formal Methods in System Design*, 24(2):189–215, 2004.

[6] T. Hérault, R. Lassaigne, F. Magniette, and S. Peyronnet. Approximate probabilistic model checking. In *Proc. 5th Int. Conf. Verification, Model Checking, and Abstract Interpretation (VMCAI'04), Venice, Italy, Jan. 2004*, volume 2937 of *LNCS*, pages 73–84. Springer, 2004.

[7] M. Jurdziński. Deciding the winner in parity games is in $\mathbf{UP} \cap \mathbf{coUP}$. *Information Processing Letters*, 68(3):119–124, 1998.

[8] N. Markey and Ph. Schnoebelen. Model checking a path (preliminary report). In *Proc. 14th Int. Conf. Concurrency Theory (CONCUR'03), Marseille, France, August 2003*, volume 2761 of *LNCS*, pages 251–265. Springer, 2003.

[9] R. Mateescu. Local model-checking of modal mu-calculus on acyclic labeled transition systems. In *Proceedings of the 8th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'02), Grenoble, France, April 2002*, volume 2280 of *LNCS*, pages 281–295. Springer, 2002.

[10] D. Niwiński. On fixed point clones. In *Proc. 13th Int. Coll. Automata, Languages and Programming (ICALP'86), Rennes, France, July 1986*, volume 226 of *LNCS*, pages 464–473. Springer, 1986.

[11] M. Roger and J. Goubault-Larrecq. Log auditing through model checking. In *Proc, 14th IEEE Computer Security Foundations Workshop (CSFW'01)*, pages 220–236, Cape Breton, Nova Scotia, Canada, June 2001. IEEE Comp. Soc. Press.

[12] M. Y. Vardi. A temporal fixpoint calculus. In *Proc. 15th ACM Symp. Principles of Programming Languages (POPL'88), San Diego, CA, USA, Jan. 1988*, pages 250–259, 1988.

[13] M. Y. Vardi. Reasoning about the past with two-way automata. In *Proc. 25th Int. Coll. Automata, Languages, and Programming (ICALP'98), Aalborg, Denmark, July 1998*, volume 1443 of *LNCS*, pages 628–641. Springer, 1998.