



Mu-Calculus Path Checking

Nicolas Markey, Philippe Schnoebelen

► To cite this version:

Nicolas Markey, Philippe Schnoebelen. Mu-Calculus Path Checking. Information Processing Letters, 2006, 97 (6), pp.225-230. 10.1016/j.ipl.2005.11.010 . hal-01194606

HAL Id: hal-01194606

<https://hal.science/hal-01194606>

Submitted on 7 Sep 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

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).

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 $\mathbf{UP} \cap \mathbf{coUP}$ [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 $l : Q \rightarrow 2^{\text{AP}}$ is the state labeling. As usual, and when R is understood, we write $x \rightarrow y$ rather than $(x, y) \in R$, and we say y is a *successor* of x . Given $S \subseteq Q$, we write $\text{Pre}(S)$ for the set $\{x \in Q \mid \exists y \in S. x \rightarrow y\}$, and $\overline{S}$ for $Q \setminus S$. Then $x \in \text{Pre}(\overline{S})$ iff all the successors of x (if any) are in S .

Formally, for a KS $K = (Q, R, l)$ and a context $v : \mathcal{V} \rightarrow 2^Q$, the set $\llbracket \varphi \rrbracket_v^K$ of states

where φ holds is defined inductively:

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

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

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

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

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

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

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 $|L| = O(|K|)$,

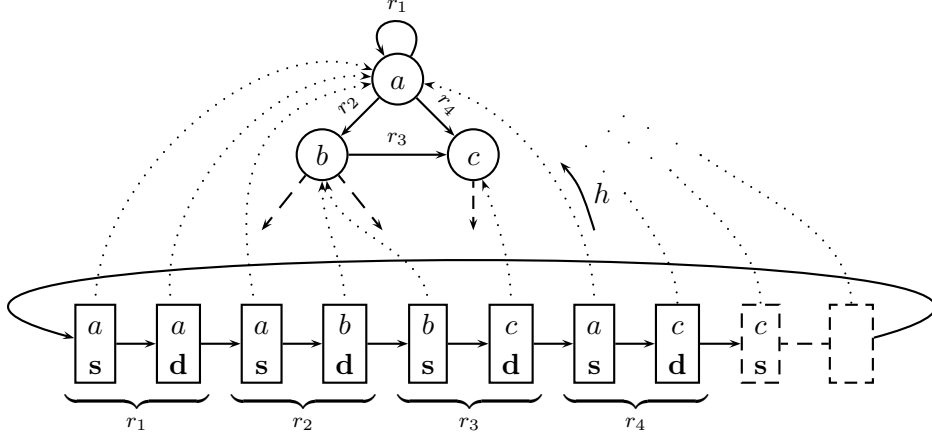


Fig. 1. From non-deterministic to deterministic Kripke structure

and $|\tilde{\varphi}| = O(|K| \cdot |\varphi|)$. Furthermore, the transformation from φ to $\tilde{\varphi}$ 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 $\text{AP}' \stackrel{\text{def}}{=} \text{AP} \cup \{\mathbf{s}, \mathbf{d}\}$ where $\mathbf{s}$ (for *source*) and $\mathbf{d}$ (for *destination*) are two new atomic propositions. Assume $R = \{r_1, \dots, r_n\}$ contains n transitions: then $L = (Q', R', l')$ has $Q' \stackrel{\text{def}}{=} \{s_1, d_1, s_2, d_2, \dots, s_n, d_n\}$. R' has transitions $s_i \rightarrow d_i$ and $d_i \rightarrow s_{(i \bmod n) + 1}$ for $1 \leq i \leq n$, 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 $h: Q' \rightarrow Q$ 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 $\text{Pre}_K(S) = h(\llbracket \mathbf{s} \rrbracket^L \cap \text{Pre}_L(h^{-1}(S)))$.*

PROOF. Assume $x \in \text{Pre}_K(S)$ because of a transition r_i of the form $x \rightarrow y$ with $y \in S$. In L , $s_i \rightarrow d_i$ has $d_i \in h^{-1}(y) \subseteq h^{-1}(S)$ and $s_i \in \llbracket \mathbf{s} \rrbracket^L$. Hence $x = h(s_i) \in h(\llbracket \mathbf{s} \rrbracket^L \cap \text{Pre}_L(h^{-1}(S)))$. Conversely, if $x \in h(\llbracket \mathbf{s} \rrbracket^L \cap \text{Pre}_L(h^{-1}(S)))$, then $x = h(s_i)$ for some i such that $h(d_i) \in S$. Therefore r_i shows that $x \in \text{Pre}_K(S)$. $\square$

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 , $\llbracket \Theta(Z) \rrbracket_v^L = h^{-1}(h(\llbracket Z \rrbracket_v^L))$ and $\llbracket \Xi(Z) \rrbracket_v^L = \overline{h^{-1}(h(\llbracket Z \rrbracket_v^L))}$.*

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 disjunction of (the propositions denoting) the states where it holds. This transformation is logspace.

PROOF. $\llbracket \Theta(Z) \rrbracket_v$ is $\bigcup_{x \in Q} \llbracket x \wedge \mathbf{EF}(x \wedge Z) \rrbracket_v$. Since L is strongly connected, this is $\{x' \mid \exists y' \in \llbracket Z \rrbracket_v, h(x') = h(y')\}$ by definition of l' . We end up with $h^{-1}(h(\llbracket Z \rrbracket_v))$. The second result follows by duality. $\square$

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

$$\begin{aligned} \llbracket \mu Z.(Y \vee \Theta(Z)) \rrbracket_v^L &= \Theta(Y) = h^{-1}\left(h\left(\llbracket Y \rrbracket_v^L\right)\right) \\ \llbracket \nu Z.(Y \wedge \Xi(Z)) \rrbracket_v^L &= \Xi(Y) = \overline{h^{-1}\left(h\left(\overline{\llbracket Y \rrbracket_v^L}\right)\right)}. \end{aligned}$$

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

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

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

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 $\tilde{\varphi}$ 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:

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

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

$$h^{-1}\left(\llbracket \varphi \rrbracket_v^K\right) = \llbracket \tilde{\varphi} \rrbracket_{v'}^L \quad (1)$$

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 $\varphi = p \in AP$: Since $AP = Q$, 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 $h^{-1}(\llbracket \mu Z^{\alpha+1}.\psi \rrbracket_v) = h^{-1}(\llbracket \psi \rrbracket_{v[Z \mapsto \llbracket \mu Z^\alpha.\psi \rrbracket_v]}) = \llbracket \tilde{\psi} \rrbracket_{h^{-1} \circ v[Z \mapsto \llbracket \mu Z^\alpha.\psi \rrbracket_v]}$ by ind. hyp. (Lemma 3.6 on ψ). This is $\llbracket \tilde{\psi} \rrbracket_{v'[Z \mapsto h^{-1}(\llbracket \mu Z^\alpha.\psi \rrbracket_v)]} = \llbracket \tilde{\psi} \rrbracket_{v'[Z \mapsto \llbracket \mu Z^\alpha.\tilde{\psi} \rrbracket_{v'}]}$ (by ind. hyp. on α), hence equals $\llbracket \mu Z^{\alpha+1}.\tilde{\psi} \rrbracket_{v'}$.

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

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 “ $\Rightarrow$ ” direction, and the “ $\Leftarrow$ ” direction too once we observe that $h \circ h^{-1} = \text{Id}_Q$. $\square$

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. $\square$

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 **P**TIME-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 $d_n \rightarrow s_1$ that closed the loop, we obtain a finite path where, assuming that the transitions $R = \{r_1, \dots, r_n\}$ 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 $v: \mathcal{V} \rightarrow 2^Q$, writing $v' = h^{-1} \circ v$, we have: if $y' \in \llbracket \tilde{\varphi} \rrbracket_{v'}^{K'}$, then $x' \in \llbracket \tilde{\varphi} \rrbracket_{v'}^{K'}$.*

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*

$$h\left(\llbracket \Theta(Y) \rrbracket_{v'}^{K'}\right) = h\left(\llbracket Y \rrbracket_{v'}^{K'}\right) = h\left(\llbracket \mu Z.(Y \vee \Theta(Z)) \rrbracket_{v'}^{K'}\right)$$

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

$$\llbracket \varphi \rrbracket_v^K = h\left(\llbracket \tilde{\varphi} \rrbracket_{v'}^{K'} \cap \llbracket \mathbf{s} \rrbracket\right) \quad h^{-1}\left(\llbracket \varphi \rrbracket_v^K\right) \cap \llbracket \mathbf{d} \rrbracket = \llbracket \tilde{\varphi} \rrbracket_{v'}^{K'} \cap \llbracket \mathbf{d} \rrbracket$$

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 $x \in \llbracket \Diamond^{-1}\varphi \rrbracket$ iff there is some $y \in \llbracket \varphi \rrbracket$ with $y \rightarrow x$, 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 $x_1 \rightarrow x_2 \rightarrow \dots x_n (\rightarrow x_1)$. With L , the reduction associates a finite path F of the form $x_0 \rightarrow x_1 \rightarrow x_2 \rightarrow \dots x_n \rightarrow x_{n+1}$. 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 $\llbracket \varphi' \rrbracket^F \setminus \{x_0, x_{n+1}\} = \llbracket \varphi \rrbracket^L$. The translation is obtained with

$$\begin{aligned} (\Diamond \psi)' &\stackrel{\text{def}}{=} \mu Z. \left((\Diamond \psi' \wedge \Diamond \Diamond \top) \vee (\Diamond^{-1})^n Z \right) \\ (\Diamond^{-1} \psi)' &\stackrel{\text{def}}{=} \mu Z. \left((\Diamond^{-1} \psi' \wedge \Diamond^{-1} \Diamond^{-1} \top) \vee (\Diamond)^n Z \right) \end{aligned}$$

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 $x_1 \rightarrow x_2 \rightarrow \dots x_n$. A loop L is obtained from F by adding a transition $x_n \rightarrow x_1$ and labeling x_1 with a new additional proposition **i**. The reduction then translates a formula φ to a φ' without backwards modalities, and such that $\llbracket \varphi' \rrbracket^L = \llbracket \varphi \rrbracket^F$. We use

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

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

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 **P**TIME-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 **P**TIME.

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. Héruault, 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.