Bounded Model Checking for All Regular Properties

Markus Jehle Jan Johannsen Martin Lange
Nicolas Rachinsky

Institut für Informatik, LMU München

Abstract

The technique of bounded model checking is extended to the linear time μ-calculus, a temporal logic that can express all monadic second-order properties of ω-words, in other words, all ω-regular languages. Experimental evidence is presented showing that the method can be successfully employed for properties that are hard or impossible to express in the weaker logic LTL that is traditionally used in bounded model checking.

Key words: model checking, satisfiability solving, expressiveness

1 Introduction

Bounded model checking is a verification technique for linear time properties. Only paths of a certain length through a transition system are considered. It is therefore not complete but only an approximation method relying on the fact that unsatisfied formulas often have short counterexamples.

On the other hand, the boundedness plus the fact that models are linear structures make the problem suitable for a reduction to SAT - the satisfiability problem for propositional logic. It is known from a different symbolic technique, namely BDD-based model checking [4], that transition systems can be encoded as boolean functions, and that these encodings can be significantly smaller than explicit representations.

So far, bounded model checking has been employed for LTL [10] and variants thereof. But the expressive power of LTL is rather limited: it is equi-expressive to First-Order Logic over ω-words [7,6], resp. star-free languages [13].

Inspired by the success that bounded model checking for LTL has had so far [3], we show how to do bounded model checking for the linear time μ-calculus μTL [1]. It is a natural, nonetheless less known, temporal fixpoint logic that can be obtained in two different ways. Either one replaces the temporal *until* in LTL by arbitrary least fixpoint constructs; or one replaces

This is a preliminary version. The final version will be published in
Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

the modal operators in the modal μ-calculus by a single *next* operator. μTL is a natural logic because it is expressive complete w.r.t. Monadic Second Order Logic over infinite words [1], i.e. it can define every ω-regular property. Thus, this paper increases the set of properties which are known to be verifiable using bounded model checking up to all ω-regular properties.

Unlike the modal μ-calculus, μTL does not have a strict alternation hierarchy. Therefore, every μTL formula can be transformed into an equivalent alternation-free formula. This translation is exponential in the alternation depth of the original formula. However, formulas with a lot of alternation are hardly seen as specifications because they are not easy to read. The encoding into SAT presented here makes use of this result.

The rest of the paper is organised as follows. Section 2 recalls μTL. Section 3 presents some examples of regular, i.e. μTL-definable properties that cannot be expressed in LTL. Section 4 defines a bounded semantics for μTL along the same lines as the one for LTL [3]. Section 5 contains the reduction from μTL formulas over paths of bounded length into SAT. Section 6 reports on a prototype implementation of this translation and presents experimental results.

What remains to do done is to check which known optimisations for bounded model checking LTL can be transferred to μTL, to find small completeness thresholds like it was done for LTL, too [3,5], etc.

2 Preliminaries

2.1 *The Linear Time μ-Calculus* μTL

Let $\mathcal{P}$ be a set of propositions which contains $\mathtt{tt}$ and $\mathtt{ff}$ and is closed under complementation, i.e., for every $q \in \mathcal{P}$ there is an $\bar{q} \in \mathcal{P}$ with $\bar{\bar{q}} = q$. Let $\mathcal{V}$ be a set of monadic second-order variables. Formulas of μTL in positive normal form are given by the following grammar.

$$\varphi \quad ::= \quad q \mid X \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \bigcirc \varphi \mid \mu X.\varphi \mid \nu X.\varphi$$

where $q \in \mathcal{P}$ and $X \in \mathcal{V}$. The set $Sub(\varphi)$ of subformulas of φ is defined as usual, e.g. $Sub(\mu X.\varphi) := \{\mu X.\varphi\} \cup Sub(\varphi)$.

Formulas are assumed to be well-named, i.e., no variable is bound more than once in a formula. Then for each $\varphi \in \mu$TL there is a function $fp_\varphi : \mathcal{V} \cap Sub(\varphi) \to Sub(\varphi)$ that maps each variable X occurring in φ to its defining fixpoint formula $\sigma X.\psi$. If $fp_\varphi(X)$ is $\mu X.\psi$ for some formula ψ, we say that X is of type μ, otherwise X is of type ν.

A total, labeled transition system (LTS) is a tuple $\mathcal{T} = (\mathcal{S}, \longrightarrow, \mathcal{I}, S_0)$ where $\mathcal{S}$ is a set of states. $\longrightarrow$ is a binary relation on states s.t. for every $s \in \mathcal{S}$ there is a $t \in \mathcal{S}$ with $s \longrightarrow t$. $\mathcal{I} : \mathcal{P} \to 2^{\mathcal{S}}$ interprets the propositional constants from $\mathcal{P}$ in $\mathcal{T}$ respecting $\mathtt{tt}$, $\mathtt{ff}$ and complementation. $S_0 \subseteq \mathcal{S}$ is the set of all *starting* states.

A path through $\mathcal{T}$ is an infinite sequence $\pi = s_0 s_1 \ldots$, s.t. $s_0 \in S_0$ and for all $i \in \mathbb{N}$: $s_i \longrightarrow s_{i+1}$.

We write π^k for the k-th state of π, $Pos(\pi)$ for the set of states in π, and $Pos^k(\pi)$ for $\{\pi^i \in Pos(\pi) \mid i \leq k\}$.

Formulas of μTL are interpreted over a path $\pi = s_0 s_1 \ldots$ of an LTS $\mathcal{T}$. Free variables are interpreted using an *environment* $\rho : \mathcal{V} \to 2^{Pos(\pi)}$. With $\rho[X \mapsto T]$ we denote the function that maps X to T and behaves like ρ on all other arguments. Since π will always be derivable from the context we avoid mentioning it explicitly.

$$\begin{array}{lcl}
[\![q]\!]_\rho & := & \mathcal{I}(q) \\
[\![X]\!]_\rho & := & \rho(X) \\
[\![\varphi \vee \psi]\!]_\rho & := & [\![\varphi]\!]_\rho \cup [\![\psi]\!]_\rho \\
[\![\varphi \wedge \psi]\!]_\rho & := & [\![\varphi]\!]_\rho \cap [\![\psi]\!]_\rho \\
[\![\bigcirc \varphi]\!]_\rho & := & \{\pi^k \mid \pi^{k+1} \in [\![\varphi]\!]_\rho\} \\
[\![\mu X.\varphi]\!]_\rho & := & \bigcap \{T \subseteq Pos(\pi) \mid [\![\varphi]\!]_{\rho[X \mapsto T]} \subseteq T\} \\
[\![\nu X.\varphi]\!]_\rho & := & \bigcup \{T \subseteq Pos(\pi) \mid T \subseteq [\![\varphi]\!]_{\rho[X \mapsto T]}\}
\end{array}$$

We write $\pi^k \models_\rho \varphi$ if $\pi^k \in [\![\varphi]\!]_\rho$. If φ is closed, i.e., it does not contain any free variables we write $\pi^k \models \varphi$ instead. Finally, we write $\pi \models \varphi$ if $\pi^0 \in [\![\varphi]\!]$.

Lemma 2.1 *For every closed $\varphi \in \mu$TL, there is a closed $\overline{\varphi} \in \mu$TL s.t. for all paths π of all LTSs $\mathcal{T}$: $\pi \models \varphi$ iff $\pi \not\models \overline{\varphi}$.*

Proof. The complement φ' can inductively be constructed using complementation closure of atomic propositions, deMorgan's laws and the rules $\overline{\overline{\psi}} := \psi$, $\overline{\bigcirc \psi} := \bigcirc \overline{\psi}$, $\overline{\mu X.\psi(X)} := \nu X.\overline{\psi(\overline{X})}$, and $\overline{\nu X.\psi(X)} := \mu X.\overline{\psi(\overline{X})}$. □

We also allow ourselves to write $\neg\varphi$ instead of $\overline{\varphi}$.

Approximants of a formula $\sigma X.\varphi$ w.r.t. a linear time structure π and an environment $\rho : \mathcal{V} \to 2^{Pos(\pi)}$ are defined for every $i \in \mathbb{N}$ as usual:

$$X^0_\rho \;:=\; \begin{cases} \emptyset & \text{for } \sigma = \mu \\ Pos(\pi) & \text{for } \sigma = \nu \end{cases} \qquad\qquad X^{i+1}_\rho \;=\; [\![\varphi]\!]_{\rho[X \mapsto X^i_\rho]}$$

The following is a standard results about fixpoint logics. It follows immediately from the Knaster-Tarski Theorem and the fact that the semantics of a formula with a free variable is a monotone function on the subset lattice of states on a path.

Lemma 2.2 *For all $\varphi \in \mu$TL and environment ρ we have:*

$$[\![\mu X.\varphi]\!]_\rho \;\equiv\; \bigcup_{i \in \mathbb{N}} X^i_\rho \;, \qquad [\![\nu X.\varphi]\!]_\rho \;\equiv\; \bigcap_{i \in \mathbb{N}} X^i_\rho$$

We say that X depends on Y in φ, written $Y \prec_\varphi X$, if Y is free in $fp_\varphi(X)$. We write $\leq_\varphi$ for the reflexive-transitive closure of $\prec_\varphi$. The alternation depth $ad(\varphi)$ of φ is n if there is a maximal chain $X_0 \leq_\varphi \ldots \leq_\varphi X_n$ with consecutive variables having different fixpoint types. Let $\mu\mathrm{TL}^k := \{\varphi \in \mu\mathrm{TL} \mid ad(\varphi) \leq k\}$.

Proposition 2.3 [16,8] *Every closed $\varphi \in \mu\mathrm{TL}$ is equivalent to a closed $\varphi' \in \mu\mathrm{TL}^0$.*

The translation presented in [8] from a μTL formula φ yields a $\mu\mathrm{TL}^0$ formula of size $O(|\varphi| \cdot 2^{4 \cdot ad(\varphi)})$.

2.2 Symbolic Representations

Propositional Logic over a set $\mathcal{V}$ of propositional variables is the closure of $\mathcal{V}$ under the boolean connectives $\neg$, $\vee$, and consequently also $\wedge$, $\rightarrow$, etc. Here we assume a finite LTS $\mathcal{T} = (\mathcal{S}, \longrightarrow, \mathcal{I}, S_0)$ to be given symbolically, i.e., by propositional formulas

- $f_{\text{start}} : \mathbb{B}^n \rightarrow \mathbb{B}$ with $f_{\text{start}}(\bar{x}) = \mathtt{tt}$ iff $\bar{x} \in S_0$,
- $f_q : \mathbb{B}^n \rightarrow \mathbb{B}$ for every $q \in \mathcal{P}$ with $f_q(\bar{x}) = \mathtt{tt}$ iff $\bar{x} \in \mathcal{I}(q)$,
- $f_{\text{trans}} : \mathbb{B}^{2 \cdot n} \rightarrow \mathbb{B}$ with $f_{\text{trans}}(\bar{x}, \bar{y}) = \mathtt{tt}$ iff $\bar{x} \longrightarrow \bar{y}$.

where $n := \lceil \log |\mathcal{S}| \rceil$. I.e. every state is identified by a unique number in binary coding.

A formula is in conjunctive normal form (CNF), if it is of the form $C_1 \wedge \ldots \wedge C_m$, where each C_i is a clause, i.e., a disjunction $a_1 \vee \ldots \vee a_k$ of literals a_j, i.e., each a_j is a variable x or negated variable $\neg x$.

Most SAT solvers expect that the input formula is given in CNF. Our translation as defined below produces arbitrary formulas, but it is well-known that such formulas can be translated into CNF with only a linear blow-up in size and a linear number of additional variables.

3 μTL vs. LTL

Formulas of LTL are built from atomic propositions using the boolean operators $\wedge$, $\vee$ and $\neg$, as well as the temporal operators $\bigcirc$ (next) and $\mathtt{U}$ (until) with their usual semantics [10].

Lemma 3.1 *Every LTL-definable property is also definable in $\mu\mathrm{TL}^0$.*

Proof. The translation is easy and well-known. The only interesting case is that of the *until* operator which is translated as $\varphi \mathtt{U} \psi \equiv \mu X.\psi \vee (\varphi \wedge \bigcirc X)$. $\square$

It follows that μTL model checking over labelled transition systems is also PSPACE-hard [11] where the size of the input is the number of states in explicit representation. In fact, it is also PSPACE-complete [15].

Proposition 3.2 [1] *A language is ω-regular iff it is μTL-definable.*

Together with Proposition 2.3 we obtain that $\mu\mathrm{TL}^0$ is already capable of defining all ω-regular properties.

In the following, we will give a few examples of properties that are either μTL- but not LTL-definable, or that can be written down more succinctly in μTL.

Example 1 "Formula ψ holds on every even state of a path" is not LTL-definable, but can be expressed in μTL:

$$\varphi_{\mathrm{even}} \;:=\; \nu X.\psi \wedge \bigcirc\bigcirc X$$

Example 2 Suppose we have a set $Q = \{q_0, \ldots, q_{n-1}\}$ of atomic propositions and require them to occur repeatedly in this order. This can be done in μTL with the following formula of size linear in n.

$$\varphi \;:=\; \nu X.q_0 \wedge \bigcirc(q_1 \wedge \bigcirc(q_2 \wedge \ldots \bigcirc(q_n \wedge \bigcirc X)\ldots))$$

The property is still star-free, hence, LTL definable. But note that propositions do not exclude each other. Thus, an equivalent LTL formula would have to assert the label of the next state in accordance with the labels of the last n states – for every starting point in the order $q_0, \ldots, q_{n-1}$. Hence, its size would be quadratic in n.

Example 3 The next formula describes the capacity property of a bounded message buffer of size n. A word $w \in \{push, pop, nop\}^\omega$ satisfies β_n if for every prefix v of w, the difference between the numbers of occurrences of *push* and *pop* in v is between 0 and n. This is also a star-free property, but for growing n it occurs arbitrarily high in the dot-depth hierarchy of star-free languages [14], and thus it is notoriously hard to formalize in LTL. The formula β_n is φ_0, where φ_i is inductively defined as follows.

$$\begin{aligned}
\varphi_0 &:= \nu X_0.(push \rightarrow \bigcirc\varphi_1) \wedge \neg pop \wedge (nop \rightarrow \bigcirc X_0)\\
\varphi_i &:= \nu X_i.(push \rightarrow \bigcirc\varphi_{i+1}) \wedge (pop \rightarrow \bigcirc X_{i-1}) \wedge (nop \rightarrow \bigcirc X_i) \quad \text{if } 1 \leq i < n\\
\varphi_n &:= \nu X_n.\neg push \wedge (pop \rightarrow \bigcirc X_{n-1}) \wedge (nop \rightarrow \bigcirc X_n)
\end{aligned}$$

The size of β_n is obviously linear in n, whereas only exponential size LTL formulas specifying this property are known [12].

4 A Bounded Semantics for μTL

Assume an LTS $\mathcal{T} = (\mathcal{S}, \longrightarrow, \mathcal{I}, S_0)$ to be fixed and of finite size. Every path through $\mathcal{T}$ starting with a state in S_0 induces a linear time structure π.

Definition 1 *A path π of $\mathcal{T}$ is called a (k,ℓ)-*loop *for $\ell \leq k \in \mathbb{N}$ if $\pi^{k+1+i} = \pi^{\ell+i}$ for all $i \in \mathbb{N}$.*

Note that if φ is satisfied by a path of a finite transition system ($|\mathcal{S}| < \infty$), then it is already satisfied by a path which is a (k, ℓ)-loop for some $\ell \leq k \leq |\mathcal{S}|$. This is a consequence of Proposition 3.2.

Definition 2 Given a $k \in \mathbb{N}$, a path π of $\mathcal{T}$ and an environment $\rho : \mathcal{V} \to Pos(\pi)$, we define the k-bounded semantics $[\![\varphi]\!]^k_\rho$ by distinguishing two cases:

Case 1, π is a (k, ℓ)-loop for some $\ell \leq k$: Then the bounded semantics does not differ from the unbounded semantics of Section 2, i.e. we define

$$[\![\varphi]\!]^k_\rho \;:=\; [\![\varphi]\!]_\rho$$

Case 2, π is not a (k, ℓ)-loop for any $\ell \leq k$: Then we define

$$\begin{array}{lcl}
[\![q]\!]^k_\rho & := & \mathcal{I}(q) \cap Pos^k(\pi) \\
[\![X]\!]^k_\rho & = & \rho(X) \cap Pos^k(\pi) \\
[\![\varphi \vee \psi]\!]^k_\rho & := & [\![\varphi]\!]^k_\rho \cup [\![\psi]\!]^k_\rho \\
[\![\varphi \wedge \psi]\!]^k_\rho & := & [\![\varphi]\!]^k_\rho \cap [\![\psi]\!]^k_\rho \\
[\![\bigcirc \varphi]\!]^k_\rho & := & \{\pi^i \mid i < k \text{ and } \pi^{i+1} \in [\![\varphi]\!]^k_\rho\} \\
[\![\mu X.\varphi]\!]^k_\rho & := & \bigcap \{T \subseteq Pos^k(\pi) \text{ and } [\![\varphi]\!]^k_{\rho[X \mapsto T]} \subseteq T\} \\
[\![\nu X.\varphi]\!]^k_\rho & := & \emptyset
\end{array}$$

As for the unbounded case, we define *bounded approximants* for the iterative evaluation of the bounded semantics of fixpoint formulas.

Definition 3 Bounded approximants *for least fixpoint formulas $\mu X.\varphi$, a $k \in \mathbb{N}$, a path π and environment ρ are defined for all $i \in \mathbb{N}$ as*

$$X^{k,0}_\rho \;:=\; \emptyset \;, \quad X^{k,i+1}_\rho \;:=\; [\![\varphi]\!]^k_{\rho[X \mapsto X^{k,i}_\rho]}$$

For greatest fixpoint formulas, bounded approximants depend on the type of the underlying path. If π is a (k, ℓ)-loop for some $\ell \leq k$ then we define

$$X^{k,0}_\rho \;:=\; Pos^k(\pi) \;, \quad X^{k,i+1}_\rho \;:=\; [\![\varphi]\!]^k_{\rho[X \mapsto X^{k,i}_\rho]}$$

Otherwise, we set $X^{k,i}_\rho := \emptyset$ for all $i \in \mathbb{N}$.

The following lemmas form the basis for the correctness of the reduction in the next section. Lemma 4.1 expresses the monotonicity of the bounded semantics, and Lemma 4.2 states that the bounded approximants really approximate the bounded semantics. They are proved by simultaneous induction on the structure of μTL formulas, in a way similar to the corresponding statements for the unbounded semantics.

Lemma 4.1 *For all $k \in \mathbb{N}$, all $X \in \mathcal{V}$, all $\varphi \in \mu\mathrm{TL}$, all paths π, all environments ρ and all $P \subseteq Q \subseteq Pos^k(\pi)$ we have: $[\![\varphi]\!]^k_{\rho[X \mapsto P]} \subseteq [\![\varphi]\!]^k_{\rho[X \mapsto Q]}$.*

Lemma 4.2 *For all $k \in \mathbb{N}$, all $X \in \mathcal{V}$, all environments ρ, all $\varphi \in \mu\mathrm{TL}$ and all paths π we have: $[\![\mu X.\varphi]\!]^k_\rho = \bigcup_{i\in\mathbb{N}} X^{k,i}_\rho$ and $[\![\nu X.\varphi]\!]^k_\rho = \bigcap_{i\in\mathbb{N}} X^{k,i}_\rho$.*

The following lemma states that the bounded semantics is an under-approximation of the unbounded semantics. This entails that any counterexample found by bounded model checking is an actual counterexample to the checked specification.

Lemma 4.3 *For all $\varphi \in \mu\mathrm{TL}$, all environments ρ, all $k \in \mathbb{N}$ and all paths π we have: $[\![\varphi]\!]^k_\rho \subseteq [\![\varphi]\!]_\rho$.*

Proof. The only interesting case is the one of φ being $\mu X.\psi$, and the path π is not a (k,ℓ)-loop for any ℓ. For this case, we prove by a side induction on i that $X^{k,i}_\rho \subseteq X^i_\rho$ for all $i \in \mathbb{N}$, from which the lemma follows by Lemmas 4.2 and 2.2. The induction basis for the claim is trivial. For the induction step, note that

$$X^{k,i+1}_\rho \;=\; [\![\psi]\!]^k_{\rho[X\mapsto X^{k,i}]} \;\subseteq\; [\![\psi]\!]_{\rho[X\mapsto X^{k,i}_\rho]} \;\subseteq\; [\![\psi]\!]_{\rho[X\mapsto X^i_\rho]} \;=\; X^{i+1}_\rho$$

where the first inclusion follows by the main induction hypothesis, and the second one by the side induction hypothesis and monotonicity. □

The next lemma shows that the bounded semantics is monotone in the bound k. This entails that by increasing the bound, one does not lose any counterexamples that would have been found with a smaller bound.

Lemma 4.4 *For all $k \in \mathbb{N}$, all $\varphi \in \mu\mathrm{TL}$, all environments ρ and all paths π we have: $[\![\varphi]\!]^k_\rho \subseteq [\![\varphi]\!]^{k+1}_\rho$.*

Proof. The only non-trivial case is the one of π not being a $(k+1,\ell)$-loop for any $\ell \leq k+1$. Again, the proof is by induction on φ. The only interesting case is $\varphi = \mu X.\psi$, where we prove by side induction on i that $X^{k,i}_\rho \subseteq X^{k+1,i}_\rho$, from which the claim follows by Lemma 4.2. For $i = 0$ this is trivial again, and the inductive step follows by

$$X^{k,i+1}_\rho \;=\; [\![\psi]\!]^k_{\rho[X\mapsto X^{k,i}_\rho]} \;\subseteq\; [\![\psi]\!]^{k+1}_{\rho[X\mapsto X^{k,i}_\rho]} \;\subseteq\; [\![\psi]\!]^{k+1}_{\rho[X\mapsto X^{k+1,i}_\rho]} \;=\; X^{k+1,i+1}_\rho$$

where the first inclusion follows by the main induction hypothesis, and the second one by the side induction hypothesis and Lemma 4.1. □

Lemma 4.5 *For any $\sigma \in \{\mu, \nu\}$, any formula φ, environment ρ, and $k \in \mathbb{N}$ we have $[\![\sigma X.\varphi]\!]^k_\rho = X^{k,k}_\rho$.*

Proof. This is a consequence of Lemma 4.2, since the chain of bounded approximants must become stationary after at most k steps. The reason is that all bounded approximants are subsets of $Pos^k(\pi)$, and $|Pos^k(\pi)| = k$. □

By use of this lemma, for a fixpoint formula φ containing m nested fixpoint operators, $[\![\varphi]\!]^k$ can be computed in k^m steps. For alternation-free formulas in

μTL^0 one can do better. We present the construction for least fixpoints, for greatest fixpoints it is completely analogous.

Let $\varphi = \mu X.\psi$ be a closed fixpoint formula, and let $X = X_1, \ldots, X_r$ be those variables in φ that depend on X, i.e., $X \leq_\varphi X_i$ for $i = 1, \ldots, r$. Since $\varphi \in \mu$TL^0, all the variables X_i are of type μ. Now φ is transformed into a system of equations

$$\begin{array}{rcl} X_1 & = & \psi_1(X_1, \ldots, X_r) \\ & \vdots & \\ X_r & = & \psi_r(X_1, \ldots, X_r) \end{array} \tag{1}$$

where the formulas ψ_j contain no fixpoint subformulas that depend on the variables $X_1, \ldots, X_r$, i.e., every fixpoint subformula of $\psi_j(X_1, \ldots, X_r)$ is a subformula of some closed fixpoint subformula of $\psi_j(X_1, \ldots, X_r)$. The translation is obtained as follows: let

$$fp_\varphi(X_i) \quad = \quad \mu X_i.\psi_i(X_1, \ldots, X_i, \mu Y_1.\theta_1, \ldots, \mu Y_s.\theta_s)$$

containing free variables among $X_1, \ldots, X_{i-1}$, where the subformulas $\mu Y_j.\theta_j$ for Y_j among $X_{i+1}, \ldots, X_r$ are those outermost fixpoint subformulas of ψ_i that contain any free variables from $X_1, \ldots, X_i$. This formula yields the equation $X_i \;=\; \psi_i(X_1, \ldots, X_i, Y_1, \ldots, Y_s)$ in (1).

For the system of equations (1), the bounded simultaneous approximants $X_i^{k,(j)}$ for $1 \leq i \leq r$ and $j \in \mathbb{N}$ are inductively defined as follows:

$$X_i^{k,(0)} \;=\; \emptyset \qquad\qquad X_i^{k,(j+1)} \;=\; [\![\psi_i(X_1, \ldots, X_r)]\!]^k_{\rho_j} \tag{2}$$

where ρ_j is the environment that maps each variable X_h to $X_h^{k,(j)}$, for $1 \leq h \leq r$.

Lemma 4.6 *For a closed fixpoint formula $\mu X.\varphi$ as above, $[\![\mu X.\varphi]\!]^k = X_1^{k,(kr)}$.*

Proof. The fixpoint of the simultaneous iteration (2) is the same as $[\![\mu X.\varphi]\!]^k$ by Békič' Theorem [2]. Moreover, (2) reaches its fixpoint after at most $k \cdot r$ iterations, since there are r subsets of $Pos^k(\pi)$ being computed, and in the worst case, in each iteration only one of the sets increases by one element. $\square$

5 The Reduction to SAT

For a transition system $\mathcal{T}$ with 2^n states, represented symbolically by boolean formulas as described in Section 2.2, a formula $\varphi \in \mu$TL^0 and a $k \in \mathbb{N}$ we define a boolean formula $\langle\!\langle \mathcal{T}, \varphi \rangle\!\rangle^k$ in the following variables:

- the *path variables* $\bar{s}_i = s_{i,1}, \ldots, s_{i,n}$ for $1 \leq i \leq k$, coding the i-th state on a path.

- auxiliary variables $v(X)_i$ for every second-order variable X and $1 \leq i \leq k$. These variables will not occur in the final formula $\langle\!\langle\, \mathcal{T}, \varphi \,\rangle\!\rangle^k$, they are only used during the construction as placeholders for free variables in subformulas.
- the *approximant variables* $a(X,j)_i^k$ and $a(X,j)_i^{k,\ell}$ for every second-order variable X and $1 \leq i, \ell \leq k$ and $j \in \mathbb{N}$. These variables express that state i is in the bounded approximant $X^{k,(j)}$.

First, we define a formula $\langle\!\langle\, \mathcal{T} \,\rangle\!\rangle^k$ saying that the path variables $\bar{s}_1, \ldots, \bar{s}_k$ actually encode a path in $\mathcal{T}$ by

$$\langle\!\langle\, \mathcal{T} \,\rangle\!\rangle^k \;:=\; f_{\mathrm{start}}(\bar{s}_1) \wedge \bigwedge_{i=1}^{k-1} f_{\mathrm{trans}}(\bar{s}_i, \bar{s}_{i+1}) \;.$$

Next, as usual we define formulas to distinguish between the cases where the path is a (k,ℓ)-loop for $\ell \leq k$, and where it is not, by

$$\mathrm{Loop}^{k,\ell} \;:=\; f_{\mathrm{trans}}(\bar{s}_k, \bar{s}_\ell) \qquad \neg\mathrm{Loop}^k \;:=\; \bigwedge_{i=1}^{k} \neg\mathrm{Loop}^{k,i}$$

and using these, we define the translation by

$$\langle\!\langle\, \mathcal{T}, \varphi \,\rangle\!\rangle^k \;:=\; \langle\!\langle\, \mathcal{T} \,\rangle\!\rangle^k \wedge \Big(\big(\neg\mathrm{Loop}^k \wedge \langle\!\langle\, \varphi \,\rangle\!\rangle^k\big) \vee \bigvee_{\ell=1}^{k} \big(\mathrm{Loop}^{k,\ell} \wedge \langle\!\langle\, \varphi \,\rangle\!\rangle^{k,\ell}\big) \Big) \;.$$

The formula $\langle\!\langle\, \varphi \,\rangle\!\rangle^k$ that actually encodes φ in the case of a non-loop is defined as $\langle\!\langle\, \varphi \,\rangle\!\rangle_1^k \wedge \mathit{Defs}(\varphi)^k$, where the formulas $\langle\!\langle\, \psi \,\rangle\!\rangle_i^k$ for subformulas ψ of φ and $1 \leq i \leq k$ express that the i^{th} state satisfies ψ. For formulas without fixpoint operators, these are inductively defined by:

$$\begin{aligned}
\langle\!\langle\, q \,\rangle\!\rangle_i^k &\;:=\; f_q(\bar{s}_i) \\
\langle\!\langle\, X \,\rangle\!\rangle_i^k &\;:=\; v(X)_i \\
\langle\!\langle\, \varphi \vee \psi \,\rangle\!\rangle_i^k &\;:=\; \langle\!\langle\, \varphi \,\rangle\!\rangle_i^k \vee \langle\!\langle\, \psi \,\rangle\!\rangle_i^k \\
\langle\!\langle\, \varphi \wedge \psi \,\rangle\!\rangle_i^k &\;:=\; \langle\!\langle\, \varphi \,\rangle\!\rangle_i^k \wedge \langle\!\langle\, \psi \,\rangle\!\rangle_i^k \\
\langle\!\langle\, \bigcirc \varphi \,\rangle\!\rangle_i^k &\;:=\; \begin{cases} \langle\!\langle\, \varphi \,\rangle\!\rangle_{i+1}^k & \text{if } i < k \\ \mathtt{ff} & \text{otherwise} \end{cases}
\end{aligned}$$

Next, we define the translation for a closed greatest fixpoint formula as the constant $\mathtt{ff}$,

$$\langle\!\langle\, \nu X.\psi \,\rangle\!\rangle_i^k \;:=\; \mathtt{ff} \;,$$

and for a closed least fixpoint formula as the approximant variable

$$\langle\!\langle\, \mu X.\psi \,\rangle\!\rangle_i^k \;:=\; a(X, kr)_i^k \;,$$

where r is the number of second-order variables Y in $\mu X.\psi$ with $X \leq_\varphi Y$.

Note that in a fixpoint formula, the bound variable can occur several times. Therefore a straightforward translation of the approximants by syntactic unfolding would lead to an exponential blowup. To prevent this, we use the approximant variables to abbreviate the approximants, and the formula $\mathit{Defs}(\varphi)^k$ takes care of their proper interpretation. It is defined as the conjunction of the defining formulas $\mathit{Def}(\psi)^k$, over all subformulas ψ of φ that are closed least fixpoint formulas.

Another exponential blowup would occur if nested fixpoints were translated straightforwardly inside out, since the unfolding of a formula with m nested fixpoints would produce k^m subformulas. Therefore we use the transformation of a closed least fixpoint subformula ψ into a system of r equations (1), as described at the end of Section 4:

$$\begin{aligned} X_1 &= \psi_1(X_1, \ldots, X_r) \\ &\vdots \\ X_r &= \psi_r(X_1, \ldots, X_r) \end{aligned}$$

The formula $\mathit{Def}(\psi)^k$ describes the evaluation of this system of equations by the simultaneous approximants (2) by giving definitions for the corresponding approximant variables. I.e., $\mathit{Def}(\psi)^k$ is the conjunction of the equivalences [1] $a(X,s)^k_i \leftrightarrow F(X_j,s)^k_i$ over all $1 \leq j \leq r$, $1 \leq s \leq kr$ and $1 \leq i \leq k$, where

- $F(X_j,1)^k_i$ is the translation $\langle\!\langle\, \psi_j(X_1,\ldots,X_r) \,\rangle\!\rangle^k_i$ with the variables $v(X_h)^k_g$ for $1 \leq h \leq r$ and $1 \leq g \leq k$ replaced by $\mathtt{ff}$, and
- $F(X_j,s)^k_i$ for $s > 1$ is $\langle\!\langle\, \psi_j(X_1,\ldots,X_r) \,\rangle\!\rangle^k_i$ with the variables $v(X_h)^k_g$ replaced by $a(X_h, s-1)^k_g$, for $1 \leq h \leq r$ and $1 \leq g \leq k$.

Similarly, the translation $\langle\!\langle\, \varphi \,\rangle\!\rangle^{k,\ell}$ of φ in the case of a loop is defined as $\langle\!\langle\, \varphi \,\rangle\!\rangle^{k,\ell}_1 \wedge \mathit{Defs}(\varphi)^{k,\ell}$, where the inductive definition of the formulas $\langle\!\langle\, \psi \,\rangle\!\rangle^{k,\ell}_i$ differs only in the clause for $\bigcirc\psi$, which becomes:

$$\langle\!\langle\, \bigcirc\varphi \,\rangle\!\rangle^{k,\ell}_i \;:=\; \begin{cases} \langle\!\langle\, \varphi \,\rangle\!\rangle^{k,\ell}_{i+1} & \text{if } i < k \\ \langle\!\langle\, \varphi \,\rangle\!\rangle^{k,\ell}_{\ell} & \text{otherwise} \end{cases}$$

For both closed least and greatest fixpoint formulas we now define the translation by

$$\langle\!\langle\, \sigma X.\psi \,\rangle\!\rangle^{k,\ell}_i \;:=\; a(X,kr)^{k,\ell}_i \;,$$

where like above, r is the number of second-order variables Y in $\sigma X.\psi$ with $X \leq_\varphi Y$.

The formula $\mathit{Defs}(\varphi)^{k,\ell}$ is the conjunction of the formulas $\mathit{Def}(\psi)^{k,\ell}$ over all closed least and greatest fixpoint subformulas of φ. For such a subfor-

[1] If the formulas are transformed into CNF, these equivalences need not be written, but are implicitly produced by the transformation. One only needs to identify the variable $a(X,s)^k_i$ with the new variable abbreviating the formula $F(X_j,s)^k_i$.

n	Var	Cls	Red	SAT	n	Var	Cls	Red	SAT
22	6 k	42 k	0.24	0.09	102	143 k	1322 k	91.29	22.05
32	13 k	97 k	0.86	1.87	112	173 k	1597 k	124.44	213.27
42	23 k	191 k	2.88	4.49	122	207 k	1915 k	178.86	462.75
52	36 k	298 k	6.29	26.83	132	242 k	2438 k	253.42	421.27
62	52 k	435 k	12.13	3.00	142	280 k	2831 k	338.51	1167.54
72	71 k	647 k	21.10	21.01	152	320 k	3229 k	469.33	630.07
82	92 k	847 k	35.09	107.17	162	366 k	3699 k	583.69	10.78
92	116 k	1059 k	54.94	138.97	172	409 k	4128 k	805.48	865.44

Fig. 1. The even b / odd c example.

mula, written as an equation system in the variables $X_1, \ldots, X_r$, the formula $\mathit{Def}(\psi)^{k,\ell}$ is defined exactly as $\mathit{Def}(\psi)^k$ above, only that for a variable of type ν, the defining formulas for the first approximant variables become $a(X_j,1)_i^{k,\ell} \leftrightarrow F(X_j,1)_i^{k,\ell}$, where in this case $F(X_j,1)_i^{k,\ell}$ is the translation $\langle\!\langle\, \psi_j(X_1,\ldots,X_r) \,\rangle\!\rangle_i^{k,\ell}$ with the variables $v(X_h)_g^{k,\ell}$ for $1 \leq h \leq r$ and $1 \leq g \leq k$ replaced by $\mathtt{tt}$.

The number of variables in and the size of the translation is measured in the numbers n, k, the size of the input formula s and the number of second-order variables v. They are easily estimated, and are as follows:

Proposition 5.1 *The formula $\langle\!\langle\, \mathcal{T}, \varphi \,\rangle\!\rangle^k$ contains $O(v^2k^3+kn)$ variables, and is of size $O(v^2k^3sn)$.*

Even though the number of variables produced by our translation is rather large, in particular regarding the cubic dependence on k, this might not be too problematic, since the approximant variables occur in $k+1$ disjoint parts of the formulas, each containing only $O(k^2)$ of them.

Finally, we can easily observe the correctness of our translation, which is obvious from the definition for all cases except the fixpoint formulas. For the latter the correctness follows from Lemma 4.6.

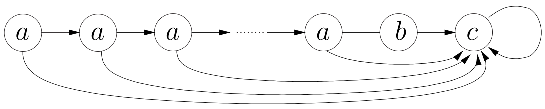
Proposition 5.2 *The formula $\langle\!\langle\, \mathcal{T}, \varphi \,\rangle\!\rangle^k$ is satisfiable iff there is a path π in $\mathcal{T}$ starting at an initial state, and for which $\pi^0 \in [\![\varphi]\!]^k$.*

6 Experimental Results

The algorithm presented here is implemented as part of the verification tool μ-Sabre that is being developed at LMU Munich. The program is implemented in the lazy functional language Haskell using the Glasgow Haskell Compiler 6.2.2, with the exception of a small part of the program, dealing with linking of the SAT solver, that was implemented in C. The SAT solver used is version 2004.5.13 of zChaff [9].

The tests were carried out on a machine with two Intel® Xeon™ 2.4 GHz processors and 4GB of RAM. The second processor remained unused.

In a first test series we consider the property "there is a path with a b at an even position and a c at an odd position" on a family $\{\mathcal{T}_n \mid n \in \mathbb{N}\}$ of transition systems, s.t. $\mathcal{T}_n$ has got n states. The transitions between these states and their labels are as follows.



The only starting state is the leftmost. The property is written in μTL as $(\mu X.b \vee \bigcirc\bigcirc X) \wedge (\mu Y.\bigcirc c \vee \bigcirc\bigcirc Y)$. It may not be an interesting property but we include it here because it cannot be formalised in LTL, c.f. Example 1.

The running times of our reduction (Red) and the SAT solver (SAT) are presented in Figure 1. The time unit is seconds. We only present satisfiable instances, i.e. those of even n. The table also contains the number of propositional variables (Var) and the number of clauses (Cls) in the resulting formulas – truncated down to multiples of 1000 in order to save space.

Our other tests use a transition system $\mathcal{B}_n$ modelling a message buffer of size n, holding messages that are single bits. Every state in $\mathcal{B}_n$ has $2n+3$ bits: The first two are the opcode for the next operation. The third bit is the output of the previous operation; its value is only specified in states following a *pop* operation. The remaining $2n$ bits represent the n buffer cells, each cell being represented by one bit indicating whether the cell is occupied, and the other being the value stored in the cell. The value of the second bit is unspecified for unoccupied cells.

The boolean formulas f_{start} and f_{trans} are hand-coded, with f_{start} saying that the buffer is initially empty, and f_{trans} specifying the changes in the buffer depending on the opcode, e.g., one disjunct of $f_{\text{trans}}(x, y)$ is

$$\neg x_1 \wedge \neg x_2 \wedge \bigwedge_{4 \le i \le 2n+3} (x_i \leftrightarrow y_i)$$

stating that a *nop* (having opcode 00) does not change the buffer content.

We test the property $\neg\beta_{n-1}$ of Example 3 on $\mathcal{B}_n$ in order to have a satisfiable example. The minimal counterexample showing that β_{n-1} is violated is a sequence of n *push* operations, thus in our second experiment we tested whether $\mathcal{B}_n \models_n \neg\beta_n$, for various n. The results are shown in Figure 2. Again, the time unit is seconds.

In the third experiment, in order to see the dependence of the performance on the bound k, we test $\mathcal{B}_n \models_k \neg\beta_{n-1}$ for various values of $k \geq n$, for fixed $n = 12$. The results are presented in Figure 3.

The example formula β_n was chosen for two reasons: First, as mentioned above, the property expressed can probably not easily and succinctly be stated in LTL. Second, it fully utilizes the syntactic possibilities of alternation-free μTL, since β_n has n nested fixpoints, and, due to the presence of the *nop* operation, each bound variable (except for X_n) occurs twice.

n	Var	Cls	Red	SAT
6	15 k	55 k	0.35	0.24
7	28 k	98 k	0.75	3.47
8	48 k	163 k	1.68	2.67
9	75 k	256 k	3.56	4.71
10	114 k	384 k	7.31	11.18
11	165 k	554 k	14.36	13.34
12	231 k	775 k	27.20	27.16
13	316 k	1056 k	49.56	39.64

n	Var	Cls	Red	SAT
14	423 k	1407 k	89.46	56.11
15	554 k	1840 k	158.40	95.49
16	713 k	2364 k	253.85	188.07
17	905 k	2994 k	392.49	146.14
18	1133 k	3741 k	608.33	157.17
19	1401 k	4620 k	947.79	293.46
20	1715 k	5646 k	1362.74	226.30
21	2078 k	6833 k	2072.00	810.71

Fig. 2. The buffer example with $k = n$

k	Var	Cls	Red	SAT
12	231 k	775 k	27.09	27.02
14	309 k	1030 k	49.92	41.57
16	398 k	1321 k	86.66	42.10
18	498 k	1648 k	145.22	73.22
20	610 k	2011 k	218.77	66.15
22	732 k	2409 k	308.51	178.23
24	866 k	2843 k	425.32	380.42

k	Var	Cls	Red	SAT
26	1010 k	3312 k	590.76	34.09
28	1166 k	3818 k	780.84	182.86
30	1333 k	4359 k	1036.56	233.37
32	1511 k	4935 k	1317.30	216.08
34	1701 k	5548 k	1659.06	161.38
36	1901 k	6196 k	2171.02	718.25
38	2113 k	6880 k	2929.20	409.74

Fig. 3. The buffer example with $n = 12$.

References

[1] H. Barringer, R. Kuiper, and A. Pnueli. A really abstract concurrent model and its temporal logic. In *Conf. Record of the 13th Annual ACM Symp. on Principles of Programming Languages, POPL'86*, pages 173–183. ACM, 1986.

[2] H. Békič. *Programming Languages and Their Definition, Selected Papers*, volume 177 of *LNCS*. Springer, 1984.

[3] A. Biere, A. Cimatti, E. M. Clarke, and Y. Zhu. Symbolic model checking without BDDs. In R. Cleaveland, editor, *Proc. 5th Int. Conf. on Tools and Algorithms for the Analysis and Construction of Systems, TACAS'99*, volume 1579 of *LNCS*, Amsterdam, NL, Mar. 1999.

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

[5] E. Clarke, D. Kroening, O. Strichman, and J. Ouaknine. Completeness and complexity of bounded model checking. In *Proc. 5th Int. Conf. on Verification, Model Checking, and Abstract Interpretation, VMCAI'04*, volume 2937 of *LNCS*, pages 85–96. Springer, 2004.

[6] D. Gabbay, A. Pnueli, S. Shelah, and J. Stavi. The temporal analysis of fairness. In *Proc. 7th Symp. on Principles of Programming Languages, POPL'80*, pages 163–173. ACM, Jan. 1980.

[7] H. W. Kamp. *On tense logic and the theory of order.* PhD thesis, Univ. of California, 1968.

[8] M. Lange. Weak automata for the linear time μ-calculus. In R. Cousot, editor, *Proc. 6th Int. Conf. on Verification, Model Checking and Abstract Interpretation, VMCAI'05*, volume 3385 of *LNCS*, pages 267–281, 2005.

[9] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an efficient sat solver. In *Proceedings of the 38th Design Automation Conference, DAC 2001*, pages 530–535, 2001.

[10] A. Pnueli. The temporal logic of programs. In *Proc. 18th Symp. on Foundations of Computer Science, FOCS'77*, pages 46–57, Providence, RI, USA, Oct. 1977. IEEE.

[11] A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. *Journal of the Association for Computing Machinery*, 32(3):733–749, July 1985.

[12] A. P. Sistla, E. M. Clarke, N. Francez, and A. R. Meyer. Can message buffers be axiomatized in linear temporal logic? *Information and Control*, 63(1-2):88–112, 1984.

[13] W. Thomas. Star-free regular sets of ω-sequences. *Information and Control*, 42(2):148–156, Aug. 1979.

[14] W. Thomas. A concatenation game and the dot-depth hierarchy. In E. Börger, editor, *Computation Theory and Logic*, volume 270 of *Lecture Notes in Computer Science*, pages 415–426. Springer, 1987.

[15] M. Y. Vardi. A temporal fixpoint calculus. In ACM, editor, *Proc. Conf. on Principles of Programming Languages, POPL'88*, pages 250–259, NY, USA, 1988. ACM Press.

[16] M. Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, Nov. 1994.