Local Model Checking Games for Fixed Point Logic with Chop

Martin Lange

Institut für Informatik
Ludwigs-Maximilian-Universität München
martin@dcs.ed.ac.uk

Abstract. The logic considered in this paper is FLC, fixed point logic with chop. It is an extension of modal μ-calculus $\mathcal{L}_\mu$ that is capable of defining non-regular properties which makes it interesting for verification purposes. Its model checking problem over finite transition systems is PSPACE-hard. We define games that characterise FLC's model checking problem over arbitrary transition systems. Over finite transition systems they can be used as a basis of a local model checker for FLC. I.e. the games allow the transition system to be constructed on-the-fly. On formulas with fixed alternation depth and so-called sequential depth deciding the winner of the games is PSPACE-complete. The best upper bound for the general case is EXPSPACE which can be improved to EXPTIME at the cost of losing the locality property. On $\mathcal{L}_\mu$ formulas the games behave equally well as the model checking games for $\mathcal{L}_\mu$, i.e. deciding the winner is in NP$\cap$co-NP.

1 Introduction

Modal and temporal logics are well established research areas in computer science, artificial intelligence, philosophy, etc. [2, 5, 11]. An important temporal logic is Kozen's modal μ-calculus $\mathcal{L}_\mu$ [8] because it contains almost all other propositional temporal logics. In fact, it is equi-expressive to the bisimulation invariant fragment of monadic second-order logic over transition graphs [7]. Therefore, properties expressed by $\mathcal{L}_\mu$ formulas are essentially "regular".

In [10], Müller-Olm introduced FLC, fixed point logic with chop, that extends $\mathcal{L}_\mu$ with sequential composition. He showed that the expressive power of FLC is strictly greater than $\mathcal{L}_\mu$ because FLC can define non-regular properties. Whereas the semantics of a modal μ-calculus formula is a subset of states of a transition system, the semantics of an FLC formula is a predicate transformer, a function from sets of states to sets of states.

In [9] it is shown that FLC can express certain non-regular properties that are very interesting for verification of protocols for example. The reason for this is FLC's restricted ability to count similar to that of context-free languages. As Müller-Olm showed, FLC is even able to express certain non-context-free properties due to the presence of boolean operators.

He also proved that, because of FLC's expressive power, its satisfiability problem as well as the model checking problem for BPA processes is undecidable. However, he notes that model checking finite transition systems is decidable. It can be done using the Tarski-Knaster Theorem [13] in a function lattice using fixed point approximants. In [9] it has been proved to be PSPACE-hard and a model checker for FLC that avoids explicit calculation of functions has been given. It builds tableaux for formulas of FLC and sets of states of a transition system. These tableaux admit global model checking, i.e. they require the entire underlying transition system to be present in the memory. In general, they result in a model checking procedure that needs time exponential in the size of the transition system and the size of the formula. The space needed is quadratic in the size of the transition system and the size of the formula.

Here we show that FLC model checking can be done locally. We define games similar to the model checking games for $\mathcal{L}_\mu$ from [12]. They work on single states instead of sets and allow the transition system to be built up on demand. The sets of states in the tableaux of [9] are not used explicitly, but are implicitly present as stacks of FLC formulas in the games of this paper. This results in a better complexity for fixed formulas, PSPACE for the games as opposed to EXPTIME for the tableaux. Moreover, if the alternation depth and the sequential depth, which is to be defined, of the input formula are fixed, then the games like the tableaux from [9] yield a PSPACE algorithm as well. Therefore, the games behave better than the tableaux, i.e. they are more suitable for verification purposes.

Furthermore, games are naturally equipped with the ability to show the user of a verification tool why a formula is not fulfilled. This is simply done by playing a game according to a winning strategy. It is not obviously clear how to show the non-existence of a successful tableaux.

It is well known that a game graph for a finite transition system and a modal μ-calculus formula is nothing else but an alternating tree automaton. No automata-based model checker for FLC is known so far. The games of this paper suggest that alternating push-down tree automata might be the right machinery for this.

For a more thorough discussion of examples of FLC formulas and properties of the logic see [10] and [9]. The rest of the paper is structured as follows. In section 2 we recall the syntax and semantics of FLC. The model checking games are defined and shown to be sound and complete in section 3. Finally, section 4 analyses the complexity of game-based FLC model checking.

2 Preliminaries

Let $\mathcal{P} = \{\mathtt{tt}, \mathtt{ff}, q, \overline{q}, \ldots\}$ be a set of propositional constants that is closed under complementation, $\mathcal{V} = \{Z, Y, \ldots\}$ a set of propositional variables, and $\mathcal{A} = \{a, b, \ldots\}$ a set of action names. A labelled *transition system* is a graph $\mathcal{T} = (\mathcal{S}, \{\stackrel{a}{\rightarrow} \mid a \in \mathcal{A}\}, L)$ where $\mathcal{S}$ is a set of states, $\stackrel{a}{\rightarrow}$ for each $a \in \mathcal{A}$ is a binary relation on states and $L : \mathcal{S} \rightarrow 2^{\mathcal{P}}$ labels the states such that, for all $s \in \mathcal{S} : q \in L(s)$

iff $\overline{q} \notin L(s)$, $\mathtt{tt} \in L(s)$, and $\mathtt{ff} \notin L(s)$. We will use infix notation $s \xrightarrow{a} t$ for transition relations.

Formulas of FLC are given by

$$\varphi \quad ::= \quad q \mid Z \mid \tau \mid \langle a \rangle \mid [a] \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \mu Z.\varphi \mid \nu Z.\varphi \mid \varphi;\varphi$$

where $q \in \mathcal{P}$, $Z \in \mathcal{V}$, and $a \in \mathcal{A}$.[1] We will write σ for μ or ν. To save brackets we introduce the convention that ; binds stronger than $\wedge$ which binds stronger than $\vee$. Formulas are assumed to be well named in the sense that each binder variable is distinct. Our main interest is with closed formulas, that do not have free variables, in which case there is a function $fp : \mathcal{V} \to \text{FLC}$ that maps each variable to its defining fixed point formula (that may contain free variables).

The set $Sub(\varphi)$ of subformulas of φ is defined as usual, with $Sub(\sigma Z.\psi) = \{\sigma Z.\psi\} \cup Sub(\psi)$. We say that Z depends on Y in φ, written $Z \prec_\varphi Y$, if Y occurs free in $fp(Z)$. We write $Z <_\varphi Y$ iff (Z, Y) is in the transitive closure of $\prec_\varphi$. The *alternation depth* of φ, $ad(\varphi)$, is the maximal k in a chain $Z_0 <_\varphi Z_1 <_\varphi \ldots <_\varphi Z_k$ of variables in φ s.t. Z_{i-1} and Z_i are of different fixed point types for $0 < i \leq k$.

The *tail* of a variable Z in a formula φ, tl_Z is a set consisting of those formulas that occur "behind" Z in $fp(Z)$ in φ. We assume that φ is derivable from the context and avoid to put it into the subscript of tl. In order to define it technically we use sequential composition for sets of formulas in a straightforward way: $\{\varphi_0, \ldots, \varphi_n\};\psi := \{\varphi_0;\psi, \ldots, \varphi_n;\psi\}$. We also use the eponymous function $tl_Z : Sub(\varphi) \to 2^{Sub(\varphi)}$ where

$$\begin{array}{llll}
tl_Z(q) & := \{q\} & tl_Z(\varphi \vee \psi) & := tl_Z(\varphi) \cup tl_Z(\psi) \\
tl_Z(\langle a \rangle) & := \{\langle a \rangle\} & tl_Z(\varphi \wedge \psi) & := tl_Z(\varphi) \cup tl_Z(\psi) \\
tl_Z([a]) & := \{[a]\} & tl_Z(\sigma Y.\psi) & := tl_Z(\psi) \\
tl_Z(\varphi;\psi) & := T_1 \cup T_2 & tl_Z(Y) & := \begin{cases} \{Y\} & \text{if } Y \neq Z \\ \{\tau\} & \text{o.w.} \end{cases}
\end{array}$$

with

$$T_1 := \begin{cases} tl_Z(\varphi);\psi & \text{if } Z \in Sub(\varphi) \\ \{\tau\} & \text{o.w.} \end{cases} \qquad T_2 := \begin{cases} tl_Z(\psi) & \text{if } Z \in Sub(\psi) \\ \{\tau\} & \text{o.w.} \end{cases}$$

The tail of Z in φ is simply calculated as $tl_Z := tl_Z(fp(Z))$.

Like the alternation depth of a formula φ, its *sequential depth* $sd(\varphi)$ is an important factor in the complexity of the model checking problem. Informally the sequential depth of a formula is the maximal number of times a variable is sequentially composed with itself. It is defined as $sd(\varphi) := max\{sd_Z(fp(Z)) \mid Z \in Sub(\varphi)\} - 1$ where

[1] In [10], τ is called `term`.

$$sd_Z(\varphi \vee \psi) := max\{sd_Z(\varphi), sd_Z(\psi)\} \qquad sd_Z(\varphi \wedge \psi) := max\{sd_Z(\varphi), sd_Z(\psi)\}$$
$$sd_Z(\varphi;\psi) := sd_Z(\varphi) + sd_Z(\psi) \qquad sd_Z(\sigma Y.\varphi) := sd_Z(\varphi)$$
$$sd_Z(\psi) := 0 \quad \text{if } \psi \in \{q, \tau, \langle a\rangle, [a]\} \qquad sd_Z(Y) := \begin{cases} 1 & \text{if } Y = Z \\ 0 & \text{o.w.} \end{cases}$$

Important syntactical fragments of FLC are those with fixed alternation and sequential depth.

$$\mathrm{FLC}^{k,n} := \{\varphi \in \mathrm{FLC} \mid ad(\varphi) \leq k, sd(\varphi) \leq n\}$$
$$\mathrm{FLC}^{k,\omega} := \bigcup_{n\in\mathbb{N}} \mathrm{FLC}^{k,n} \qquad \mathrm{FLC}^{\omega,n} := \bigcup_{k\in\mathbb{N}} \mathrm{FLC}^{k,n}$$

An *environment* $\rho : \mathcal{V} \to (2^{\mathcal{S}} \to 2^{\mathcal{S}})$ maps variables to monotone functions of sets to sets. $\rho[Z \mapsto f]$ is the function that maps Z to f and agrees with ρ on all other arguments. The semantics $[\![\cdot]\!]^{\mathcal{T}}_{\rho} : 2^{\mathcal{S}} \to 2^{\mathcal{S}}$ of an FLC formula, relative to $\mathcal{T}$ and ρ, is a monotone function on subsets of states with respect to the inclusion ordering on $2^{\mathcal{S}}$. These functions together with the partial order given by

$$f \sqsubseteq g \text{ iff } \forall X \subseteq \mathcal{S} : f(X) \subseteq g(X)$$

form a complete lattice with joins $\sqcup$ and meets $\sqcap$. By the Tarski-Knaster Theorem [13] the least and greatest fixed points of functionals $F : (2^{\mathcal{S}} \to 2^{\mathcal{S}}) \to (2^{\mathcal{S}} \to 2^{\mathcal{S}})$ exist. They are used to interpret fixed point formulas of FLC.

To simplify the notation we assume a transition system $\mathcal{T}$ to be fixed for the remainder of the paper, and drop it from the semantic brackets.

$$\begin{aligned}
[\![q]\!]_\rho &= \lambda X.\{s \in \mathcal{S} \mid q \in L(s)\} \\
[\![Z]\!]_\rho &= \rho(Z) \\
[\![\tau]\!]_\rho &= \lambda X.X \\
[\![\varphi \vee \psi]\!]_\rho &= \lambda X.[\![\varphi]\!]_\rho(X) \cup [\![\psi]\!]_\rho(X) \\
[\![\varphi \wedge \psi]\!]_\rho &= \lambda X.[\![\varphi]\!]_\rho(X) \cap [\![\psi]\!]_\rho(X) \\
[\![\langle a\rangle]\!]_\rho &= \lambda X.\{s \in \mathcal{S} \mid \exists t \in X, \text{ s.t. } s \xrightarrow{a} t\} \\
[\![[a]]\!]_\rho &= \lambda X.\{s \in \mathcal{S} \mid \forall t \in \mathcal{S}, s \xrightarrow{a} t \Rightarrow t \in X\} \\
[\![\mu Z.\varphi]\!]_\rho &= \bigsqcap\{f : 2^{\mathcal{S}} \to 2^{\mathcal{S}} \mid f \text{ monotone}, [\![\varphi]\!]_{\rho[Z\mapsto f]} \sqsubseteq f\} \\
[\![\nu Z.\varphi]\!]_\rho &= \bigsqcup\{f : 2^{\mathcal{S}} \to 2^{\mathcal{S}} \mid f \text{ monotone}, f \sqsubseteq [\![\varphi]\!]_{\rho[Z\mapsto f]}\} \\
[\![\varphi;\psi]\!]_\rho &= [\![\varphi]\!]_\rho \circ [\![\psi]\!]_\rho
\end{aligned}$$

A state s satisfies a formula φ under ρ, written $s \models_\rho \varphi$, iff $s \in [\![\varphi]\!]_\rho(\mathcal{S})$. If φ is a closed formula then ρ can be omitted and we write $[\![\varphi]\!](\mathcal{S})$ as well as $s \models \varphi$.

Two formulas φ and ψ are *equivalent*, written $\varphi \equiv \psi$, iff their semantics are the same, i.e. for every $\mathcal{T}$ and every ρ: $[\![\varphi]\!]^{\mathcal{T}}_{\rho} = [\![\psi]\!]^{\mathcal{T}}_{\rho}$.

In [10] it is shown how to embed $\mathcal{L}_\mu$ into FLC by using sequential composition: for instance, $\langle a\rangle\varphi$ becomes $\langle a\rangle;\varphi$. Therefore, we will sometimes omit the semicolon to maintain a strong resemblance to the syntax of $\mathcal{L}_\mu$. For example, $\langle a\rangle Z\langle a\rangle$ abbreviates $\langle a\rangle; Z; \langle a\rangle$. Note that $tl_Z = \emptyset$ for every Z in a formula that arises from this translation. We set $\mathrm{FLC}^- := \{\varphi \in \mathrm{FLC} \mid tl_Z = \emptyset \text{ for all}$

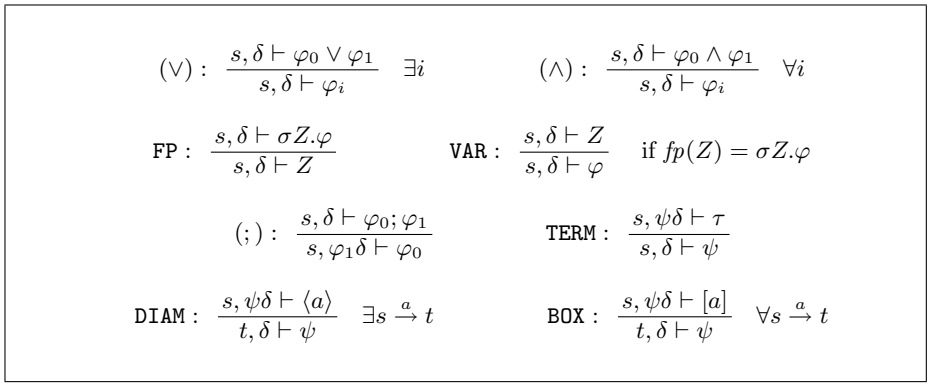

Fig. 1. The model checking game rules.

$Z \in Sub(\varphi)\}$. The fragment of FLC that is the image of $\mathcal{L}_\mu$ under the described translation is a genuine subset of FLC$^-$.

We introduce *approximants* of fixed point formulas. Let $fp(Z) = \mu Z.\varphi$ for some φ and let $\alpha, \lambda \in \mathbb{O}\mathrm{rd}$, the ordinals, where λ is a limit ordinal. Then $Z^0 := \texttt{ff}$, $Z^{\alpha+1} = \varphi[Z^\alpha/Z]$, $Z^\lambda = \bigvee_{\alpha<\lambda} Z^\alpha$. If $fp(Z) = \nu Z.\varphi$ then $Z^0 := \texttt{tt}$, $Z^{\alpha+1} = \varphi[Z^\alpha/Z]$, $Z^\lambda = \bigwedge_{\alpha<\lambda} Z^\alpha$. Note that $\mu Z.\varphi \equiv \bigvee_{\alpha\in\mathbb{O}\mathrm{rd}} Z^\alpha$ and $\nu Z.\varphi \equiv \bigwedge_{\alpha\in\mathbb{O}\mathrm{rd}} Z^\alpha$. If only finite transition systems are considered $\mathbb{O}\mathrm{rd}$ can be replaced by $\mathbb{N}$. If the size $|\mathcal{S}|$ of the underlying transition system is fixed then it serves as an upper bound for the number of approximants needed. This is expressed in the following Lemma which was proved in [9].

Lemma 1. ***(Approximants)*** *Let* $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} \mid a \in \mathcal{A}\}, L)$ *be finite with* $s \in \mathcal{S}, S \subseteq \mathcal{S}$.
a) $s \in [\![\mu Z.\varphi]\!]^{\mathcal{T}}_\rho(S)$ *iff* $\exists k \leq |\mathcal{S}|$, *s.t.* $s \in [\![Z^k]\!]^{\mathcal{T}}_\rho(S)$.
b) $s \in [\![\nu Z.\varphi]\!]^{\mathcal{T}}_\rho(S)$ *iff* $\forall k \leq |\mathcal{S}|$*:* $s \in [\![Z^k]\!]^{\mathcal{T}}_\rho(S)$.

3 Model Checking Games

Model checking games are played by two players, called $\exists$ and $\forall$, on a transition system and an FLC formula φ. Note that here we do not restrict ourselves to finite transition systems only. Player $\exists$ tries to establish that a given state s of a transition system $\mathcal{T}$ satisfies φ, whereas $\forall$ tries to show that $s \not\models \varphi$.

A play is a (possibly infinite) sequence $C_0, C_1, \ldots$ of configurations, and a configuration is an element of $Conf = \mathcal{S} \times Sub(\varphi)^* \times Sub(\varphi)$. It is written $s, \delta \vdash \psi$ where δ is interpreted as a stack of subformulas with its top on the left. The empty stack is denoted by ϵ. With a stack $\delta = \varphi_0 \ldots \varphi_k$ we associate the formula $\delta := \varphi_0; \ldots; \varphi_k$ while ϵ is associated with the formula τ.

Each play for s_0 of $\mathcal{T}$ and φ begins with $C_0 = s_0, \epsilon \vdash \varphi$. A play proceeds according to rules given in figure 1. Some of them require one of the players

to choose a subformula or a state. This is indicated at the right side of a rule. Rules $(\vee)$ and $(\wedge)$ are straightforward. Rules `VAR` and `FP` are justified by the unfolding characterisations of fixed points: $\sigma Z.\varphi \equiv \varphi[\sigma Z.\varphi/Z]$. If a formula $\varphi;\psi$ is encountered ψ is stored on the stack with rule $(;)$ to be dealt with later on while the players try to prove or refute φ. Modalities cause either of the players to choose a successor state. After that rules `DIAM` and `BOX` pop the top formula from the stack into the right side of the actual configuration. Rule `TERM` does the same without a choice of one of the players. In both cases the last formula on the right-hand side has been proved and those formulas that have been collected on the stack need to be proved or refuted.

Before we can define the winning conditions we need another definition. A variable Z is called *stack-increasing* in a play $C_0, C_1, \ldots$ if there are infinitely many configurations $C_{i_0}, C_{i_1}, \ldots$, s.t.

- $i_j < i_{j+1}$ for all $j \in \mathbb{N}$
- $C_{i_j} = s_j, \delta_j \vdash Z$ for some s_j and δ_j,
- for all $j \in \mathbb{N}$ exists $\gamma \in tl_Z \cup \{\epsilon\}$ s.t. $\delta_{j+1} = \gamma\delta_j$, and $\gamma = \epsilon$ iff $tl_Z = \emptyset$.

Player $\exists$ wins a play $C_0, \ldots, C_n, \ldots$ iff

1. $C_n = s, \delta \vdash q$ and $q \in L(s)$, or
2. $C_n = s, \epsilon \vdash \tau$, or
3. $C_n = s, \epsilon \vdash \langle a \rangle$ and there is a $t \in \mathcal{S}$, s.t. $s \xrightarrow{a} t$, or
4. $C_n = s, \delta \vdash [a]$, and $\delta = \epsilon$ or $\nexists t \in \mathcal{S}$, s.t. $s \xrightarrow{a} t$, or
5. the play is infinite, and there is a $Z \in \mathcal{V}$ s.t. Z is the greatest, w.r.t. $<_\varphi$, stack-increasing variable and $fp(Z) = \nu Z.\psi$ for some ψ.

Player $\forall$ wins such a play iff

6. $C_n = s, \delta \vdash q$ and $q \notin L(s)$, or
7. $C_n = s, \delta \vdash \langle a \rangle$, and $\nexists t \in \mathcal{S}$, s.t. $s \xrightarrow{a} t$, or
8. the play is infinite, and there is a $Z \in \mathcal{V}$ s.t. Z is the greatest, w.r.t. $<_\varphi$, stack-increasing variable and $fp(Z) = \mu Z.\psi$ for some ψ.

Winning conditions 1 and 6 suggest that game rule $(;)$ can be refined. Whenever the formula to be put on the stack is a $q \in Prop$ then the existing stack can be discarded. This does not effect the worst-case complexities, therefore we merely mention this optimisation.

A player has a winning strategy, or simply wins the game, for $s, \delta \vdash \varphi$ if she can enforce a winning play for herself, starting with this configuration.

The *full game tree* T for $C = s, \delta \vdash \varphi$ is the tree of configurations whose paths are plays starting with C. If player p has a winning strategy for C then the *game tree for player* p arises from T in the following way. If the next move is deterministic or taken by player $\overline{p}$ preserve and continue with all successor nodes. If the next configuration is chosen by p then preserve one successor C' s.t. p wins the game starting with C'. Obviously, the game tree for p is a representation of a winning strategy for p on C.

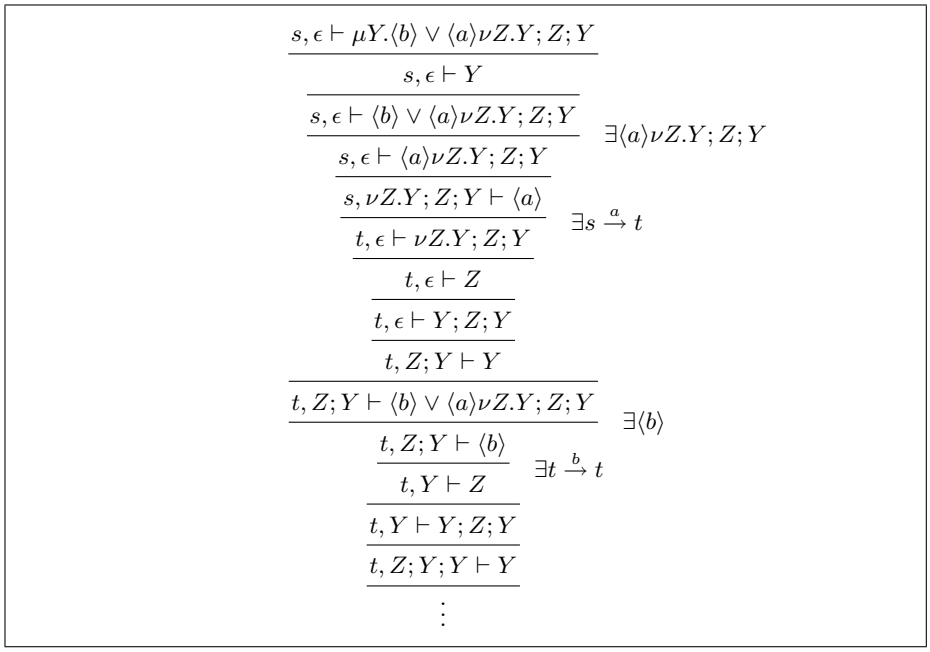

Fig. 2. ∃'s winning play from example 1.

The following example illustrates the importance of being stack-increasing. Note that in a $\mathcal{L}_\mu$ model checking game the winner is determined by the outermost variable that occurs infinitely often. There, if two variables Y and Z occur infinitely often then, say, $Y <_\varphi Z$, and $fp(Y)$ occurs infinitely often, too. Thus two occurrences of Y cannot be related to each other in terms of approximants. FLC only has this property for stack-increasing variables.

Example 1. Let $\varphi = \mu Y.\langle b\rangle \vee \langle a\rangle \nu Z.Y; Z; Y$. $ad(\varphi) = 1$ and $sd(\varphi) = 2$. Let $\mathcal{T}$ be the transition system consisting of states $\{s, t\}$ and transitions $s \xrightarrow{a} t$ and $t \xrightarrow{b} t$. $s \models \varphi$. The game tree for player ∃ is shown in figure 2. Since φ does not contain any $\wedge$, $[a]$, or $[b]$ player ∀ does not make any choices and the tree is in fact a single play.

Y and Z occur infinitely often in the play. However, neither $fp(Y)$ nor $fp(Z)$ does. Note that $Z <_\varphi Y$. Y gets "fulfilled" each time it is replaced by its defining fixed point formula, but reproduced by Z. On the other hand, Y does not start a new computation of $fp(Z)$ each time it is reproduced. But Y is not stack-increasing whereas Z is. And Z denotes a greatest fixed point, therefore player ∃ wins this play.

Before we can prove soundness and completeness of the games we need a few technical lemmas. Let $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} \mid a \in \mathcal{A}\}, L)$, $s \in \mathcal{S}$, $\varphi \in$ FLC, and $C = s, \delta \vdash \psi$ be a configuration in a game for s and φ. C is called *true* if $s \in [\![\varphi]\!]([\![\delta]\!](\mathcal{S}))$, and *false* otherwise.

Lemma 2. *Player $\exists$ preserves falsity and can preserve truth with her choices. Player $\forall$ preserves truth and can preserve falsity with his choices.*

Proof. First consider those choices involving disjuncts and conjuncts. Take a configuration $C = s, \delta \vdash \varphi_0 \vee \varphi_1$. Suppose C is false, i.e. $s \notin [\![\varphi_0]\!]([\![\delta]\!](\mathcal{S}))$ and $s \notin [\![\varphi_1]\!]([\![\delta]\!](\mathcal{S}))$. Regardless of which i player $\exists$ chooses, the configuration $s, \delta \vdash \varphi_i$ will be false. On the other hand, suppose C is true. Then $s \in [\![\varphi_0]\!]([\![\delta]\!](\mathcal{S}))$ or $s \in [\![\varphi_1]\!]([\![\delta]\!](\mathcal{S}))$, and player $\exists$ can preserve truth by choosing i accordingly.

Now consider a configuration $C = s, \psi\delta \vdash \langle a \rangle$. If C is true then there is a t s.t. $s \xrightarrow{a} t$ and $t \in [\![\psi; \delta]\!](\mathcal{S})$. By choosing this t, player $\exists$ can make the next configuration $t, \delta \vdash \psi$ true. If C is false then there is no such t and regardless of which transition $\exists$ chooses the following configuration will be false, too.

The proofs of the other cases are dual. □

Note that the rules that do not require a player to make a choice preserve both truth and falsity if variables are interpreted via their approximants.

Lemma 3. *Let $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} \mid a \in \mathcal{A}\}, L)$, $s \in \mathcal{S}$, $\varphi \in FLC$. In a play $C_0, C_1, \ldots$ for s and φ there is a unique greatest, with respect to $<_\varphi$, stack-increasing variable Z, if such one exists.*

Proof. Obviously, a finite play cannot have a stack-increasing variable. Let the infinite play at hand be $C_0, C_1, \ldots$. Suppose first there are two stack-increasing variables Z and Y. Then there must be two configurations $C_i = s, \delta \vdash Z$ and $C_j = t, \delta' \vdash Y$ with $i < j$. Either Y has been generated from the unfolding of Z in which case one of them is greater than the other. The reason is that the stack only contains elements of tl_V for some variable V. But $Y \in tl_Z$ implies either Y is free in $fp(Z)$ or $fp(Y) \in Sub(fp(Z))$. Therefore they must be comparable.

Suppose $\delta = \delta_0 Y \delta_1$. But then Z has either been generated from the unfolding of Y and they are comparable or $\delta' = \delta'_0 Z \delta'_1$. At every configuration the stack can only hold a finite number of variables. Therefore, in such an infinite play it is not possible that neither of the variables generates the other one infinitely often, and they must be comparable.

It remains to show that at least one variable is stack-increasing. Obviously, there must be a variable Z that occurs infinitely often. Moreover, this Z must generate itself infinitely often. Let $fp(Z) = \sigma Z.\varphi$. This means that for every occurrence of Z in a $C_i = s, \delta \vdash Z$, when Z is replaced by φ, the play must follow the syntactical structure of φ to one occurrence of Z in φ. In order to pop an element from δ an atomic formula in φ must have been reached, and Z in C_i did no regenerate itself. Suppose it did and the stack has been increased. Since rule `VAR` replaces a variable Z with its defining fixed point formula φ the additional part of the stack must consist of subformulas of φ only. Moreover, every subformula that occurs "before" Z in φ must have been removed from the stack before Z can be reached again. Therefore, the extension of the stack must be an element of tl_Z. □

One important property of a stack-increasing variable is: If its occurrence in a configuration $s, \delta \vdash Z$ is interpreted as the approximant Z^α then in its next

occurrence Z will denote $Z^{\alpha-1}$. This is because Z is outermost in the play at hand and thus the computation of $fp(Z)$ does not get restarted.

Theorem 1. (Soundness) *Let* $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} \mid a \in \mathcal{A}\}, L)$ *with* $s \in \mathcal{S}$ *and* $\varphi, \delta_0 \in$ *FLC. If* $s \notin [\![\varphi; \delta_0]\!](\mathcal{S})$ *then* $\forall$ *wins* $s, \delta_0 \vdash \varphi$.

Proof. Suppose $s \notin [\![\varphi]\!]([\![\delta_0]\!](\mathcal{S}))$. We construct a (possibly infinite) game tree for $\forall$ starting with $s, \delta_0 \vdash \varphi$. If $\varphi = \varphi_0 \wedge \varphi_1$, $\forall$ chooses the φ_i that makes $s, \delta \vdash \varphi_i$ false. If $\varphi = \varphi_0 \vee \varphi_1$ then the game tree is extended with both false configurations $s, \delta \vdash \varphi_i$. Similar arguments hold for the applications of rules `DIAM`, `BOX`, and `TERM`. Since falsity is preserved no finite path can be won by player $\exists$ since a false leaf implies that $\forall$ is the winner of that particular play.

We show that the game tree can be constructed such that player $\exists$ cannot win an infinite play either. Suppose the construction of the game tree reaches a configuration $t, \delta \vdash \nu Z.\psi$, s.t. Z is the unique stack-increasing variable according to Lemma 3. In the following configuration $t, \delta \vdash Z$, Z is interpreted as the least approximant Z^α s.t. $t \notin [\![Z^\alpha]\!]([\![\delta]\!](\mathcal{S}))$ but $t \in [\![Z^{\alpha-1}]\!]([\![\delta]\!](\mathcal{S}))$. Note that α cannot be a limit ordinal λ since $t \notin [\![\bigwedge_{\beta<\lambda} Z^\beta]\!](S)$ for any $S \subseteq \mathcal{S}$ implies $t \notin [\![Z^\beta]\!](S)$ for some $\beta < \lambda$. The next time a configuration $t', \delta' \vdash Z$ is reached Z is consequently interpreted as $Z^{\alpha-1}$. Again, if $\alpha - 1$ is a limit ordinal λ, then there must be a $\beta < \alpha$ such that $t' \notin [\![Z^\beta]\!]([\![\delta']\!](\mathcal{S}))$.

But ordinals are well-founded, i.e. the play must eventually reach a false configuration $t'', \delta'' \vdash Z$ in which Z is interpreted as Z^0. But $Z^0 \equiv \mathtt{tt}$ and $t'' \notin [\![\mathtt{tt}]\!](S)$ is not possible for any $S \subseteq \mathcal{S}$. We conclude that there is no least α that makes $t, \delta \vdash Z^\alpha$ false and that therefore $t, \delta \vdash \nu Z.\psi$ could not have been false either.

Since player $\exists$ cannot win any play in the game tree that is constructed in the described way player $\forall$ must win the game on $s, \delta_0 \vdash \varphi$. □

Theorem 2. (Completeness) *Let* $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} \mid a \in \mathcal{A}\}, L)$ *with* $s \in \mathcal{S}$ *and* $\varphi, \delta_0 \in$ *FLC. If* $s \in [\![\varphi; \delta_0]\!](\mathcal{S})$ *then* $\exists$ *wins* $s, \delta_0 \vdash \varphi$.

Proof. The proof is dual to the proof of the soundness theorem. Here, assuming $s \in [\![\varphi]\!]([\![\delta_0]\!](\mathcal{S}))$ we build a game tree for player $\exists$ starting with the true configuration $s, \delta_0 \vdash \varphi$ and preserving truth. If the construction of the game tree reaches a leaf the corresponding play must be won by $\exists$ since only she wins a finite play that ends with a true configuration.

Again, we show that player $\forall$ cannot win an infinite play either. Suppose there is a configuration $t, \delta \vdash \mu Y.\psi$ with Y being stack-increasing according to Lemma 3. In the next step, Y is interpreted as the least approximant Y^α s.t. $t \in [\![Y^\alpha]\!]([\![\delta]\!](\mathcal{S}))$ but $t \notin [\![Y^{\alpha-1}]\!]([\![\delta]\!](\mathcal{S}))$. Again, α cannot be a limit ordinal. The next time a configuration $t', \delta' \vdash Y$ is reached it becomes true if Y is interpreted as $Y^{\alpha-1}$. If $\alpha - 1$ is a limit ordinal then there is a smaller one that makes the configuration true.

Because of well-foundedness of the ordinals every infinite play must reach a configuration $t'', \delta'' \vdash Y$ in which Y is interpreted as Y^0. But $Y^0 \equiv \mathtt{ff}$ and

therefore $t'', \delta'' \vdash Y$ cannot be true. Thus, $t, \delta \vdash \mu Y.\psi$ could not have been true either.

Since player $\forall$ cannot win any play of the game tree that is constructed in the described way player $\exists$ must win the game starting with $s, \delta_0 \vdash \varphi$. □

From Theorem 1 and 2 follows that the model checking problem for FLC can be rephrased as: $s \models \varphi$ iff player $\exists$ wins $s, \epsilon \vdash \varphi$.

4 Decidability and Complexity

In [10] Müller-Olm has shown that FLC model checking is undecidable for BPA already. We show that this result can easily be improved a bit.

Theorem 3. *FLC model checking is undecidable for normed deterministic BPA.*

Proof. Based on an early result from language theory in [4] it is shown in [6] that the simulation problem for deterministic normed BPA is undecidable. Given a BPA process Q one can construct an FLC formula ϕ_Q, s.t. $P \models \phi_Q$ iff P simulates Q. The construction for arbitrary BPA processes is shown in [10] and works in particular for normed deterministic BPA. □

However, over finite transition systems the model checking problem for FLC is decidable [10, 9]. We describe how the games of the previous section can be used to obtain a local model checker.

Theorem 4. *Let $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} \mid a \in \mathcal{A}\}, L)$ be finite with $s \in \mathcal{S}$ and $\varphi, \delta \in FLC^{k,n}$. Deciding the winner of $s, \delta \vdash \varphi$ is in PSPACE for all $k, n \in \mathbb{N}$.*

Proof. If the underlying transition system is finite then the least approximants used in the proofs of Theorem 1 and 2 are bounded by $|\mathcal{S}|$ according to Lemma 1. An algorithm deciding the winner of $s, \delta \vdash \varphi$ can index variables occurring in a play as the corresponding approximant. This means, rules `FP` and `VAR` are used as

$$\frac{s, \delta \vdash \sigma Z.\varphi}{s, \delta \vdash Z^{|\mathcal{S}|}} \qquad \frac{s, \delta \vdash Z^k}{s, \delta \vdash \varphi[Z^{k-1}/Z]} \quad \text{if } fp(Z) = \sigma Z.\varphi$$

Then, configurations of the form $t, \delta \vdash Z^0$ with $fp(Z) = \sigma Z.\psi$ for some ψ, δ and t are winning for player $\exists$ if $\sigma = \nu$ and winning for player $\forall$ if $\sigma = \mu$.

Next we analyse the maximum length of a play of $s, \delta \vdash \varphi$. Suppose $ad(\varphi) = 0$. At most $O(|\mathcal{S}| \cdot |\varphi|)$ steps are possible before a terminal configuration with a Z^0 must be reached, if the sequential depth of φ is 0. However, if it is greater than 0 then at the beginning a $Z^{|\mathcal{S}|}$ can be pushed onto the stack where it remains while another Z^k gets unfolded at most $|\mathcal{S}|$ times before it might disappear. Then the $Z^{|\mathcal{S}|}$ from the stack can be popped and create the same situation by unfolding to more than one $Z^{|\mathcal{S}|-1}$ of which one remains on the stack again. Generally, the maximum length of a play in this situation is $O((|\mathcal{S}| \cdot |\varphi|)^{sd(\varphi)})$.

Let now $ad(\varphi) = k > 0$. Take the outermost variable Z that occurs in the play at hand. With each unfolding it can start a subplay on a formula with alternation depth $k-1$. Therefore the overall maximum length of the play is $O(((|\mathcal{S}| \cdot |\varphi|)^{sd(\varphi)})^{ad(\varphi)}) = O((|\mathcal{S}| \cdot |\varphi|)^{sd(\varphi) \cdot ad(\varphi)})$.

An alternating algorithm can decide the winner of $s, \delta \vdash \varphi$ by simply playing the game for it. For input formulas $\delta, \varphi \in \mathrm{FLC}^{k,n}$ the alternation depth and sequential depth are bounded. Thus, the time needed is polynomial in the size of the formula and the size of the transition system. According to [1] there is a deterministic procedure that needs space which is polynomial in the size of the formula and in the size of the transition system. □

This argument, applied to formulas of arbitrary alternation or sequential depth, yields an EXPSPACE procedure. This follows from the fact that the alternating algorithm needs time exponential in the alternation and sequential depth of the input formula, and AEXPTIME = EXPSPACE. To show that game-based model checking for FLC can be done in EXPTIME an alternating algorithm must not use more than polynomial space. Equally, a single play must be playable using at most polynomial space.

We show why this is not likely to be possible. First we consider a slightly different way of proving soundness and completeness of the games which only applies if the underlying transition system is finite. Remember that in the proofs of Theorem 1 and 2 variables are interpreted as approximants, and contradictions arise at configurations $t, \delta' \vdash Z^0$. Suppose $fp(Z) = \mu Z.\psi$ and the game tree is constructed preserving truth. Then at its first occurrence Z is interpreted as the least Z^k which makes the configuration, say, $t, \delta' \vdash Z^k$ true. However, if later another true configuration $t, \delta'' \vdash Z^j$ is seen and $[\![\delta'']\!](\mathcal{S}) \subseteq [\![\delta']\!](\mathcal{S})$ then this contradicts the fact that k was chosen least.

This occurs trivially after $O(|\mathcal{S}| \cdot 2^{|\mathcal{S}|})$ steps since there are only $|\mathcal{S}|$ many different states and $2^{|\mathcal{S}|}$ many different sets of them. In most cases this situation will occur in a stack of polynomial size already. However, there are cases in which the stack can grow super-polynomially. That means there are m configurations $s_i, \delta_i \vdash Z$ s.t. $[\![\delta_i]\!](\mathcal{S}) \not\subseteq [\![\delta_j]\!](\mathcal{S})$ for $j < i \leq m$ and m is not polynomially bounded by the input size.

Example 2. Let $a, b \in \mathcal{A}$. Take n pairwise different prime numbers $p_1, \ldots, p_n$. Let $P_0 = 0$ and $P_i = \Sigma_{j=1}^{i} p_j$ be the sum of the first i prime numbers for $i = 1, \ldots, n-1$. We construct a transition system $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} \mid a \in \mathcal{A}\}, L)$ with $\mathcal{S} = \{0, \ldots, P_n - 1\}$. Transitions in $\mathcal{T}$ are given by $j \xrightarrow{a} j+1$ for all $j < P_n$, $j \neq P_i - 1$ for all $i \in \{1, \ldots, n\}$, and $P_i - 1 \xrightarrow{a} P_{i-1}$ for all $i \in \{1, \ldots, n\}$. Finally, $i \xrightarrow{b} j$ iff $j \xrightarrow{a} i$. $\mathcal{T}$ consists of n cycles of length $p_1, \ldots, p_n$ which can be traversed along a-transitions, say, clockwise and through b-transitions counterclockwise. Feel free to add as many c-transitions if $c \neq a$ and $c \neq b$ to make $\mathcal{T}$ connected. Finally, we use one proposition q which holds on one state of each cycle only. $q \in L(j)$ iff $j = P_i$ for some $i \in \{0, \ldots, n-1\}$.

The formula under examination is $\varphi := (\nu Z.\tau \wedge \langle a \rangle Z \langle b \rangle); q$. It says that there is an a-path s.t. after each n a's another n b's can be done to satisfy q. $0 \models \varphi$

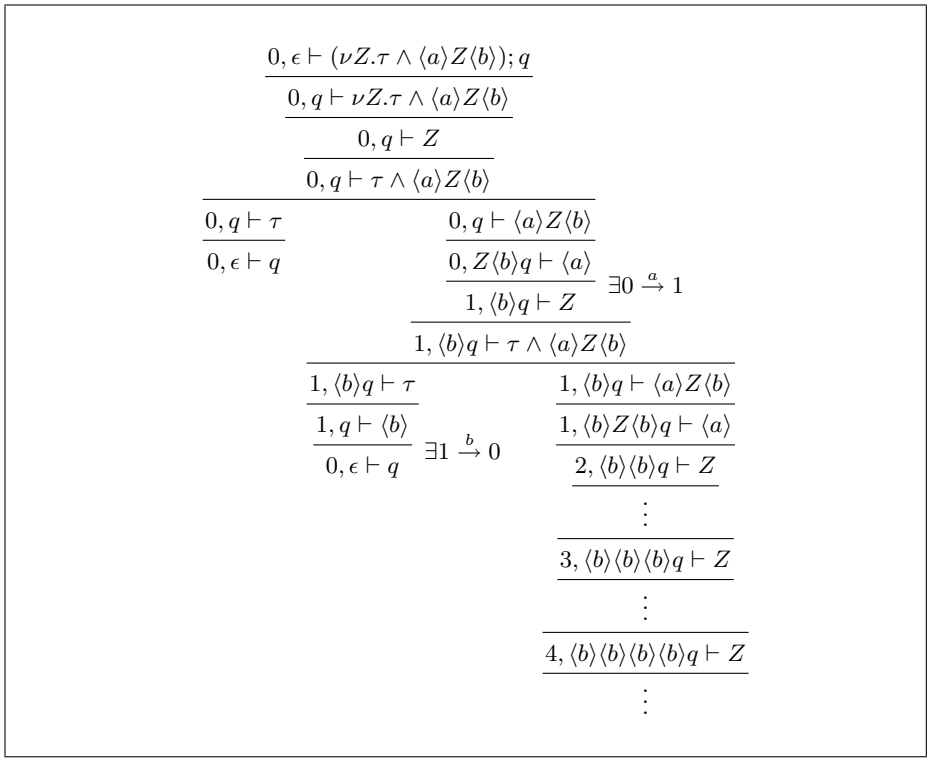

Fig. 3. $\exists$'s game tree from example 2.

which can easily be seen using the games of the previous section. Player $\forall$ can never choose τ since $0 \models q$ and every sequence of m a-transitions away from 0 leads to a state that can do m b-transitions back to 0. But then player $\exists$ wins because the play repeats on a ν-variable. Her game tree is shown in figure 3.

If approximants are used explicitly as suggested in the proof of Theorem 4 the stack cannot grow larger than P_n. This is not surprising since $\varphi \in \mathrm{FLC}^{0,0}$. However, let $S_q := \{P_i \mid i \in \{0, \ldots, n-1\}\}$ be the set of all states satisfying q. We write $\langle b\rangle^i$ to abbreviate $\langle b\rangle; \ldots; \langle b\rangle$, where it occurs exactly i times. We claim that $[\![\langle b\rangle^i]\!](S_q) \neq [\![\langle b\rangle^j]\!](S_q)$ for $i, j < \Pi_{i=1}^n p_i$, $i \neq j$. And, since $|[\![\langle b\rangle^i]\!](S_q)| = n$ for all $i < \Pi_{i=1}^n p_i$, even $[\![\langle b\rangle^i]\!](S_q) \not\subseteq [\![\langle b\rangle^j]\!](S_q)$ for $i, j < \Pi_{i=1}^n$, $i \neq j$. Take a state in the k-th cycle. It belongs to $[\![\langle b\rangle^i]\!](S_q)$ iff it is the $(i \bmod p_k)$-th b-predecessor of P_{k-1}. In other words, the sets $[\![\langle b\rangle^i]\!](S_q)$ can be defined by moving markers along a-transitions in each cycle starting with S_q. Since the lengths of the cycles are pairwise different prime numbers the same set is only marked after $\Pi_{i=1}^n p_i$ steps.

This means that the first $(\Pi_{i=1}^n p_i) - 1$ sequences $\langle b\rangle^i; q$ define pairwise uncomparable sets of states. Clearly, $\Pi_{i=1}^n p_i \notin O(n^k)$ for all $k \in \mathbb{N}$.

Corollary 1. *Let $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} \mid a \in \mathcal{A}\}, L)$ be finite with $s \in \mathcal{S}$ and $\varphi, \delta \in FLC^{\cdot}$. Deciding the winner of $s, \delta \vdash \varphi$ is in EXPSPACE.*

It should be mentioned that, if the requirement of locality is dropped, then model checking can be done in EXPTIME in the general case. For this, a configuration like $s, \delta \vdash \varphi; \psi$ can be model checked in the following way. Player $\exists$ chooses a set T s.t. $T = [\![\psi; \delta]\!](\mathcal{S})$. Then, player $\forall$ either selects to continue with $s, \epsilon \vdash \varphi$ and the additional winning requirement that player $\exists$ can only win if the last state reached in a subplay is a $t \in T$. Or he chooses a $t \in T$ and continues with $t, \delta \vdash \psi$. It is easy to see that the stack becomes obsolete and that therefore configurations are of polynomial size only. However, player $\exists$'s choice of T prevents the model checker from being local.

The next theorem analyses the complexity of the games if applied to $\mathcal{L}_\mu$ formulas. In this case it is helpful to start the game with an empty stack.

Theorem 5. *Let $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} \mid a \in \mathcal{A}\}, L)$ be finite with $s \in \mathcal{S}$ and $\varphi \in FLC^{-}$. Deciding the winner of $s, \epsilon \vdash \varphi$ is in NP∩co-NP.*

Proof. The stack can never grow larger than φ and will be empty each time a variable is reached. The resulting games are essentially the same as the model checking games for $\mathcal{L}_\mu$ from [12]. It is known from [3] for example that the winner of those games can be decided in NP∩co-NP. The same technique applies here.

Clearly, the game graph for $s, \epsilon \vdash \varphi$ is finite and of size polynomial in the input. To decide whether player $\exists$ wins $s, \epsilon \vdash \varphi$ a nondeterministic algorithm can guess annotations $(k_1, \ldots, k_n)$ for each μ-variable Y. The meaning of such an annotation is: Y has to be unfolded k_n times at this moment and there are outer variables $Z_1, \ldots, Z_{n-1}$ which have been unfolded $k_1, \ldots, k_{n-1}$ times. The maximal size of such an annotation is $O(ad(\varphi) \cdot log|\mathcal{S}|)$.

Finally, the algorithm has to verify that the order of the annotations is well-founded, i.e. for every μ-variable Y: if there is a path from $s, \delta \vdash Y$ with annotation $K = k_1, \ldots, k_n$ to $t, \delta' \vdash Y$ with annotation $K' = k'_1, \ldots, k'_n$ then K' is lexicographically smaller than K.

This proves that deciding the winner of $s, \epsilon \vdash \varphi$ is in NP. Inclusion in co-NP follows from the fact that the same argument applies to player $\forall$ and ν-variables to decide whether he wins $s, \epsilon \vdash \varphi$. □

This is not a contradiction to the PSPACE-hardness proved in [9]. There, reductions from the validity problem for QBF and from the universal acceptance problem for NFAs are presented. In both cases the constructed formulas are not in FLC^{-}.

Even if the starting stack in the game of Theorem 5 is non-empty the semantics of approximants will always be evaluated on the same set of states. However, if the stack is $\delta = \psi\delta'$ and deciding the winner of $t, \delta' \vdash \psi$ is in the complexity class $\mathcal{C}$ for any $t \in \mathcal{S}$, then deciding the winner of $s, \delta \vdash \varphi$ is in (NP∩co-NP)∪$\mathcal{C}$.

Theorem 5 becomes interesting if applied to formulas in FLC^{-} that are not a translation of a $\mathcal{L}_\mu$ formula but are equivalent to a formula in $\mathcal{L}_\mu$. One example is $\nu Z.(\langle a_0 \rangle \wedge \langle b_0 \rangle); \ldots; (\langle a_0 \rangle \wedge \langle b_0 \rangle); Z$ which is exponentially more succinct than

its equivalent in $\mathcal{L}_\mu$, see [9]. Theorem 5 suggests that the model checking games for FLC behave better than those for $\mathcal{L}_\mu$ in this case, although equation-based model checkers for $\mathcal{L}_\mu$ can do equally well.

5 Conclusion

We have given a game-based account of the model checking problem for FLC. The main feature of the games is their ability to allow local model checking. This makes them suitable for verification purposes. We have also shown that the games are optimal for fixed alternation and sequential depth in terms of complexity bounds. These parameters can be assumed to be small for formulas that express properties desired in the verification of finite transition systems.

It is not known whether FLC's model checking problem for unbounded alternation depth is EXPTIME-hard. Compare this to the $\mathcal{L}_\mu$ situation: NP∩co-NP for the general case and P-complete for bounded alternation.

One candidate for establishing EXPTIME-hardness is the model checking problem for $\mathcal{L}_\mu$ and BPA or PDA. However, this cannot work since these problems are EXPTIME-complete for alternation free $\mathcal{L}_\mu$ formulas already. It remains to see whether EXPTIME-hardness can be established, and whether local FLC model checking for unbounded alternation depth is possible to do in EXPTIME.

References

[1] A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, January 1981.

[2] E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, chapter 14, pages 996–1072. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, New York, N.Y., 1990.

[3] E. A. Emerson, C. S. Jutla, and A. P. Sistla. On model checking for the μ-calculus and its fragments. *Theoretical Computer Science*, 258(1–3):491–522, 2001.

[4] E. P. Friedman. The inclusion problem for simple languages. *TCS*, 1(4):297–316, April 1976.

[5] R. Goré. Tableau methods for modal and temporal logics. In M. D'Agostino, D. Gabbay, R. Hähnle, and J. Posegga, editors, *Handbook of Tableau Methods*. Kluwer, Dordrecht, 1999.

[6] J. F. Groote and H. Hüttel. Undecidable equivalences for basic process algebra. *Information and Computation*, 115(2):354–371, December 1994.

[7] D. Janin and I. Walukiewicz. On the expressive completeness of the propositional μ-calculus with respect to monadic second order logic. In U. Montanari and V. Sassone, editors, *CONCUR '96: Concurrency Theory, 7th Int. Conf.*, volume 1119 of *LNCS*, pages 263–277, Pisa, Italy, 26–29 August 1996. Springer.

[8] D. Kozen. Results on the propositional mu-calculus. *TCS*, 27:333–354, December 1983.

[9] M. Lange and C. Stirling. Model checking fixed point logic with chop. In M. Nielsen and U. H. Engberg, editors, *Proc. Foundations of Software Science and Computation Structures, FOSSACS'02*, volume 2303 of *LNCS*, pages 250–263, Grenoble, France, 2002. Springer.

[10] M. Müller-Olm. A modal fixpoint logic with chop. In C. Meinel and S. Tison, editors, *Proc. 16th Annual Symp. on Theoretical Aspects of Computer Science, STACS'99*, volume 1563 of *LNCS*, pages 510–520, Trier, Germany, 1999. Springer.

[11] C. Stirling. Modal and temporal logics. In *Handbook of Logic in Computer Science*, volume 2 (Background: Computational Structures), pages 477–563. Clarendon Press, Oxford, 1992.

[12] C. Stirling. Local model checking games. In I. Lee and S. A. Smolka, editors, *Proc. 6th Int. Conf. on Concurrency Theory, CONCUR'95*, volume 962 of *LNCS*, pages 1–11, Berlin, Germany, August 1995. Springer.

[13] A. Tarski. A lattice-theoretical fixpoint theorem and its application. *Pacific J.Math.*, 5:285–309, 1955.