Verification of Non-Regular Properties

Martin Lange
Institut für Informatik
Ludwig-Maximilians-Universität München
e-mail: mlange@informatik.uni-muenchen.de

ABSTRACT

We present a game-based formalism that can be used to do local model checking for FLC, a modal fixed point logic that extends the μ-calculus with a sequential composition operator. This logic is capable of expressing non-regular properties which are interesting for verification purposes.

1. Introduction

The modal μ-calculus [2] is an important specification language for the verification of temporal properties. This is mainly due to the fact that it subsumes most temporal, dynamic or description logics.

On the other hand, the modal μ-calculus is equi-expressive to Monadic Second Order Logic [1], i.e. it can only describe "regular" properties. In [7], Müller-Olm has introduced FLC, *Fixed Point Logic with Chop*, which extends the modal μ-calculus with a sequential composition operator, and has shown that its expressive power is not limited by regularity. In fact, it is not hard to define the language $\{a^n b^n c^n \mid n \in \mathbb{N}\}$ in FLC on words for example.

On finite structures, FLC's model checking problem is decidable [7]. In [5], a tableau formalism was introduced that can be used to do *global* model checking. Here we present a game-based framework [4] for *local* model checking. These games cannot deny a certain similarity with alternating push-down automata over trees. Just as FLC extends the modal μ-calculus, they extend its model checking games [8].

We also give examples of FLC-definable properties that are interesting for verification purposes and present the most important results on FLC.

2. Preliminaries

A *transition system* $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} \mid a \in \mathcal{A}\}, L)$ over a set $\mathcal{P} = \{\mathtt{tt}, \mathtt{ff}, q, \overline{q}, \ldots\}$ of propositional constants consists of *states*, *transition relations* and a *labelling function* from states to sets of propositions. 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 $q \in \mathcal{P}$, $a \in \mathcal{A}$, and $Z \in \mathcal{V}$, a set of propositional variables. We will write σ for μ or ν. 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. The set $Sub(\varphi)$ of *subformulas* of φ and its *alternation depth* $ad(\varphi)$ are defined as usual.

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 use sequential composition for sets of formulas in a straightforward way: $\{\varphi_0, \ldots, \varphi_n\}; \psi := \{\varphi_0; \psi, \ldots, \varphi_n; \psi\}$. Let

$$tl_Z(\psi) := \{\psi\} \text{ if } \psi \in \mathcal{P} \cup \{\tau, \langle a\rangle, [a] \mid a \in \mathcal{A}\} \qquad tl_Z(\sigma Y.\psi) := tl_Z(\psi)$$

$$tl_Z(\varphi \vee \psi) = tl_Z(\varphi \wedge \psi) := tl_Z(\varphi) \cup tl_Z(\psi) \qquad tl_Z(Y) := \begin{cases} \{Y\} & \text{if } Y \neq Z \\ \{\tau\} & \text{o.w.} \end{cases}$$

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

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

An important factor in the complexity of FLC's model checking problem is the *sequential depth* $sd(\varphi)$ of a formula. 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

$$sd_Z(\varphi \vee \psi) := max\{sd_Z(\varphi), sd_Z(\psi)\} \qquad sd_Z(\varphi \wedge \psi) := max\{sd_Z(\varphi), sd_Z(\psi)\}$$

$$sd_Z(\varphi; \psi) := sd_Z(\varphi) + sd_Z(\psi) \qquad sd_Z(\sigma Y.\varphi) := sd_Z(\varphi)$$

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

To simplify the notation of a formula's semantics we assume a transition system $\mathcal{T}$ to be fixed for the remainder of the paper. In order to handle open formulas we need environments $\rho : \mathcal{V} \to (2^{\mathcal{S}} \to 2^{\mathcal{S}})$. $\rho[Z \mapsto f]$ 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 [9] 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.

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

A state s satisfies a formula φ under ρ, written $s \models_\rho \varphi$, iff $s \in [\![\varphi]\!]_\rho(\mathcal{S})$.

Example Let $\mathcal{A} = \{a, b\}$ and $\varphi = \nu Y.[b]\mathtt{ff} \wedge [a](\nu Z.[b] \wedge [a](Z; Z)); (([a]\mathtt{ff} \wedge [b]\mathtt{ff}) \vee Y)$. It says "the number of bs never exceeds the number of as" which is non-regular and, therefore, is not expressible in $\mathcal{L}_\mu$. This is an interesting property of protocols when a and b are the actions *send* and *receive*.

The subformula $\psi = \nu Z.[b] \wedge [a](Z; Z)$ expresses "there can be at most one b more than there are as". This can be understood best by unfolding the fixed point formula and thus obtaining

sequences of modalities and variables. It is easy to see that replacing a Z with a $[b]$ reduces the number of Zs whereas replacing it with the other conjunct adds a new Z to the sequence.

Then, $[b]\mathtt{ff} \wedge [a]\psi$ postulates that at the beginning no b is possible and for every n as there can be at most n bs. Finally, the Y in φ allows such sequences to be composed or finished in a deadlock state.

Example The property of *repeating words on every path* is expressible in FLC, if there is an end marker #. The following formula says that every sequence of actions w ending in a # is followed by another w.

$$\varphi := \# \vee \bigwedge_{a \in \Sigma} \psi_a$$
$$\psi_a := (\mu Z.\#; \mathtt{tt} \vee ([-]; Z \wedge [a]; (\mu Y.\# \vee [-]; Y)); [-]); \langle a \rangle$$

Each ψ_a checks for the letter $a \in \Sigma$: if a occurs before # on a path at position n then there is an a-action n steps after the later #.

3. Local model checking games for FLC

Model checking games are played by two players, called $\exists$ and $\forall$, on a transition system $\mathcal{T}$ and an FLC formula φ. Player $\exists$ tries to establish that a given state s of $\mathcal{T}$ 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 $\mathit{Conf} = \mathcal{S} \times \mathit{Sub}(\varphi)^* \times \mathit{Sub}(\varphi)$. 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.

$$(\vee): \ \frac{s, \delta \vdash \varphi_0 \vee \varphi_1}{s, \delta \vdash \varphi_i} \ \exists i \qquad (\wedge): \ \frac{s, \delta \vdash \varphi_0 \wedge \varphi_1}{s, \delta \vdash \varphi_i} \ \forall i \qquad (\mathtt{FP}): \ \frac{s, \delta \vdash \sigma Z.\varphi}{s, \delta \vdash Z}$$

$$(\mathtt{VAR}): \ \frac{s, \delta \vdash Z}{s, \delta \vdash \varphi} \ \text{if } \mathit{fp}(Z) = \sigma Z.\varphi \qquad (;): \ \frac{s, \delta \vdash \varphi_0; \varphi_1}{s, \varphi_1 \delta \vdash \varphi_0} \qquad (\mathtt{TERM}): \ \frac{s, \psi\delta \vdash \tau}{s, \delta \vdash \psi}$$

$$(\mathtt{DIAM}): \ \frac{s, \psi\delta \vdash \langle a \rangle}{t, \delta \vdash \psi} \ \exists\, s \xrightarrow{a} t \qquad (\mathtt{BOX}): \ \frac{s, \psi\delta \vdash [a]}{t, \delta \vdash \psi} \ \forall\, s \xrightarrow{a} t$$

Figure 1: The model checking game rules.

Each play for s_0 of $\mathcal{T}$ and φ begins with $C_0 = s_0, \epsilon \vdash \varphi$. A play proceeds according to the rules depicted 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 then 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. The winning conditions are not straight-forward but require another definition.

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.

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 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 occur infinitely often then one of them is outer and the inner one's defining fixed point formula, say $fp(Y)$, occurs infinitely often, too. Thus two occurrences of Y cannot be related to each other in terms of approximants indices. FLC only has this property for stack-increasing variables.

Example 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 $\exists$ is shown in Figure 2. Since φ does not contain any $\wedge$, $[a]$ or $[b]$, player $\forall$ does not make any choices and the tree is in fact a single play.

Both 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 $\exists$ wins this play.

4. Results on FLC('s model checking problem)

Theorem ***(Decidability)*** *FLC's satisfiability checking problem is undecidable; FLC does not have the finite model property; its model checking problem is decidable for finite structures* [7]*; but undecidable for normed deterministic BPA already* [4].

Theorem ***(Expressiveness)*** *On linear structures (infinite words) the class of FLC-definable languages coincides with the class of alternating context-free grammars (with a parity acceptance condition)* [3].

Theorem ***(Complexity)*** *FLC's model checking problem is in EXPTIME and PSPACE-hard* [5]*, even for a fixed formula* [6]*. For formulas with fixed alternation (and sequential) depth, the model checking problem is in PSPACE.*

Obviously, the complexity results refer to finite structures only.

$$
\begin{array}{c}
s, \epsilon \vdash \mu Y.\langle b\rangle \vee \langle a\rangle \nu Z.Y; Z; Y \\ \hline
s, \epsilon \vdash Y \\ \hline
s, \epsilon \vdash \langle b\rangle \vee \langle a\rangle \nu Z.Y; Z; Y \\ \hline
s, \epsilon \vdash \langle a\rangle \nu Z.Y; Z; Y \\ \hline
s, \nu Z.Y; Z; Y \vdash \langle a\rangle \\ \hline
t, \epsilon \vdash \nu Z.Y; Z; Y \\ \hline
t, \epsilon \vdash Z \\ \hline
t, \epsilon \vdash Y; Z; Y \\ \hline
t, Z; Y \vdash Y \\ \hline
t, Z; Y \vdash \langle b\rangle \vee \langle a\rangle \nu Z.Y; Z; Y \\ \hline
t, Z; Y \vdash \langle b\rangle \\ \hline
t, Y \vdash Z \\ \hline
t, Y \vdash Y; Z; Y \\ \hline
t, Z; Y; Y \vdash Y \\ \hline
\vdots
\end{array}
$$

(Rule labels: $\exists\langle a\rangle \nu Z.Y; Z; Y$ at the step from $s, \epsilon \vdash \langle b\rangle \vee \langle a\rangle \nu Z.Y; Z; Y$ to $s, \epsilon \vdash \langle a\rangle \nu Z.Y; Z; Y$; $\exists s \xrightarrow{a} t$ at the step from $s, \nu Z.Y; Z; Y \vdash \langle a\rangle$ to $t, \epsilon \vdash \nu Z.Y; Z; Y$; $\exists\langle b\rangle$ at the step from $t, Z; Y \vdash \langle b\rangle \vee \langle a\rangle \nu Z.Y; Z; Y$ to $t, Z; Y \vdash \langle b\rangle$; $\exists t \xrightarrow{b} t$ at the step from $t, Z; Y \vdash \langle b\rangle$ to $t, Y \vdash Z$.)

Figure 2: ∃'s winning play from the example.

References

[1] 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, *Proc. 7th Conf. on Concurrency Theory, CONCUR'96*, volume 1119 of *LNCS*, pages 263–277, Pisa, Italy, August 1996. Springer.

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

[3] M. Lange. Alternating context-free languages and linear time μ-calculus with sequential composition. In P. Panangaden and U. Nestmann, editors, *Proc. 9th Workshop on Expressiveness in Concurrency, EXPRESS'02*, volume 68.2 of *ENTCS*, pages 71–87, Brno, Czech Republic, August 2002. Elsevier.

[4] M. Lange. Local model checking games for fixed point logic with chop. In L. Brim, P. Jančar, M. Křetínský, and A. Kučera, editors, *Proc. 13th Conf. on Concurrency Theory, CONCUR'02*, volume 2421 of *LNCS*, pages 240–254, Brno, Czech Republic, August 2002. Springer.

[5] M. Lange and C. Stirling. Model checking fixed point logic with chop. In M. Nielsen and U. H. Engberg, editors, *Proc. 5th Conf. on Foundations of Software Science and Computation Structures, FOSSACS'02*, volume 2303 of *LNCS*, pages 250–263, Grenoble, France, April 2002. Springer.

[6] M. Müller-Olm. private communication.

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

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

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