Non-Regular Fixed-Point Logics and Games

Stephan Kreutzer[1]
Martin Lange[2]

[1] Oxford University Computing Laboratory,
Wolfson Building,
Parks Road,
Oxford, OX1 3QD,
England.

[2] Institut für Informatik,
Ludwig-Maximilians-Universität München,
Oettingenstr. 67,
80538 München,
Germany.

kreutzer@comlab.ox.ac.uk, martin.lange@ifi.lmu.de

1 Introduction

Modal and temporal logics. The most commonly used specification logics in the theory of computer aided verification are based on propositional modal logic augmented by temporal operators. Among those one can broadly distinguish between linear and branching time logics, depending on how they treat the temporal development of processes. The modal μ-calculus, $\mathcal{L}_\mu$ for short, provides a common generalization of most temporal logics. It is defined as the extension of basic propositional modal logic by rules to form the least and the greatest fixed point of definable monotone operators.

$\mathcal{L}_\mu$ is a *regular* logic in the sense that it can be translated into monadic second order logic (MSO) and therefore can only define regular classes of trees and their representations as transition systems. It is even equi-expressive to the bisimulation-invariant fragment of MSO over trees or graphs [9] and can therefore be seen as *the* regular branching time temporal logic.

Temporal logics such as LTL, CTL or CTL* are all embeddable into $\mathcal{L}_\mu$. They can express important properties – such as reachability, safety, liveness, fairness, etc. – and specifications in these languages can be verified automatically and in many cases also efficiently in process models. However, a number of natural properties of processes are no longer regular and therefore cannot be expressed in any of these logics. For instance, one cannot express that a specific event occurs in all possible execution traces at

the same time [7], that every transmission is acknowledged, or that there are no more *returns* than *calls*.

To express these properties in a logic, the logic needs to be able to count to some extent, at least to compare cardinalities, i.e. it needs to incorporate non-regular properties. There are various potential ways of defining logics with non-regular features.

One option is to add a bisimulation preserving form of counting explicitly, i.e. to consider a modal analogue to first-order plus counting. Similarly, one could add specific operators for the tasks at hand, an operator to compare cardinalities, for instance. In this way, logics tailored towards specific tasks can be obtained.

Another possibility is to enrich the models over which a regular logic is interpreted with some extra information and let the operators of the logic make use of this. This has been done in the linear time temporal logic CaRet for example [1]. It is interpreted in an LTL-like fashion over infinite words that represent runs of recursive processes, i.e. positions in these words are marked with *call* and *return* symbols. CaRet then extends LTL by allowing its operators to access *return* positions that match the previous *call* position in the sense that in between the *calls* and *returns* form a balanced Dyck-language. This way, non-regularity is added into the meta-logic rather than the logic itself.

A different approach is to consider general purpose logics employing more expressive fixed-point constructs than least fixed points of monotone operators. This is the trait we follow in this paper. There are (at least) two ways in which the modal μ-calculus can be extended in this way: one can relax the restriction to monotone operators or one can stick to monotone operators but allow fixed-point inductions of higher order. We consider these options and introduce two modal fixed-point logics: (1) the Modal Iteration Calculus (MIC) which replaces least and greatest fixed points in $\mathcal{L}_\mu$ by inflationary and deflationary ones; and (2) Fixed-Point Logic with Chop (FLC) which extends $\mathcal{L}_\mu$ with an operator for sequential composition. This necessitates a higher-order semantics.

Non-regular properties. We illustrate these logics by a set of examples of non-regular properties, i.e. properties that cannot be expressed in $\mathcal{L}_\mu$.

The most obvious choices come from formal language theory. The first hurdle to take for a logic that wants to be able to express non-regular properties is the standard example of a context-free and non-regular language, i.e. $L = \{a^n b^n \mid n \geq 1\}$. Note that MIC and FLC are branching time logics, and hence, we will look for formulas that are satisfied by a state if, and only if, it has a maximal outgoing path whose labels form a word in L. While this is a toy example, there are also formal languages which give rise to interesting program correctness properties. Let $\Sigma = \{a, b\}$ and consider the

language L consisting of all words that do not have a prefix in which there are more b's than a's. It is easily seen to be non-regular but context-free, and it is the formal language basis of the aforementioned property about *calls* and *returns*. A suitable reformulation of this language in a formula of MIC or FLC would show that these logics can express properties of recursive processes like "no process is ended unless it has been started" etc. Note that this is also the same as absence of underflows in FIFO or LIFO buffers of unbounded size.

Non-regularity, however, need not be rooted in the theory of formal word languages. Branching time logics whose expressive power exceeds that of $\mathcal{L}_\mu$ may also be able to express properties that are unrelated to context-free languages. For example, the aforementioned *uniform inevitability* property – some event occurs in all executions at the same time – cannot be expressed by a finite tree automaton. As we will see, it can be expressed in both MIC and FLC. Note that this is a generalization of the property of being bisimilar to a balanced tree – the globally occurring event is just a deadlock in this case.

Games. Closely related to modal logics are games since model checking problems for modal logics often have game-theoretic characterizations. Games in this context are played by two players who push a token along a path through the game arena formed by some product of the underlying structure and the syntax tree of the formula at hand. The logic influences the type of winning condition.

Modal logic for instance induces simple reachability games, while the fixed-point recursion mechanism in the modal μ-calculus requires games with winning conditions on infinite plays, namely parity games [18].

There is often a reverse connection between games and modal logics as well. Game graphs can be seen as labeled transition systems again, and it is reasonable to ask whether the winning regions – the parts from which one of the players has a winning strategy – can in turn be defined by a formula of that logic. This is the case for the modal μ-calculus and parity games.

As the logics considered here are proper extensions of $\mathcal{L}_\mu$, this gives an intuitive explanation of why simple parity games do not suffice to characterize their model checking problems. Instead, an interesting game model for the logics presented here is that of *stair parity games* which are played on the configuration graph of a visibly pushdown system [15]. The name is due to the fact that the parity condition is not evaluated on the whole of a play but only on that part that looks like stairs w.r.t. the stacks involved in these games. We show how the model checking problems for both MIC and FLC can be characterized by stair parity games.

Outline. The paper is organized as follows. Sec. 2 contains preliminary definitions about transition systems and recalls some necessary fixed-point

theory and the modal μ-calculus. In Sec. 3 we then introduce MIC and FLC formally and give examples of formulas defining non-regular properties in these logics. At the end of this section we compare the two logics by giving an overview of the known complexity and expressivity results about them. Sec. 4 then defines stair parity games and shows how to characterize MIC's and FLC's model checking problems by them. We also introduce *backtracking games*, which are non-regular games extending ordinary parity games in a different way. They were originally introduced as game model for inflationary fixed-point logics. Finally, Sec. 5 concludes the paper with some remarks about further research.

2 Preliminaries

Labeled transition systems. For the remainder of this paper we fix a finite non-empty set $\mathcal{A}$ of *actions* and $\mathcal{P}$ of *proposition symbols*.

A *labeled transition system* is a structure $\mathcal{T} := (\mathcal{S}, \{\xrightarrow{a} : a \in \mathcal{A}\}, L)$, where $\mathcal{S}$ is a finite non-empty set of states, $\xrightarrow{a}$ is a binary relation on states for each $a \in \mathcal{A}$, and $L : \mathcal{S} \to 2^{\mathcal{P}}$ is a function labeling each state s with the set of propositions true at s.

Fixed-point theory. Let A be a set and $F : 2^A \to 2^A$ be a function. F is called *monotone* if $F(X) \subseteq F(Y)$ for all $X \subseteq Y \subseteq A$. A *fixed point* of F is any set $P \subseteq A$ such that $F(P) = P$. A *least fixed point of* F is a fixed point that is contained in any other fixed point of F.

It is a consequence of the Knaster-Tarski theorem [19] that every monotone function $F : 2^A \to 2^A$ has a least and a greatest fixed point, written as $\mathbf{lfp}(F)$ and $\mathbf{gfp}(F)$, which can be defined as

$$\mathbf{lfp}(F) \;:=\; \bigcap\{X \subseteq A : F(X) = X\} \;=\; \bigcap\{X \subseteq A : F(X) \subseteq X\},$$

and

$$\mathbf{gfp}(F) \;:=\; \bigcup\{X \subseteq A : F(X) = X\} \;=\; \bigcup\{X \subseteq A : F(X) \supseteq X\}.$$

Least fixed points of monotone operators can also be obtained inductively by the ordinal-indexed sequence X^α of subsets of A defined as

$$X^0 \;:=\; \varnothing\ , \quad X^{\alpha+1} \;:=\; F(X^\alpha)\ , \quad X^\kappa \;:=\; \bigcup_{\alpha<\kappa} X^\alpha$$

where κ is a limit ordinal. As F is monotone, this sequence of sets is increasing, i.e. for all α, β: if $\alpha < \beta$ then $X^\alpha \subseteq X^\beta$, and therefore reaches a fixed point X^∞, with $X^\infty := X^\alpha$ for the least ordinal α such that $X^\alpha = X^{\alpha+1}$. The fixed point X^∞ is called the *inductive fixed point of* F. Again it follows from Knaster and Tarski's theorem that for every monotone operator $F : 2^A \to 2^A$, the least and the inductive fixed point coincide.

Similarly, the greatest fixed point of a monotone operator can also be defined inductively by the following sequence of sets:

$$X^0 \;:=\; A \;, \quad X^{\alpha+1} \;:=\; F(X^\alpha) \;, \quad X^\kappa \;:=\; \bigcap_{\alpha<\kappa} X^\alpha$$

where, again, κ is a limit ordinal.

Least and greatest fixed points are dual to each other. For every operator F define the dual operator $F^d : X \mapsto (F(X^c))^c$ where $X^c := A \setminus X$. If F is monotone, then F^d is also monotone and we have that

$$\mathbf{lfp}(F) = (\mathbf{gfp}(F^d))^c \quad \text{and} \quad \mathbf{gfp}(F) = (\mathbf{lfp}(F^d))^c.$$

The modal μ-calculus. We briefly recall the definition of $\mathcal{L}_\mu$. Let $\mathcal{V}$ be a countable infinite set of variables. The formulas of L_μ are given by the following grammar.

$$\varphi ::= q \mid \neg q \mid X \mid (\varphi \vee \varphi) \mid (\varphi \wedge \varphi) \mid [a]\varphi \mid \langle a\rangle\varphi \mid \mu X.\varphi \mid \nu X.\varphi$$

where $q \in \mathcal{P}$, $a \in \mathcal{A}$, and $X \in \mathcal{V}$. The semantics of $\mathcal{L}_\mu$ is that of basic modal logic where in addition formulas $\mu X.\varphi$ and $\nu X.\varphi$ are interpreted as follows. On any labeled transition system $\mathcal{T}$ with state set $\mathcal{S}$, an $\mathcal{L}_\mu$-formula $\varphi(X)$ with free variable $X \in \mathcal{V}$ induces an operator $F_\varphi : 2^\mathcal{S} \to 2^\mathcal{S}$ which takes a set U of states to the set $[\![\varphi]\!]^\mathcal{T}_{X\mapsto U}$. Here, we write $[\![\varphi]\!]^\mathcal{T}_{X\mapsto U}$ for the set of states from $\mathcal{T}$ at which the formula φ holds under the interpretation that interprets the variable X by the set U. As, by definition, X occurs only positively in φ, this operator is monotone. We define $[\![\mu X.\varphi]\!]^\mathcal{T} := \mathbf{lfp}(F_\varphi)$ and $[\![\nu X.\varphi]\!]^\mathcal{T} := \mathbf{gfp}(F_\varphi)$.

Notation 2.1. Sometimes we want to speak about transitions labeled with any action, and therefore use the abbreviations $\Diamond\varphi := \bigvee_{a\in\mathcal{A}}\langle a\rangle\varphi$, and $\Box\varphi := \bigwedge_{a\in\mathcal{A}}[a]\varphi$. We will also use terms $\mathtt{tt} := q \vee \neg q$, $\mathtt{ff} := q \wedge \neg q$ for some $q \in \mathcal{P}$.

3 Non-Regular Logics

In this section we introduce two extensions of the modal μ-calculus by non-regular constructs. We first recall the *Modal Iteration Calculus*, introduced in [4] which incorporates inflationary fixed points into L_μ. In Sec. 3.2 we then introduce the *Fixed-Point Logic with Chop*, introduced in [16], based on extending $\mathcal{L}_\mu$ by sequential composition. To illustrate the logics and to help comparing them, we exhibit a set of examples and give formalizations for them in both logics.

3.1 The Modal Iteration Calculus

Informally, MIC is propositional modal logic ML, augmented with simultaneous inflationary fixed points.

3.1.1 Syntax and Semantics

Definition 3.1. Let $\mathcal{V}$ be a countable infinite set of variables. The formulas of the *Modal Iteration Calculus* (MIC) are given by the following grammar.

$$\varphi ::= q \mid X \mid \neg\varphi \mid (\varphi \vee \varphi) \mid (\varphi \wedge \varphi) \mid [a]\varphi \mid \langle a\rangle\varphi \mid \mathbf{ifp}X.S \mid \mathbf{dfp}X.S$$

where $X \in \mathcal{V}$, $q \in \mathcal{P}$, $a \in \mathcal{A}$, and

$$S := \begin{cases} X_1 & \leftarrow & \varphi_1 \\ & \vdots & \\ X_k & \leftarrow & \varphi_k \end{cases}$$

is a *system* of rules with $\varphi_j \in$ MIC and $X_i \in \mathcal{V}$ for $1 \leq i \leq k$. If S consists of a single rule $X \leftarrow \varphi$ we simplify the notation and write $\mathbf{ifp}X.\varphi$ instead of $\mathbf{ifp}\ X.\{X \leftarrow \varphi\}$.

We define $Sub(\varphi)$ as the set of subformulas of φ as usual. In particular, the variables X_i occurring on the left-hand side of rules in a system S as above count as subformulas. The semantics of the various operators are as in propositional modal logic with the semantics of **ifp** and **dfp** being as follows. On every transition system $\mathcal{T} := (\mathcal{S}, \{\xrightarrow{a} : a \in \mathcal{A}\}, L)$, the system S defines, for each ordinal α, a tuple $\bar{X}^\alpha = (X_1^\alpha, \ldots, X_k^\alpha)$ of sets of states, via the following inflationary induction:

$$\begin{aligned} X_i^0 &:= \varnothing, \\ X_i^{\alpha+1} &:= X_i^\alpha \cup [\![\varphi_i]\!]^{\mathcal{T}}_{\bar{X}\mapsto\bar{X}^\alpha}, \\ X_i^\kappa &:= \bigcup_{\alpha<\kappa} X_i^\alpha \end{aligned}$$

where κ is a limit ordinal. We call $(X_1^\alpha, \ldots, X_k^\alpha)$ the α-th stage of the inflationary induction of S on $\mathcal{T}$. As the stages are increasing (i.e. $X_i^\alpha \subseteq X_i^\beta$ for any $\alpha < \beta$), this induction reaches a fixed point $(X_1^\infty, \ldots, X_k^\infty)$. Now we put $[\![(\mathbf{ifp}\ X_i : S)]\!]^{\mathcal{T}} := X_i^\infty$.

The semantics of the deflationary fixed-point operator is defined analogously as the i-th component of the deflationary fixed point $(X_1^\infty, \ldots, X_k^\infty)$ obtained from the sequence $X_i^0 := \mathcal{S}$, $X_i^{\alpha+1} := X_i^\alpha \cap [\![\varphi_i]\!]^{\mathcal{T}}_{\bar{X}\mapsto\bar{X}^\alpha}$, and $X_i^\kappa := \bigcap_{\alpha<\kappa} X_i^\alpha$.

3.1.2 Properties Expressible in MIC

We demonstrate the Modal Iteration Calculus by some examples. It is immediately clear from the definition that every $\mathcal{L}_\mu$-formula is equivalent to a MIC-formula (by replacing every μ-operator by **ifp** and ν-operator by **dfp**). We will therefore use least fixed points as well as inflationary fixed points in the examples below.

Example 3.2. Let us first consider the language $\{a^n b^n \mid n \geq 1\}$ mentioned above. We model a finite word by a transition system consisting of a simple path whose edges are labeled by the letters in the word. For example, the word $aabb \in L$ is modeled by the system $\bullet \xrightarrow{a} \bullet \xrightarrow{a} \bullet \xrightarrow{b} \bullet \xrightarrow{b} \bullet$. [1]

Using this encoding of words, the language L can be defined as follows. The formula $\varphi := \langle a\rangle \mathtt{tt} \wedge \mathtt{EF}(\langle b\rangle \mathtt{tt}) \wedge \neg \mathtt{EF}(\langle b\rangle\langle a\rangle \mathtt{tt})$ defines all words starting with an a, containing at least one b, and where all b's come after all a's, i.e. the language a^+b^+. Here, $\mathtt{EF}(\vartheta)$ is the $\mathcal{L}_\mu$ formula $\mu R.(\vartheta \vee \Diamond R)$ saying that a state satisfying φ is reachable. Within a^+b^+ the language L can then be defined by the formula

$$\neg\Big(\mathbf{ifp} Z.\begin{cases} X & \leftarrow \langle b\rangle(\Box \mathtt{ff} \vee X) \\ Y & \leftarrow \langle a\rangle\big(\neg\langle a\rangle \mathtt{tt} \wedge \langle b\rangle \mathtt{tt}\big) \vee \langle a\rangle Y \\ Z & \leftarrow \big(\langle a\rangle Y \ \wedge\ \neg \mathtt{EF}(\langle a\rangle\langle b\rangle\neg X)\big) \ \vee\ \big(\neg\langle a\rangle Y \ \wedge\ \mathtt{EF}(\langle a\rangle\langle b\rangle X)\big) \end{cases}\Big).$$

We demonstrate the evaluation of the fixed points by the following two words $w_1 \in L$ and $w_2 \notin L$.

$w_1 := 1 \xrightarrow{a} 2 \xrightarrow{a} 3 \xrightarrow{b} 4 \xrightarrow{b} 5$ $\qquad$ $w_2 := 1 \xrightarrow{a} 2 \xrightarrow{a} 3 \xrightarrow{a} 4 \xrightarrow{b} 5 \xrightarrow{b} 6$

$$\begin{array}{lcl} X^1 & := & \{4\} \\ Y^1 & := & \{2\} \\ Z^1 & := & \varnothing \end{array} \qquad \begin{array}{lcl} X^1 & := & \{5\} \\ Y^1 & := & \{3\} \\ Z^1 & := & \varnothing \end{array}$$

$$\begin{array}{lcl} X^2 & := & \{3,4\} \\ Y^2 & := & \{1,2\} \\ Z^2 & := & \varnothing \end{array} \qquad \begin{array}{lcl} X^2 & := & \{4,5\} \\ Y^2 & := & \{2,3\} \\ Z^2 & := & \{1\} \end{array}$$

$$\ldots$$

Hence, w_1 satisfies the formula whereas w_2 does not. The idea is that at stage i of the fixed-point induction, X^i contains all states from which a b-labeled path of length i leads to a leaf. To define the induction on Y, let u be a state in $\mathcal{T}$ which has an incoming a-transition but only outgoing b-transitions, i.e.

$$\ldots \bullet \xrightarrow{a} u \xrightarrow{b} \bullet \ldots$$

Note that for words in a^+b^+ this state is unique. The state u is "in the middle" of the word. Then Y^i contains all states from which there is a path to u of length i labeled by a's.

Finally, a state occurs in Z if at some stage i its a-successor is in Y^i but the b-successor of u is not in X^i or vice versa. Hence, the root occurs in Z

[1] There are two common ways of modeling a word by a transition system: labeling edges by letters, as we do it here, or labeling states by the corresponding letters. For MIC, the latter is often more convenient and helps to simplify formulas. For the logic FLC, which we will consider below, the formalization used here is preferable. To unify the examples for MIC and FLC we prefer to use the edge labeling model for both logics.

if the labels of the path leading from the root to the leaf is not a word in $a^n b^n$. ⊣

The example demonstrates a general technique of how counting can be implemented in MIC: we let an induction on a variable X start at a leaf and in each iteration proceed to a predecessor of a state already contained in X. At each stage i, X^i contains the states of distance at most i from a leaf. We can then use a formula $\neg X \wedge \Box X$ to define the states of distance exactly i from a leaf. This techniques is employed in various proofs showing expressibility and complexity results for MIC. We demonstrate it in the following example, where we define the class of transition systems bisimilar to a well-founded tree of finite height. Here the height of a leaf is 0 and the height of an inner node is the maximum height of its successors plus 1.

Example 3.3. Let $\mathcal{T} := (\mathcal{S}, \{\xrightarrow{a} : a \in \mathcal{A}\}, L)$ be a transition system and $s \in \mathcal{S}$. Then $s \in [\![\mu X.\Box X]\!]^{\mathcal{T}}$ if, and only if, there is no infinite path emerging from s, i.e. $(\mathcal{T}, s)$ is bisimilar to a well-founded tree – disregarding labels.

Using a similar trick as in the previous example, we can define all nodes of infinite height in a well-founded tree. For this, consider the formula

$$\varphi \;:=\; \mathbf{ifp}\; Z. \begin{cases} X & \leftarrow \Box X \\ Y & \leftarrow X \\ Z & \leftarrow \big(\Box X \wedge \Diamond \neg Y\big) \vee \Box \mathtt{ff} \end{cases}$$

After $\alpha < \omega$ iterations, the stage X^α contains all nodes of height $< \alpha$ and Y^α contains all nodes of height $< \alpha - 1$. Hence, every node of finite height will at some point have all its successors in X but at least one successor outside of Y (except for the leaves which are included into Z by the disjunct $\Box\mathtt{ff}$) and therefore after ω iterations Z contains all nodes of finite height. However, as $X^\omega = Y^\omega$ a node r of height exactly ω will never occur in Z. Hence, a tree has finite height if, and only if, its root satisfies $\mu X.\Box X \wedge \neg \mathtt{EF}(\neg\varphi)$. ⊣

The next example shows how to define the class of transition systems bisimilar to balanced trees of finite height.[2]

Example 3.4. All that remains is to define in the class of trees of finite height the class of balanced trees. This is done by the formula

$$\neg\Big(\; \mathbf{ifp}\; Y. \begin{cases} X & \leftarrow \Box X \\ Y & \leftarrow \Diamond X \wedge \Diamond \neg X \end{cases} \Big).$$

Again, for $i > 0$, the i-th stage X^i contains all states from which no path of length $\geq i$ emerges. Hence, a state occurs in Y if it has two successors of different length. ⊣

[2] Note that all we can hope for is to define trees of finite height, as finiteness itself is not preserved under bisimulation and hence not definable in *any* modal logic.

Finally, we give an example showing that the class of all transition systems which are bisimilar to a word of finite length is is MIC-definable.

Example 3.5. We have already seen in the previous examples that we can axiomatize transition systems bisimilar to balanced trees of finite height. So all that is left to do is to give a formula that defines in such trees that all paths carry the same labels. This is easily expressed by the formula

$$\neg\Big(\mathbf{ifp}\ Y.\begin{cases} X & \leftarrow \Box X \\ Y & \leftarrow \bigwedge\limits_{a\in\mathcal{A}} \Big(\begin{array}{l} \mathtt{EF}(\neg X \wedge \Box X \wedge \langle a\rangle X) \wedge \\ \mathtt{EF}(\neg X \wedge \Box X \wedge \bigvee\limits_{b\in\mathcal{A},a\neq b} \langle b\rangle X) \end{array} \Big) \end{cases}\Big).$$

⊣

Using similar tricks we can express buffer underflow in finite words, i.e. the context-free language

$$\mathcal{L} := \{w \in \{a,b\}^* \mid \forall u,v : w = uv \Rightarrow |u|_b \leq |u|_a\}.$$

Here, $|u|_b$ denotes the number of b's in the word u and likewise for $|u|_a$. It is not known, however, whether the "buffer underflow" and "bisimilarity-to-a-word" formulas can be amended for infinite words as well. The problem is that there no longer is a natural starting point for fixed-point inductions.

3.2 Fixed-Point Logic with Chop

We proceed by introducing a different extension of the modal μ-calculus. It differs from MIC in that we again consider monotone inductions only, but the individual fixed-point stages are no longer sets of states but monotone functions from the complete lattice of all monotone operators over the state space.

3.2.1 Syntax and Semantics

Let $\mathcal{P}$ and $\mathcal{A}$ be as before, and $\mathcal{V}$ be a countable infinite set of variable names. Formulas of *Fixed-Point Logic with Chop* (FLC) over $\mathcal{P}$, $\mathcal{A}$ and $\mathcal{V}$ are given by the following grammar.

$$\varphi \ ::= \ q \mid \neg q \mid X \mid \tau \mid \langle a\rangle \mid [a] \mid (\varphi \vee \varphi) \mid (\varphi \wedge \varphi) \mid (\varphi;\varphi) \mid \mu X.\varphi \mid \nu X.\varphi$$

where $q \in \mathcal{P}$, $a \in \mathcal{A}$, and $X \in \mathcal{V}$. We will write σ for either μ or ν. In the following, we will also omit parentheses and introduce the convention that ";" binds stronger than the Boolean operators which, in turn, bind stronger than fixed-point quantifiers.

The set of subformulas $Sub(\varphi)$ of an FLC formula φ is defined as usual, for example $Sub(\sigma X.\varphi) = \{\sigma X.\varphi\} \cup Sub(\varphi)$, etc. Also, we assume that

variables are quantified at most once in each formula. Hence, each φ comes with a function fp_φ which associates to each variable X in $Sub(\varphi)$ its defining fixed-point function $\sigma X.\psi$.

FLC extends the modal μ-calculus $\mathcal{L}_\mu$ with the sequential composition ("chop") operator $\_;\_$. Remember that variables in $\mathcal{L}_\mu$ formulas can only occur in rightmost positions within Boolean formulas, possibly prefixed by modal operators. This gives an intuitive explanation of the fact that the expressive power of $\mathcal{L}_\mu$ is restricted to regular languages of infinite words or trees – formulas of $\mathcal{L}_\mu$ resemble (alternating) right-linear grammars with modal operators as terminal symbols.

Variables in FLC formulas, however, can also be suffixed with modal operators through the use of sequential composition, e.g. $\langle a\rangle; X; \langle b\rangle$. Since this is supposed to generalize the restricted composition of modal operators with formulas on the right, there is no need to include formulas of the form $\langle a\rangle\varphi$ in FLC. Instead, this is supposed to be simulated by $\langle a\rangle;\varphi$, and this is why modal operators are chosen as atomic formulas in FLC.

The semantics of the modal μ-calculus cannot simply be extended by clauses for the additional operators in FLC, in particular not for sequential composition. Remember that the semantics of $\mathcal{L}_\mu$ assigns to each formula and environment interpreting its free variables a *set of states* of the underlying LTS. In other words, the semantics of a $\mathcal{L}_\mu$ formula is a *predicate*.

In order to interpret sequential composition naturally, the semantics of FLC lifts the $\mathcal{L}_\mu$ semantics to the space of monotone functions of type $2^\mathcal{S} \to 2^\mathcal{S}$, where $\mathcal{S}$ is the state space of an LTS. Hence, FLC formulas get interpreted by *predicate transformers*. This allows sequential composition to be interpreted naturally using function composition.

Let $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} \mid a \in \mathcal{A}\}, L)$ be an LTS, and

$$2^\mathcal{S} \rightarrowtail 2^\mathcal{S} \;:=\; \{f : 2^\mathcal{S} \to 2^\mathcal{S} \mid \forall S, T \subseteq \mathcal{S} : \text{ if } S \subseteq T \text{ then } f(S) \subseteq f(T)\}$$

be the set of all monotone predicate transformers over $\mathcal{T}$. This can be ordered partially by the well-known pointwise order

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

In fact, $(2^\mathcal{S} \rightarrowtail 2^\mathcal{S}, \sqsubseteq)$ forms a complete lattice with top and bottom elements $\top = \lambda T.\mathcal{S}$, $\bot = \lambda T.\varnothing$, as well as meets and joins $\bigsqcap$, $\bigsqcup$. The following is easily verified. Let $f_i \in 2^\mathcal{S} \rightarrowtail 2^\mathcal{S}$, $i \in I$ for some set of indices I. Then

$$\bigsqcap_{i\in I} f_i \;:=\; \lambda T.\bigcap_{i\in I} f_i(T) \qquad \bigsqcup_{i\in I} f_i \;:=\; \lambda T.\bigcup_{i\in I} f_i(T)$$

are monotone too, and form the infimum, resp. supremum of $\{f_i \mid i \in I\}$ in $2^\mathcal{S} \rightarrowtail 2^\mathcal{S}$.

This function space will now act as the domain of interpretation for FLC formulas. A formula $\varphi(X)$ with a free variable X gives rise to a second-order function $F_\varphi : (2^{\mathcal{S}} \rightarrowtail 2^{\mathcal{S}}) \to (2^{\mathcal{S}} \rightarrowtail 2^{\mathcal{S}})$ which is monotone itself w.r.t. the partial order $\sqsubseteq$. According to the Knaster-Tarski Theorem, least and greatest fixed points of such second-order functions exist uniquely in $2^{\mathcal{S}} \rightarrowtail 2^{\mathcal{S}}$ and can be used to give meaning to formulas with fixed-point quantifiers just as it is done in the modal μ-calculus and first-order functions.

Let $\rho : \mathcal{V} \to (2^{\mathcal{S}} \rightarrowtail 2^{\mathcal{S}})$ be an environment interpreting (free) variables by monotone predicate transformers. As usual, we write $\rho[X \mapsto f]$ to denote the environment that maps X to f and agrees with ρ on all other arguments. The semantics of an FLC formula w.r.t. an underlying LTS and the environment ρ is defined inductively as follows.

$$
\begin{aligned}
[\![q]\!]^{\mathcal{T}}_\rho &:= \lambda T.\{s \in \mathcal{S} \mid q \in L(s)\} \\
[\![\neg q]\!]^{\mathcal{T}}_\rho &:= \lambda T.\{s \in \mathcal{S} \mid q \notin L(s)\} \\
[\![X]\!]^{\mathcal{T}}_\rho &:= \rho(X) \\
[\![\tau]\!]^{\mathcal{T}}_\rho &:= \lambda T.T \\
[\![\langle a \rangle]\!]^{\mathcal{T}}_\rho &:= \lambda T.\{s \in \mathcal{S} \mid \exists t \in \mathcal{S}, \text{ s.t. } s \xrightarrow{a} t \text{ and } t \in T\} \\
[\![[a]]\!]^{\mathcal{T}}_\rho &:= \lambda T.\{s \in \mathcal{S} \mid \forall t \in \mathcal{S} : \text{ if } s \xrightarrow{a} t \text{ then } t \in T\} \\
[\![\varphi \vee \psi]\!]^{\mathcal{T}}_\rho &:= [\![\varphi]\!]^{\mathcal{T}}_\rho \sqcup [\![\psi]\!]^{\mathcal{T}}_\rho \\
[\![\varphi \wedge \psi]\!]^{\mathcal{T}}_\rho &:= [\![\varphi]\!]^{\mathcal{T}}_\rho \sqcap [\![\psi]\!]^{\mathcal{T}}_\rho \\
[\![\varphi ; \psi]\!]^{\mathcal{T}}_\rho &:= \lambda T.[\![\varphi]\!]^{\mathcal{T}}_\rho \left([\![\psi]\!]^{\mathcal{T}}_\rho (T)\right) \\
[\![\mu X.\varphi]\!]^{\mathcal{T}}_\rho &:= \bigsqcap \{f \in 2^{\mathcal{S}} \rightarrowtail 2^{\mathcal{S}} \mid [\![\varphi]\!]^{\mathcal{T}}_{\rho[X \mapsto f]} \sqsubseteq f\} \\
[\![\nu X.\varphi]\!]^{\mathcal{T}}_\rho &:= \bigsqcup \{f \in 2^{\mathcal{S}} \rightarrowtail 2^{\mathcal{S}} \mid f \sqsubseteq [\![\varphi]\!]^{\mathcal{T}}_{\rho[X \mapsto f]}\}
\end{aligned}
$$

Thus, the operators of FLC are simply translated into related operators on the lattice structure of the function space $2^{\mathcal{S}} \rightarrowtail 2^{\mathcal{S}}$ with τ being the identity function as the neutral element of the sequential composition operator.

Since FLC is supposed to be a program logic, it is necessary to explain when a single state satisfies a (closed) formula of FLC. Note that in the case of the modal μ-calculus this is simply done using the element relation on the semantics of the formula. This is clearly not possible if the semantics is a function. The usual *models*-relation is therefore – by arbitrary choice – defined as follows. Let $\mathcal{T}$ be an LTS with state set $\mathcal{S}$ and $s \in \mathcal{S}$.

$$\mathcal{T}, s \models_\rho \varphi \quad \text{iff} \quad s \in [\![\varphi]\!]^{\mathcal{T}}_\rho(\mathcal{S})$$

This gives rise to two different equivalence relations in FLC: two formulas φ and ψ are *strongly equivalent* if they have the same semantics.

$$\varphi \equiv \psi \quad \text{iff} \quad \text{for all LTS } \mathcal{T} \text{ and all } \rho : \mathcal{V} \to (2^{\mathcal{S}} \rightarrowtail 2^{\mathcal{S}}) : [\![\varphi]\!]^{\mathcal{T}}_\rho = [\![\psi]\!]^{\mathcal{T}}_\rho$$

On the other hand, they are *weakly equivalent* if they are satisfied by the same set of states in any LTS.

$$\varphi \approx \psi \quad \text{iff} \quad \text{for all LTS } \mathcal{T} \text{ with state set } \mathcal{S} \text{ and all } \rho : \mathcal{V} \to (2^{\mathcal{S}} \rightarrowtail 2^{\mathcal{S}}) : \\ [\![\varphi]\!]^{\mathcal{T}}_{\rho}(\mathcal{S}) = [\![\psi]\!]^{\mathcal{T}}_{\rho}(\mathcal{S})$$

Clearly, strong equivalence is at least as strong as weak equivalence: $\equiv \subseteq \approx$ – two functions that agree on all arguments certainly agree on a particular one. Here we are mainly interested in weak equivalence because it formalizes "expressing the same property", and it is therefore the right notion for comparing FLC to other logics like MIC w.r.t. expressive power.

3.2.2 Properties Expressible in FLC

In the following we want to exemplify the use of FLC by formalizing a few non-regular properties.

Example 3.6. Consider the language $L = \{a^n b^n \mid n \geq 1\}$ again. It is generated by the context-free grammar with productions

$$S \;\to\; ab \mid aSb$$

FLC can express the property "there is a maximal path whose label forms a word in L".

$$(\mu X.\langle a\rangle; \langle b\rangle \vee \langle a\rangle; X; \langle b\rangle); \Box; \mathtt{ff}$$

Notice the apparent similarity to the grammar above.

To illustrate the semantics of FLC formulas, we give the first few stages X^i of the fixed-point iteration for the subformula $\mu X.\langle a\rangle; \langle b\rangle \vee \langle a\rangle; X; \langle b\rangle$.

$$\begin{array}{lcl} X^0 & := & \lambda T.\varnothing \\ X^1 & := & \lambda T.[\![\langle a\rangle]\!]([\![\langle b\rangle]\!](T)) \cup [\![\langle a\rangle]\!](X^0([\![\langle b\rangle]\!](T))) = \lambda T.[\![\langle a\rangle]\!]([\![\langle b\rangle]\!](T)) \\ X^2 & := & \lambda T.[\![\langle a\rangle]\!]([\![\langle b\rangle]\!](T)) \cup [\![\langle a\rangle]\!](X^1([\![\langle b\rangle]\!](T))) \\ & = & \lambda T.[\![\langle a\rangle]\!]([\![\langle b\rangle]\!](T)) \cup [\![\langle a\rangle]\!]([\![\langle a\rangle]\!]([\![\langle b\rangle]\!]([\![\langle b\rangle]\!](T)))) \\ & \vdots & \end{array}$$

In general, X^i is the function taking any set T to the set of states from which there is a path to a state in T under any of the words $\{ab, aabb, \ldots, a^i b^i\}$. Hence, $[\![(\mu X.\langle a\rangle; \langle b\rangle \vee \langle a\rangle; X; \langle b\rangle); \Box; \mathtt{ff}]\!]$ takes any set T to the set of nodes from which a node without successors can be reached by some $a^n b^n$-path. ⊣

Example 3.7. FLC can, like MIC, axiomatize the tree of height (at least) ω upto bisimulation. Again, we first say that there is no infinite path utilizing the $\mathcal{L}_\mu$ formula $\mu X.\Box X$.

$$\varphi_{fin} \;:=\; \mu X.\Box; X$$

Then we need to say that there are paths of unbounded length.

$$\varphi_{unb} \;:=\; (\nu X.\tau \wedge X; \Diamond); \mathtt{tt}$$

Note that, by unfolding, this is equivalent to $\bigwedge_{n\in\mathbb{N}} \Diamond^n; \mathtt{tt}$.

The following then expresses that a transition system is bisimilar to a tree of height exactly ω.

$$\varphi_{fin} \wedge \varphi_{unb} \wedge \Box\overline{\varphi}_{unb}$$

where the latter is supposed to express the complement of the respective unboundedness property. It can be obtained straight-forwardly as $\overline{\varphi}_{unb} := (\mu X.\tau \vee X; \Box); \mathtt{ff}$. ⊣

Recall that MIC can express bisimilarity to a finite word but possibly not to an infinite one. In FLC it does not seem to be possible to express either of these, and the problem is not to do with (in-)finiteness. For FLC the difficulty is to speak about two different paths. Note that an LTS $\mathcal{T}$ with starting state s is not bisimilar to any linear model if, and only if, there are two different actions a and b and a natural number $n \geq 0$, s.t.

$$\mathcal{T}, s \models \underbrace{\Diamond \ldots \Diamond}_{n\text{times}}\langle a\rangle\mathtt{tt} \wedge \underbrace{\Diamond \ldots \Diamond}_{n\text{times}}\langle b\rangle\mathtt{tt}$$

The model checking games for FLC presented below will give an idea of why the existence of such an n cannot be expressed in FLC. Essentially, non-bisimilarity could be decided using two stacks of formulas which would be used by one player to build the two conjuncts while the other player then decides which formula to prove in a given model. The FLC model checking games, however, only provide a single stack.

Example 3.8. Consider the related but simpler property of bisimilarity to a balanced tree. While non-bisimilarity to a word can be characterized in meta-logic using quantification over three different sorts – there are actions a and b and a natural number n, s.t. there are paths of length n ending in a, resp. b – describing that an LTS looks like a balanced tree only needs two: there are $n, m \in \mathbb{N}$ s.t. $n < m$ and two paths, one of which is maximal and has length n, the other has length m.

Take the FLC formula

$$\big(\mu X.\tau \vee X; (\Diamond; \mathtt{tt} \wedge \Box)\big); \Box; \mathtt{ff}$$

Now note that for any FLC formula φ we have $\mathtt{tt}; \varphi \equiv \mathtt{tt}$. Let $\Phi := \mu X.\tau \vee X; (\Diamond; \mathtt{tt} \wedge \Box)$. Unfolding the formula above and rewriting it using some basic equalities yields:

$$\Phi; \Box\mathtt{ff} \;\equiv\; \big(\tau \vee \Phi; (\Diamond\mathtt{tt} \wedge \Box)\big); \Box\mathtt{ff}$$

$$\begin{aligned}
&\equiv \Box\mathtt{ff} \vee \Phi; (\Diamond\mathtt{tt} \wedge \Box; \Box\mathtt{ff}) \\
&\equiv \Box\mathtt{ff} \vee \big(\tau \vee \Phi; (\Diamond\mathtt{tt} \wedge \Box)\big); (\Diamond\mathtt{tt} \wedge \Box\Box\mathtt{ff}) \\
&\equiv \Box\mathtt{ff} \vee (\Diamond\mathtt{tt} \wedge \Box\Box\mathtt{ff}) \vee \Phi; (\Diamond\mathtt{tt} \wedge \Box\Diamond\mathtt{tt} \wedge \Box\Box\Box\mathtt{ff}) \\
&\equiv \ldots
\end{aligned}$$

The i-th unfolding of this formula asserts that there is a path of length i, all paths of length less than i can be extended by at least one state, but there is no path of length $i+1$. Hence, the union over all these approximations defines exactly the class of all balanced trees of finite height. ⊣

There is a straight-forward translation $\Phi_{\langle\cdot\rangle} : \mathrm{CFG} \to \mathrm{FLC}$ which assigns to each context-free grammar G an FLC formula $\Phi_{\langle G\rangle}$ s.t.

$$\mathcal{T}, s \models \Phi_{\langle G\rangle} \quad \text{iff} \quad \text{there is a } t \in \mathcal{S} \text{ and a } w \in L(G) \text{ s.t. } s \xrightarrow{w} t$$

where $L(G)$ denotes, as usual, the context-free language generated by G. Φ_G simply is the uniform translation of G which replaces nonterminals in the grammars with μ-quantified FLC variables, concatenation with sequential composition, and alternatives between rules with disjunctions [13].

However, there is no translation $\Phi_{[\cdot]} : \mathrm{CFG} \to \mathrm{FLC}$ s.t. for all LTS $\mathcal{T}$ with state set $\mathcal{S}$ and all $s \in \mathcal{S}$ we have

$$\mathcal{T}, s \models \Phi_{[G]} \quad \text{iff} \quad \text{for all } t \in \mathcal{S} \text{ and all } w \in \mathcal{A}^*\text{: if } s \xrightarrow{w} t \text{ then } w \in L(G)$$

Such a translation would contradict the decidability of FLC's model checking problem by a simple reduction from the universality problem for context-free languages.

Example 3.9. Finally, we consider the property of not doing more *returns* than *calls*, i.e. we want to specify a tree (or transition system) in which all paths, including non-maximal ones, are labeled by a word from the language $L = \{w \in \{a,b\}^* \mid \forall v \preceq w : |v|_b \leq |v|_a\}$ where $\preceq$ denotes the prefix relation on words, and $|v|_a$ stands for the number of occurrences of the letter a in v.

This language is context-free, and for example generated by the grammar G

$$S \;\to\; aTS \mid \epsilon \qquad T \;\to\; b \mid aTT \mid \epsilon$$

The specification would be completed if there was a translation $\Phi_{[\cdot]}$ as mentioned above. Even though this cannot exist, the desired property can still be specified in FLC. Note that the language $L(G)$ of all non-underflowing buffer runs is *deterministic context-free*, and its complement is generated by the grammar G' defined as

$$S \;\to\; bU \mid aTbU \qquad U \;\to\; \epsilon \mid aU \mid bU \qquad T \;\to\; b \mid aTT$$

This can then be transformed into the FLC formula $\Phi_{\langle G' \rangle}$ and consecutively be complemented to obtain

$$\varphi \;:=\; [b];\mathtt{ff} \wedge [a];\big(\nu T.[b] \wedge [a];T;T\big);[b];\mathtt{ff}$$

which expresses lack of buffer underflows on all runs. ⊣

3.3 Complexity and Expressive Power

First of all, it is not hard to see that both MIC and FLC are genuine extensions of the modal μ-calculus w.r.t. expressive power.

Proposition 3.10 ([4, 16])**.** $\mathcal{L}_\mu \lneq \text{MIC}$, $\mathcal{L}_\mu \lneq \text{FLC}$.

It is obvious that any $\mathcal{L}_\mu$-formula is equivalent to a formula in MIC – simply replace μ-operators by **ifp**- and ν- by **dfp**-operators. Furthermore, $\mathcal{L}_\mu$ translates into FLC almost as easily. Simply replace every $\langle a \rangle \varphi$ by $\langle a \rangle;\varphi$, and every $[a]\varphi$ by $[a];\varphi$.

The strictness of both inclusions immediately follows from the previous examples showing how to express certain non-regular properties in these logics.

Related to this is also the loss of the finite model property compared to $\mathcal{L}_\mu$, as already shown in Ex. 3.3 and 3.7.

Proposition 3.11 ([4, 16])**.** Both MIC and FLC do not have the finite model property.

The tree model property, however, can be proved by embedding MIC and FLC into infinitary modal logic. This is particularly simple for MIC where it only requires fixed-point elimination.

For every FLC formula φ we obtain a formula φ' of infinitary modal logic s.t. $\varphi \approx \varphi'$ by eliminating fixed points first, then followed by the elimination of sequential composition and the formula τ. This is possible because $\varphi \approx \varphi;\mathtt{tt}$, and can easily be done by successively pushing sequential composition inwards from the right.

Proposition 3.12 ([4, 14])**.** Both MIC and FLC are invariant under bisimulation and, hence, have the tree-model property.

Not much is known about the expressive power of each of these logics relative to other formalisms like Predicate Logic, or – when restricted to word models – formal grammars and automata. For MIC, it is known that it is not the bisimulation-invariant fragment of monadic inflationary fixed-point logic, which would have been the natural candidate as $\mathcal{L}_\mu$ is the bisimulation-invariant fragment of monadic least fixed-point logic. As to grammars and automata, FLC is slightly easier to compare in this respect because of the similarity between formulas and context-free grammars. Also, the characterisation of least and greatest fixed points by the

Knaster-Tarski Theorem gives a straight-forward embedding of FLC into Third-Order Logic. To gain a good intuition about the expressive power of temporal logics, however, it is often useful to consider word or well-founded models.

Proposition 3.13 ([4, 10])**.** When interpreted over word models only,

i) there is a language that is not context-free but definable in MIC.

ii) FLC is equi-expressive to alternating context-free languages.[3]

iii) every language in DTime($\mathcal{O}(n)$) is definable in MIC.

iv) every language definable in MIC or FLC is in DSpace($\mathcal{O}(n)$), i.e. deterministic context-sensitive.

As usual, great expressive power also comes at a price. One can show that arithmetic is expressible in MIC on trees of height ω, i.e. the tree unraveling of the ordinal ω. For this, a natural number $n \in \mathbb{N}$ is identified with the set of nodes of height at most n. Then, arithmetic on the height of nodes can be shown to be definable in MIC. By doing so, one can translate any first-order sentence ψ over the arithmetic $\mathcal{N} := (\mathbb{N}, <, +, \cdot)$ into a MIC-formula ψ^* such that $\mathcal{N} \models \psi$ if, and only if, ψ^* is satisfiable. Here, ψ^* enforces its models to be bisimilar to a tree of height ω and encodes the arithmetical sentence ψ on such trees. This immediately implies undecidability.

Satisfiability in FLC is undecidable as well. This was first shown by Müller-Olm using a reduction from the simulation equivalence problem for context-free processes [16]. An embedding of Propositional Dynamic Logic of Non-Regular Programs, however, yields a quantitatively similar result as the one for MIC.

Proposition 3.14 ([4, 13])**.** The satisfiability problem for both MIC and FLC is undecidable. They are not even in the arithmetical hierarchy.

Concerning the model checking complexity, it is easily seen that a naïve evaluation of MIC-formulas by iteratively computing the stages of the fixed-point inductions leads to an algorithm that correctly checks whether a given

[3] These are generated by *alternating context-free grammars* which enrich ordinary context-free grammars by two types of non-terminals: *existential* and *universal* ones. While the generation of sentential forms and, thus, words for existential non-terminals is the usual one, universal non-terminals derive a sentential form only if *all* (rather than any) of their productions derive it. There are various ways of defining a precise semantics that captures this idea. Alternating Context-Free Grammars as defined by the second author [10] are in fact the same as Conjunctive Grammars by Okhotin [17]. There are also presumably non-equivalent models like the grammars by Moriya [8].

MIC-formula φ is true in a given transition system $\mathcal{T}$ in time $\mathcal{O}(|\mathcal{T}|^{|\varphi|})$ and space $\mathcal{O}(|\mathcal{T}| \cdot |\varphi|)$. It is therefore in P whenever the formula is fixed. It is, however, PSPACE-hard already on a fixed 1-state transition system if the formula is part of the input.

FLC differs from MIC w.r.t. model checking. First of all, fixed-point approximations can be exponentially long in the size of the transition system [12]. FLC can even express problems which are hard for deterministic exponential time, namely Walukiewicz's Pushdown Game problem [21].

An upper bound of deterministic exponential time is not immediately seen. Note that naïve fixed-point iteration in the function space $2^{\mathcal{S}} \rightarrowtail 2^{\mathcal{S}}$ would lead to a doubly exponential procedure. But remember that model checking in FLC means that the value of a function on a particular argument, namely $\mathcal{S}$, needs to be computed rather than the entire function itself. This observation leads – with the aid of stair parity games, see below – to a singly exponential model checking algorithm.

Proposition 3.15 ([4, 3, 12])**.** The combined complexity of the model checking problem for

i) MIC is PSPACE-complete,

ii) FLC is EXPTIME-complete.

Regarding the data complexity, we have the following result.

iii) For every fixed formula the model checking complexity of MIC is in P.

iv) There are fixed FLC formulas for which the model checking problem is EXPTIME-hard.

Regarding the expression complexity, we have the following results.

v) Model checking MIC on a fixed transition system is PSPACE-complete.

PSPACE-hardness of the expression complexity is obtained by a reduction from QBF, the problem to decide if a given quantified boolean formula is satisfiable. It can easily be reduced to the model checking problem of MIC on a trivial transition system consisting of one state only.

The only lower bound for the expression complexity of FLC that is currently known is P-hardness trivially inherited from $\mathcal{L}_\mu$ [6].

An interesting question for non-regular logics is decidability of the model checking problem over infinite state systems. The known results there are negative.

Proposition 3.16 ([16, 14])**.** The model checking problem for FLC over the class of normed deterministic BPA processes is undecidable.

The proof uses the fact that characteristic formulas for simulation (equivalence) of BPA processes can easily be constructed in FLC. It is currently not known whether or not MIC has a decidable model checking problem over the class of context-free processes.

Finally, we can use the model checking complexity results to prove an inexpressibility theorem, and partially separate MIC and FLC in expressive power.

Theorem 3.17. FLC $\not\leq$ MIC.

Proof. Take an FLC formula φ whose set of models is EXPTIME-hard according to Prop. 3.15 (*iv*). Suppose FLC $\leq$ MIC. Then there would be a MIC formula φ' with the same set of models. However, according to Prop. 3.15 (*iii*), this set would also have to be in P, and we would have P=EXPTIME which is not the case. Q.E.D.

It is not yet known whether every MIC-definable property is also FLC definable or whether the two logics are incomparable w.r.t. expressive power. We suspect that the latter is the case. The difficulty in establishing this as a theorem though is the lack of machinery for showing inexpressibility in FLC.

4 Non-Regular Games

There are many ways of extending ordinary parity games. One option, which we will consider first, is to introduce the concept of stacks to the games. Formally, these games are played on configuration graphs of pushdown processes. In this approach we increase the modeling power of the game arenas while keeping the traditional way of playing games, i.e. the two players push a token along paths in the game arena and the priorities of this path determine the winner.

A different approach is to stick to standard parity game arenas but change the way the games are played. This approach is taken in the concept of backtracking games, where a play no longer is a path through the arena but defines a complex subgraph.

4.1 Stair Parity Games

A *pushdown alphabet* $\mathcal{A}$ is a tuple $(\mathcal{A}_c, \mathcal{A}_r, \mathcal{A}_i)$ consisting of three disjoint finite alphabets, a finite set $\mathcal{A}_c$ of *calls*, a finite set $\mathcal{A}_r$ of *returns* and a finite set $\mathcal{A}_i$ of internal states.

Definition 4.1. Let $\mathcal{A} := (\mathcal{A}_c, \mathcal{A}_r, \mathcal{A}_i)$ be a pushdown alphabet. A *visibly pushdown system* (VPS) over $(\mathcal{A}_c, \mathcal{A}_r, \mathcal{A}_i)$ is a tuple $\mathcal{B} = (Q, \mathcal{A}, \Gamma, \delta)$ where Q is a finite set of states, and Γ is a finite stack alphabet. We simply write

$\Gamma_\perp$ for $\Gamma \cup \{\perp\}$ assuming that Γ itself does not contain the special stack bottom symbol $\perp$. Finally, $\delta = \delta_c \cup \delta_r \cup \delta_i$ is the transition relation with

$$\begin{aligned} \delta_c &\subseteq Q \times \mathcal{A}_c \times Q \times \Gamma \\ \delta_r &\subseteq Q \times \mathcal{A}_r \times \Gamma_\perp \times Q \\ \delta_i &\subseteq Q \times \mathcal{A}_i \times Q \end{aligned}$$

A transition (q, a, q', γ), where $a \in \mathcal{A}_c$, means that if the system is in the control state q and reads an a, it can change its state to q' and push the symbol γ onto the stack. Similarly, upon a transition (q, a, q', γ), where $a \in \mathcal{A}_r$, it reads γ from the top of the stack (and pops it unless $\gamma = \perp$) and changes its state from q to q'. Transitions reading $a \in \mathcal{A}_i$ are *internal* transitions that do not change the stack.

We now turn to defining stair parity games, which are parity games played on the configuration graph of visibly pushdown systems with a slightly modified winning condition.

Definition 4.2. A *stair parity game* (SPG) over a VPS $\mathcal{B}$ is a tuple $\mathcal{G}_\mathcal{B} = (V, v_0, Q_\exists, Q_\forall, E, \Omega)$ such that

- $V := Q \times \Gamma^*\{\perp\}$ is the set of nodes in this game,
- $v_0 \in V$ is a designated starting node,
- Q is partitioned into $Q_\exists$ and $Q_\forall$,
- $E \subseteq V \times V$ consists of edges $\big((q, \delta), (q', \delta')\big)$ s.t.
  - there is a $(q, a, q', \gamma) \in \delta_c$ and $\delta' = \gamma\delta$, or
  - there is a $(q, a, \gamma, q') \in \delta_r$ and $\delta = \gamma\delta'$, or
  - there is a $(q, a, q') \in \delta_i$ and $\delta' = \delta$.
- $\Omega : Q \to \mathbb{N}$ assigns to each node a priority.

For simplicity we assume that SPGs always are total, i.e. every node has an outgoing edge. A play in such a game is, as usual, an infinite sequence of nodes. It is played starting in v_0, and continued by a choice along an outgoing edge of that player who owns the last visited node. Unlike the case of parity games, the winner is not determined by the least or greatest priority occurring infinitely often in a play. Instead, one only considers those nodes that form stairs, i.e. nodes with a stack that persists for the entire remainder of the play.

Definition 4.3. Let $\mathcal{G}_\mathcal{B} = (V, v_0, Q_\exists, Q_\forall, E, \Omega)$ be a SPG over a VPS $\mathcal{B}$, and let $\pi = v_0, v_1, v_2, \ldots$ be an infinite play of this game s.t. $v_i = (q_i, \delta_i)$ for all $i \in \mathbb{N}$.

$$\frac{s \vdash (\varphi_0 \vee \varphi_1)\ ;\ \delta}{s \vdash \varphi_i\ ;\ \delta}\ \exists i \in \{0,1\} \qquad \frac{s \vdash (\varphi_0 \wedge \varphi_1)\ ;\ \delta}{s \vdash \varphi_i\ ;\ \delta}\ \forall i \in \{0,1\}$$

$$\frac{s \vdash (\sigma X.\varphi)\ ;\ \delta}{s \vdash X\ ;\ \delta} \qquad \frac{s \vdash X\ ;\ \delta}{s \vdash \varphi\ ;\ \delta}\ \text{if } fp(X) = \sigma X.\varphi \qquad \frac{s \vdash (\varphi;\psi)\ ;\ \delta}{s \vdash \varphi\ ;\ (\psi;\delta)}$$

$$\frac{s \vdash \tau\ ;\ (\varphi;\delta)}{s \vdash \varphi\ ;\ \delta} \qquad \frac{s \vdash \langle a \rangle\ ;\ (\varphi;\delta)}{t \vdash \varphi\ ;\ \delta}\ \exists s \xrightarrow{a} t \qquad \frac{s \vdash [a]\ ;\ (\varphi;\delta)}{t \vdash \varphi\ ;\ \delta}\ \forall s \xrightarrow{a} t$$

FIGURE 1. The rules of the FLC model checking games.

Define $Steps(\pi) = \{i \in \mathbb{N} : \forall j \geq i\ |\delta_j| \geq |\delta_i|\}$ where $|\delta|$ denotes the length of the stack δ. Note that $|Steps(\pi)| = \infty$ whenever π is infinite.

Player $\exists$ wins the play π if, and only if, $\max\{c$: there are infinitely many q_i with $i \in Steps(\pi)$ and $\Omega(q_i) = c\}$ is even. Otherwise, Player $\forall$ is the winner of π.

The *stair parity game problem* is: given a SPG $(V, v_0, Q_\exists, Q_\forall, E, \Omega)$, decide whether or not Player $\exists$ has a winning strategy from node v_0 in this game. It can be shown that such games are determined, and that this problem is decidable. In fact, a reduction to an exponentially large ordinary parity game yields a moderate upper complexity bound.

Theorem 4.4 ([15])**.** The stair parity game problem can be decided in EXPTIME.

4.2 A Game-Theoretic Characterization of FLC

Let $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} : a \in \mathcal{A}\}, L)$ be an LTS, $s_0 \in \mathcal{S}$ and Φ be a closed FLC formula. The model checking game $\mathcal{G}_\mathcal{T}(s_0, \Phi)$ is played between Players $\exists$ and Player $\forall$ in order to establish whether or not $\mathcal{T}, s_0 \models \Phi$ holds. The set of configurations is $\mathcal{C} := \mathcal{S} \times Sub(\varphi) \times Sub(\varphi)^*$. We usually write configurations in the form $s \vdash \varphi\ ;\ \delta$ where $\delta = \psi_1; \ldots; \psi_k$ acts as a stack of FLC formulas with its top on the left. The formula φ will in this case also be called the *principal formula* of this configuration.

The intuitive meaning of such a configuration is the following. Player $\exists$ wants to show that $s \in [\![\varphi;\delta]\!]^\mathcal{T}_\rho(\mathcal{S})$ holds under a ρ which interprets the free variables in $\varphi;\delta$ by suitable approximants.

The initial configuration is $s_0 \vdash \Phi\ ;\ \mathtt{tt}$ – remember that $\Phi \approx \Phi;\mathtt{tt}$. The rules of the model checking game are shown in Fig. 1.

The idea underlying these games is to defer the examination of ψ in a formula $\varphi;\psi$ and to first consider whether or not φ determines the winner

already. This is in contrast to the Boolean binary constructs $\wedge$ and $\vee$ in which both operands have equal importance. However, this is not the case for the sequential composition operator. A natural choice would be to let Player $\exists$ provide a witness for the chop (a set of states for example) and then to let Player $\forall$ respond by choosing either of the composed subformulas. This is not sound though, as the following example shows.

Example 4.5. Let $\Phi = \nu X.\mu Y.X;Y$. The exact meaning of this rather simple formula is not too difficult to guess. It can also be computed using fixed-point iteration on an imaginary model with state set $\mathcal{S}$. Remember that $\top$ and $\bot$ are the top- and bottom-elements in the function lattice $2^{\mathcal{S}} \rightarrowtail 2^{\mathcal{S}}$. At the beginning, the ν-quantified X gets mapped to $\top$, and in the inner fixed-point iteration, Y gets mapped to $\bot$. We use the symbol $\circ$ to denote function composition semantically as opposed to the syntactical operator ";".

$$
\begin{array}{lllllllll}
X^0 & := & \top & & & & & & \\
 & & & Y^{00} & := & \bot & & & \\
 & & & Y^{01} & := & X^0 \circ Y^{00} & = & \top & \\
 & & & Y^{02} & := & X^0 \circ Y^{01} & = & \top & = \; Y^{01} \\
X^1 & := & Y^{02} & = & \top & & & & \\
 & & & Y^{10} & := & \bot & & & \\
 & & & Y^{11} & := & X^1 \circ Y^{10} & = & \top & \\
 & & & Y^{12} & := & X^1 \circ Y^{11} & = & \top & = \; Y^{11} \\
X^2 & := & Y^{12} & = & \top & = & X^1 & &
\end{array}
$$

Hence, $\Phi \equiv \mathtt{tt}$. Now suppose that Player $\forall$ was given the opportunity to choose a subformula of a sequential composition. In this case he could enforce a play which traverses solely through the μ-quantified variable Y only. Hence, for such games we would have to abolish the correspondence between infinite unfoldings of fixed points and wins for either of the players known from parity games. $\dashv$

This example only explains why the games do a left-depth-first traversal through formulas w.r.t. sequential compositions. This does not mean though that parity winning conditions on these games provide a correct characterization of FLC's model checking problem. The next example shows that parity winning conditions indeed do not suffice.

Example 4.6. Consider the two-state LTS $\mathcal{T} = (\{s,t\}, \{\xrightarrow{a}, \xrightarrow{b}\}, L)$ with $L(s) = L(t) = \varnothing$, and $s \xrightarrow{a} t$, $t \xrightarrow{b} t$. We will evaluate the formula $\mu Y.\langle b\rangle \vee$

$\langle a\rangle; \nu X.Y; X$ on $\mathcal{T}$. Its precise semantics can be computed using fixed-point iteration again.

$$
\begin{aligned}
Y^0 &:= \bot \\
X^{00} &:= \top \\
X^{01} &:= Y^0 \circ X^{00} = \bot \circ \top = \bot \\
X^{02} &:= Y^0 \circ X^{01} = \bot \circ \bot = \bot = X^{01} \\
Y^1 &:= [\![\langle b\rangle]\!]^{\mathcal{T}} \sqcup ([\![\langle a\rangle]\!]^{\mathcal{T}} \circ X^{02}) = \lambda T.[\![\langle b\rangle]\!]^{\mathcal{T}}(T) \cup [\![\langle a\rangle]\!]^{\mathcal{T}}(\varnothing) \\
&= [\![\langle b\rangle]\!]^{\mathcal{T}}(T) \\
X^{10} &:= \top \\
X^{11} &:= Y^1 \circ X^{10} = [\![\langle b\rangle]\!]^{\mathcal{T}} \circ \top = \lambda T.[\![\langle b\rangle]\!]^{\mathcal{T}}(\{s,t\}) \\
&= \lambda T.\{t\} \\
X^{12} &:= Y^1 \circ X^{11} = [\![\langle b\rangle]\!]^{\mathcal{T}} \circ \lambda T.\{t\} = \lambda T.[\![\langle b\rangle]\!]^{\mathcal{T}}(\{t\}) \\
&= \lambda T.\{t\} = X^{11} \\
Y^2 &:= [\![\langle b\rangle]\!]^{\mathcal{T}} \sqcup ([\![\langle a\rangle]\!]^{\mathcal{T}} \circ X^{12}) = \lambda T.[\![\langle b\rangle]\!]^{\mathcal{T}}(T) \cup [\![\langle a\rangle]\!]^{\mathcal{T}}(\{t\}) \\
&= \lambda T.[\![\langle b\rangle]\!]^{\mathcal{T}}(T) \cup \{s\}
\end{aligned}
$$

Even though the fixed point is not found yet, we can deduce – by monotonicity – that $\mathcal{T}, s \models \Phi$ holds. Note that we will have $\lambda T.[\![\langle b\rangle]\!]^{\mathcal{T}}(T) \cup \{s\} \sqsubseteq [\![\Phi]\!]^{\mathcal{T}}$ and therefore $s \in [\![\Phi]\!]^{\mathcal{T}}(\{s,t\})$.

On the other hand, consider the following infinite play of $\mathcal{G}_{\mathcal{T}}(s,\Phi)$ which Player $\exists$ can enforce. It is also not hard to see that all other plays he can enforce should lead to a win for Player $\forall$ immediately because they end in a configuration in which $\exists$ gets stuck with no transitions to chose.

$$
\begin{array}{llrcl}
s & \vdash & \mu Y.\langle b\rangle \vee \langle a\rangle; \nu X.Y; X & ; & \mathtt{tt} \\
s & \vdash & Y & ; & \mathtt{tt} \\
s & \vdash & \langle b\rangle \vee \langle a\rangle; \nu X.Y; X & ; & \mathtt{tt} \\
s & \vdash & \langle a\rangle; \nu X.Y; X & ; & \mathtt{tt} \\
s & \vdash & \langle a\rangle & ; & (\nu X.Y; X); \mathtt{tt} \\
t & \vdash & \nu X.Y; X & ; & \mathtt{tt} \\
t & \vdash & X & ; & \mathtt{tt} \\
t & \vdash & Y; X & ; & \mathtt{tt} \\
t & \vdash & Y & ; & X; \mathtt{tt} \\
t & \vdash & \langle b\rangle \vee \langle a\rangle; \nu X.Y; X & ; & X; \mathtt{tt} \\
t & \vdash & \langle b\rangle & ; & X; \mathtt{tt} \\
t & \vdash & X & ; & \mathtt{tt} \\
 & & \vdots & &
\end{array}
$$

This play reaches a loop and can therefore be played ad infinitum. Note that both variables X and Y occur as principle formulas in configurations on this loop. Hence, if these games are equipped with an ordinary parity condition on principle formulas then the μ-quantified variable Y determines – as the outer one of the two – Player $\forall$ to be the winner of this play. But then he would have a winning strategy, and the games would not be correct.

The crucial difference between the occurrences of Y and X is that each Y does not stem from the unfolding of the Y above but from the unfolding of the inner X. Such a phenomenon does not occur in the parity model checking games for the modal μ-calculus. $\dashv$

The question that arises is how to recognize those variables that truly regenerate themselves when a simple comparison according to the outer-relation is not possible. The answer is provided by the stacks in those configuration that have variables as principle formulas. Note that between each two occurrences of X the stack does not decrease, but between two occurrences of Y it does. This shows that Y got "fulfilled" and the play continued with a formula that was on the stack when Y was principle – intuitively the left-depth-first search has terminated on this branch and follows a branch to the right of Y. This takes us back to the notion of $\mathit{Steps}(\pi)$ for a play π in a stair parity game.

Take the play π above. Then $\mathit{Steps}(\pi)$ consists of all positions whose stack contents persist for the rest of the game. Here they are $\{0, 1, 2, 3, 5, 6, 7, 11, \ldots\}$.

Definition 4.7. If $\pi = C_0, C_1, \ldots$ is an infinite play of $\mathcal{G}_{\mathcal{T}}(s_0, \Phi)$ and $C_i = s_i \vdash \varphi_i \ ; \ \delta_i$ then $\mathit{Steps}(\pi) = \{i \in \mathbb{N} : \forall j \geq i \ |\delta_j| \geq |\delta_i|\}$. Furthermore, let $\pi|_{st}$ denote the restriction of π to $\mathit{Steps}(\pi)$, i.e.

$$\pi|_{st} := C_{i_0}, C_{i_1}, C_{i_2}, \ldots \quad \text{iff} \quad \mathit{Steps}(\pi) = \{i_0, i_1, i_2, \ldots\}$$

with $i_j < i_{j'}$ iff $j < j'$.

This allows us to define the winning conditions of the FLC games in a way that correctly characterizes its model checking problem.

Definition 4.8. Let $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} : a \in \mathcal{A}\}, L)$ be an LTS, $s_0 \in \mathcal{S}$, Φ a closed FLC formula and $\pi = C_0, C_1, \ldots$ be a play of $\mathcal{G}_{\mathcal{T}}(s_0, \Phi)$ with $C_i = s_i \vdash \varphi_i \ ; \ \delta_i$ for all $i \in \mathbb{N}$. Player $\exists$ wins π if, and only if,

1. π is finite and ends in some C_n with
   a) $C_n = s_n \vdash q \ ; \ \delta$ and $q \in L(s)$,
   b) $C_n = s_n \vdash \neg q \ ; \ \delta$ and $q \notin L(s)$,
   c) $C_n = s_n \vdash [a] \ ; \ \delta$ and there is no $t \in \mathcal{S}$ s.t. $s \xrightarrow{a} t$;

2. π is infinite and the outermost variable occurring infinitely often as a principle formula in $\pi|_{st}$ is of type ν.

Player $\forall$ wins π if, and only if,

1. π is finite and ends in some C_n with
   a) $C_n = s_n \vdash q\ ;\ \delta$ and $q \notin L(s)$,
   b) $C_n = s_n \vdash \neg q\ ;\ \delta$ and $q \in L(s)$,
   c) $C_n = s_n \vdash \langle a \rangle\ ;\ \delta$ and there is no $t \in \mathcal{S}$ s.t. $s \xrightarrow{a} t$;
2. π is infinite and the outermost variable occurring infinitely often as a principle formula in $\pi|_{st}$ is of type μ.

It is then possible to show that each play has a unique winner, that the games are determined, etc.

Theorem 4.9 ([11])**.** Player $\exists$ has a winning strategy for the game $\mathcal{G}_{\mathcal{T}}(s, \Phi)$ if, and only if, $\mathcal{T}, s \models \Phi$.

As a consequence we obtain an upper bound on the complexity of FLC's model checking problem.

Corollary 4.10. The model checking problem for FLC can be decided in EXPTIME.

Proof. Let $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} : a \in \mathcal{A}\}, L)$ be an LTS and $\varphi_0 \in$ FLC. They induce a VPS $\mathcal{B}_{\mathcal{T},\varphi_0} = (Q, \mathcal{A}', \Gamma, \delta)$ with

- $Q = \mathcal{S} \times \mathit{Sub}(\varphi_0)$,
- $\mathcal{A}'_c = \{\mathit{chop}\}$, $\mathcal{A}'_r = \{\mathit{tau}, \mathit{mod}\}$, $\mathcal{A}'_i = \{\mathit{disj}, \mathit{conj}, \mathit{unf}\}$,
- $\Gamma = \mathit{Sub}(\varphi_0)$,
- δ simply translates the rules of Fig. 1 into a transition relation:

$$
\begin{aligned}
\delta_c &:= \{\big((s, \varphi;\psi), \mathit{chop}, (s,\varphi), \psi\big) : s \in \mathcal{S}, \varphi;\psi \in \mathit{Sub}(\varphi_0)\} \\
\delta_r &:= \{\big((s,\tau), \mathit{tau}, \varphi, (s,\varphi)\big) : s \in \mathcal{S}, \varphi \in \mathit{Sub}(\varphi_0)\} \\
&\quad \cup \{\big((s,\psi), \mathit{mod}, \varphi, (t,\varphi)\big) : \psi \in \{\langle a\rangle, [a]\}, s \xrightarrow{a} t, \varphi \in \mathit{Sub}(\varphi_0)\} \\
\delta_i &:= \{\big((s, \varphi_1 \vee \varphi_2), \mathit{disj}, (s,\varphi_i)\big) : s \in \mathcal{S}, \varphi_1 \vee \varphi_2 \in \mathit{Sub}(\varphi_0), i \in \{1,2\}\} \\
&\quad \cup \{\big((s, \varphi_1 \wedge \varphi_2), \mathit{conj}, (s,\varphi_i)\big) : s \in \mathcal{S}, \varphi_1 \wedge \varphi_2 \in \mathit{Sub}(\varphi_0), i \in \{1,2\}\} \\
&\quad \cup \{\big((s, \sigma X.\varphi), \mathit{unf}, (s,X)\big) : s \in \mathcal{S}, \sigma X.\varphi \in \mathit{Sub}(\varphi_0)\} \\
&\quad \cup \{\big((s,X), \mathit{unf}, (s,\varphi)\big) : s \in \mathcal{S}, X \in \mathit{Sub}(\varphi_0), \mathit{fp}X = \sigma X.\varphi\}
\end{aligned}
$$

A stair parity game is then obtained by simply making states of the form $(s, \varphi_0 \vee \varphi_1)$ choices of Player $\exists$ etc., and by assigning priorities to nodes $\big((s,\varphi),\delta\big)$ only depending on the principal formula φ s.t. all formulas other than variables have priority 0, μ-bound, resp. ν-bound variables have odd, resp. even priorities, and outer variables have greater priorities than inner ones. Correctness of this translation is given by the fact that the winning conditions of the FLC model checking games can easily be transferred into stair parity conditions by artificially prolonging finite plays ad infinitum. The complexity bound then follows from Thm. 4.4. Q.E.D.

These games do not only provide a local model checking algorithm for FLC. They can also be used to show that the fixed-point alternation hierarchy in FLC is strict [11]. The proof proceeds along the same lines as Arnold's proof for the alternation hierarchy in the modal μ-calculus [2] by constructing hard formulas (that define the winning positions for Player $\exists$ in such a game) and by using Banach's fixed-point theorem.

4.3 Model-Checking Games for the Modal Iteration Calculus

Stair Parity Games provide an elegant framework of model checking games for logics such as CaRet and FLC. We give further evidence for the significance of this concept in relation to fixed-point logics beyond the modal μ-calculus by showing that model checking games for MIC can also be phrased in this context. However, the games we present here only work for finite transition systems. The reason for this will become clear later in the section.

To simplify notation, we will only explain the games for MIC-formulas without simultaneous inductions. Using similar ideas one can extend the games to cover simultaneous fixed points also.

Suppose first that we are given a transition system $\mathcal{T}$ and a formula $\varphi := \mathbf{ifp}X.\psi$, where $\psi \in \mathrm{ML}$ is a modal logic formula in negation normal form. If Player $\exists$ wants to show that φ holds true at a node s in $\mathcal{T}$, he has to prove that there is a stage $n \in \mathbb{N}$ so that $s \in X^n$. Here, choosing n out of the natural numbers is enough as the fixed point in a finite transition system is always reached after a finite number of steps. In other words, he chooses an $n \in \mathbb{N}$ and then has to show that the n-fold unraveling[4] ψ^n of ψ holds true at s.

This idea is modeled in a stair parity game as follows. To choose the stage $n \in \mathbb{N}$, we give Player $\exists$ the option to push as many (finitely many) symbols X onto the stack as he wishes. This done, the two players continue by playing the standard modal logic game on the ML-formula ψ, with the modification that each time the game reaches the proposition X, one symbol

[4] The n-fold unravel-ling ψ^n of ψ is defined as follows: $\psi^0 := \mathtt{ff}$ and ψ^{n+1} is obtained from ψ by replacing each occurrence of X by ψ^n. It is easily seen that $s \in [\![\psi^n]\!]^{\mathcal{T}}$ if, and only if, the state s occurs in stage X^n of the induction on ψ in $\mathcal{T}$.

X is popped from the stack and the game continues at ψ again. If the stack is empty, then Player $\exists$ has lost as he has failed to show that the starting state s satisfies ψ^n. However, there is one problem we need to solve. As φ is a MIC formula, the fixed-point variable X may be used negatively, i.e. the play on ψ may reach a literal $\neg X$. In this case, we again pop one symbol X from the stack, but then the game proceeds to the negation $\neg\psi$. To keep track of whether we are currently playing in ψ or the negation thereof, we rename the fixed-point variable X in $\neg\psi$ to X^c. If the play reaches X^c and there are no more symbols X left on the stack, then Player $\exists$ wins. Otherwise, one symbol is popped and the play continues at $\neg\psi$ again. If, however, a literal $\neg X^c$ is reached, then one symbol X is popped and the game proceeds back to the original formula ψ.

It should be clear now that Player $\exists$ can win this game on a formula $\varphi := \mathbf{ifp} X.\psi$ and a transition system $\mathcal{T}$ with initial state s if, and only if, $s \in [\![\varphi]\!]^{\mathcal{T}}$.

To extend this idea to formulas containing nested fixed points, we have to modify the game slightly. Suppose a variable Y is bound by a fixed-point operator $\mathbf{dfp}\, Y.\psi$ inside the formula $\mathbf{ifp}\, X.\vartheta$ which binds an outer fixed-point variable X. When the game reaches $\mathbf{ifp}\, X.\vartheta$, Player $\exists$ pushes as many symbols X onto the stack as he likes. The game continues inside ϑ and reaches the formula $\mathbf{dfp}\, Y.\psi$ at which Player $\forall$ can push symbols Y onto the stack. Now, when the game reaches an atom X, then before we can regenerate the formula ϑ and pop one symbol X from the stack, we have to pop all Ys first. Other than that, the rules of the game are as described above.

To present this idea in more detail, let us first fix some notation. Let $\varphi \in$ MIC be a formula in negation normal form and let $X_1, \ldots, X_k$ be the fixed-point variables occurring in it. W.l.o.g. we assume that no fixed-point variable is bound twice in φ. Hence, with each X_i we can associate the unique formula ϑ_i such that X_i is bound in φ by $\mathbf{fp}\ X_i.\vartheta_i$, where $\mathbf{fp}$ is either $\mathbf{ifp}$ or $\mathbf{dfp}$. We also assume that the X_i are numbered such that if $i < j$ then ϑ_i is not a subformula of ϑ_j.

Let φ' be the formula obtained from $\neg\varphi$ by first renaming every fixed-point variable X_i in φ to X_i^c and then bringing the formula into negation normal form. Let $\Phi := \mathit{Sub}(\varphi) \cup \mathit{Sub}(\varphi')$.

Let $\mathcal{T} := (\mathcal{S}, \{\xrightarrow{a} : a \in \mathcal{A}\}, L)$ be a finite transition system. The formula φ and the system $\mathcal{T}$ induce a visibly pushdown system $\mathcal{B}_{\mathcal{T},\varphi} := (Q, \mathcal{A}', \Gamma, \delta)$ as follows. The stack alphabet is $\Gamma := \{X_1, \ldots, X_k\}$.

For each variable X_i or X_i^c we use a gadget $\mathit{clear}(X_i)$ that pops all symbols X_j from the stack with $j > i$ until the top of the stack contains a symbol X_j with $j \leq i$. As the gadget is deterministic, we can arbitrarily assign the nodes in it to either player. To simplify the presentation, we will

treat these gadgets as black boxes, i.e. as single nodes in the game graph.

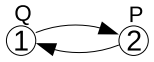
Now, Q contains all pairs $\mathcal{S} \times \Phi$ and the nodes of the gadgets $\mathit{clear}(X_i)$ and $\mathit{clear}(X_i^c)$ for $1 \leq i \leq k$. (Recall that $\Phi := \mathit{Sub}(\varphi) \cup \mathit{Sub}(\varphi')$.)

We let $\mathcal{A}' := \mathcal{A}_c \cup \mathcal{A}_r \cup \mathcal{A}_i$ where $\mathcal{A}_c := \{\mathit{push}\}$, $\mathcal{A}_r := \{\mathit{pop}\}$, and $\mathcal{A}_i := \{\mathit{int}\}$.

$$
\begin{aligned}
\delta_c &:= \big\{\big((s, \mathbf{ifp}X.\vartheta), \mathit{push}, (s, \mathbf{ifp}X.\vartheta), X\big) : s \in \mathcal{S}, \mathbf{ifp}X.\vartheta \in \Phi\big\} \\
&\quad \cup \big\{\big((s, \mathbf{dfp}X.\vartheta), \mathit{push}, (s, \mathbf{ifp}X.\vartheta), X\big) : s \in \mathcal{S}, \mathbf{dfp}X.\vartheta \in \Phi\big\} \\
\delta_r &:= \big\{\big((s, \mathit{clear}(X_i)), \mathit{pop}, X_i, (s, \vartheta_i)\big) : 1 \leq i \leq k, s \in \mathcal{S}\big\} \\
&\quad \cup \big\{\big((s, \mathit{clear}(X_i^c)), \mathit{pop}, X_i, (s, \vartheta_i^c)\big) : 1 \leq i \leq k, s \in \mathcal{S}\big\} \\
\delta_i &:= \big\{\big((s, \mathbf{ifp}X.\vartheta), \mathit{int}, (s, \vartheta)\big) : s \in \mathcal{S}, \mathbf{ifp}X.\vartheta \in \Phi\big\} \\
&\quad \cup \big\{\big((s, \mathbf{dfp}X.\vartheta), \mathit{int}, (s, \vartheta)\big) : s \in \mathcal{S}, \mathbf{dfp}X.\vartheta \in \Phi\big\} \\
&\quad \cup \big\{\big((s, \varphi_1 \vee \varphi_2), \mathit{int}, (s, \varphi_i)\big) : s \in \mathcal{S}, \varphi_1 \vee \varphi_2 \in \Phi, i \in \{1,2\}\big\} \\
&\quad \cup \big\{\big((s, \varphi_1 \wedge \varphi_2), \mathit{int}, (s, \varphi_i)\big) : s \in \mathcal{S}, \varphi_1 \vee \varphi_2 \in \Phi, i \in \{1,2\}\big\} \\
&\quad \cup \big\{\big((s, \langle a\rangle\psi), \mathit{int}, (t, \psi)\big) : s \in \mathcal{S}, t \in \mathcal{S},\ s \xrightarrow{a} t, \langle a\rangle\psi \in \Phi\big\} \\
&\quad \cup \big\{\big((s, [a]\psi), \mathit{int}, (t, \psi)\big) : s \in \mathcal{S}, t \in \mathcal{S},\ s \xrightarrow{a} t, [a]\psi \in \Phi\big\} \\
&\quad \cup \big\{\big((s, X_i), \mathit{int}, \mathit{clear}(X_i)\big) : s \in \mathcal{S}, 1 \leq i \leq k\big\} \\
&\quad \cup \big\{\big((s, X_i^c), \mathit{int}, \mathit{clear}(X_i^c)\big) : s \in \mathcal{S}, 1 \leq i \leq k\big\} \\
&\quad \cup \big\{\big((s, \neg X_i), \mathit{int}, \mathit{clear}(X_i^c)\big) : s \in \mathcal{S}, 1 \leq i \leq k\big\} \\
&\quad \cup \big\{\big((s, \neg X_i^c), \mathit{int}, \mathit{clear}(X_i)\big) : s \in \mathcal{S}, 1 \leq i \leq k\big\}
\end{aligned}
$$

To turn $\mathcal{B}_{\mathcal{T},\varphi}$ into a visibly pushdown game, we need to assign priorities and the nodes where each of the players moves. Player $\exists$ moves at disjunctions, nodes $(s, \langle a\rangle\psi)$, $(s, \mathbf{ifp}X_i.\vartheta_i)$, (s, X_i), $(s, \neg X_i^c)$, $(s, \mathit{clear}(X_i))$, and nodes (s, q), $q \in \mathcal{P}$, if $q \notin L(s)$. At all other nodes Player $\forall$ moves. Finally, nodes $(s, \mathbf{ifp}X_i.\vartheta_i)$ are assigned the priority 1 and nodes $(s, \mathbf{dfp}X_i.\vartheta_i)$ are assigned the priority 0. Note that the priority assignment only needs to ensure that no player can loop forever on a fixed-point formula $\mathbf{fp}X_i.\vartheta_i$ pushing infinitely many variables onto the stack. As there are no infinite plays unless one of the players keeps pushing symbols onto the stack forever, the priorities do not influence the winner of "interesting" plays.

Example 4.11. We illustrate the construction by an example. Let $\mathcal{T}$ be the system



and let $\varphi := \mathbf{ifp}\ X.\big(p \vee \mathbf{ifp}\ Y.(q \wedge \Diamond X)\big)$. The corresponding game graph is depicted in Fig. 2, where $\varphi_2 := (p \vee \mathbf{ifp}\ Y.(q \wedge \Diamond X))$, $\varphi_3 := \mathbf{ifp}\ Y.(q \wedge \Diamond X)$, and $\varphi_4 := q \wedge \Diamond X$ are the non-trivial sub-formulas of φ. To simplify the

presentation, we have dropped the labels *int* and annotated the *push* and *pop* labels by the variable being pushed onto the stack. Note that there are no *pop* Y or *clear*(Y) labels, as the variable Y does not occur as an atom in the formula.

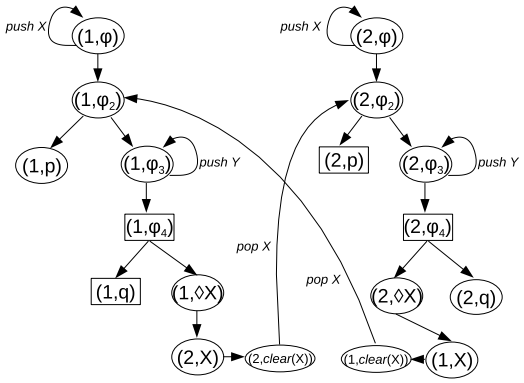


FIGURE 2. Visibly Pushdown System for Ex. 4.11.

Clearly, Player $\exists$ wins the game from position $(1, \varphi)$ by first using the push transition to push one variable X onto the stack and then continue to $(1, \varphi_2)$. In this way, the play will either terminate in $(1, q)$ or continue along the node $(2, X)$ to $(2, \mathit{clear}(X))$ and then along the *pop*-edge, where the symbol X will be popped from the stack, to $(2, \varphi)$, and finally to $(2, p)$. In both cases Player $\forall$ loses. Note, however, that Player $\exists$ cannot win without initially pushing X onto the stack, as the play will then terminate at the node $(2, \mathit{clear}(X))$ with the *pop*-edge no longer available. This corresponds to the state 1 being in the stage X^2 but not in the stage X^1. (By pushing X once onto the stack, Player $\exists$ enforces the play to go through the inner formula φ_2 twice, corresponding to the stage X^2.) $\dashv$

The following theorem can be proved along the same lines as the corresponding proof for backtracking games in [5].

Theorem 4.12. For every $\varphi \in$ MIC (without simultaneous fixed points) and finite transition system $\mathcal{T}$, Player $\exists$ has a winning strategy from a node (s, φ) in the visibly pushdown game $\mathcal{B}_{\mathcal{T},\varphi}$ if, and only if, $\mathcal{T}, s \models \varphi$.

Clearly, winning regions of general visibly pushdown games are not definable in MIC (as computing them is EXPTIME-hard), presumably not even if we restrict attention to a fixed number of priorities. However, the pushdown games constructed above have a rather simple structure. They only have two priorities but, even more important, the push transitions are

local, i.e. for each fixed-point operator in φ there is one node which has a self-loop pushing a variable onto the stack. Therefore, there is hope that we can identify a suitable fragment of visibly pushdown games containing the games arising from MIC-formulas and whose winning regions can be defined in MIC, i.e. the winner of games from this fragment are MIC-definable in the same way as the winner of parity games can be defined in $\mathcal{L}_\mu$.

We illustrate this by considering games arising from formulas $\mathbf{ifp}X.\psi$ where $\psi \in \mathrm{ML}$. Such a game has a starting node from which a path labeled by *push* emerges. To each node v on this path with distance n to the root there is a copy of the game ψ^n attached to it, where ψ^n is the n-fold unraveling of ψ w.r.t. X. Hence, to define the winner of such games we only need a formula that checks whether on this *push*-path emerging from the root there is a node such that Player $\exists$ wins the modal logic game attached to it. The latter is clearly MIC-definable, so the whole formula is easily seen to be definable in MIC.

It is conceivable that a similar construction using nested fixed points works for games arising from MIC-formulas with nested fixed points. However, a formal proof of this results is beyond the scope of this survey and is left for future work.

4.4 Backtracking Games

We now turn to a different type of non-regular games, the so-called *backtracking games*. The motivation for backtracking games comes from properties such as *the tree is balanced* as shown to be expressible in MIC and FLC. To verify such properties in a game-theoretical setting, the game needs to be able to inspect all subtrees rooted at successors of the root of a finite tree. However, *linear* games such as parity games that construct an infinite path through a game arena can only visit one subtree, unless we introduce back-edges towards the root. This motivates a game model where a play is no longer an infinite path but a more complex subgraph. Backtracking games were originally introduced as model checking games for inflationary fixed-point logics such as MIC and the general inflationary fixed-point logic IFP (see [5]). We first give an informal description.

Backtracking games are essentially parity games with the addition that, under certain conditions, players can jump back to an earlier position in the play. This kind of move is called backtracking.

A backtracking move from position v to an earlier position u is only possible if v belongs to a given set B of backtrack positions, if u and v have the same priority $\Omega(v)$ and if no position of higher priority has occurred between u and v. With such a move, the player who backtracks not only resets the play back to u, he also commits himself to a backtracking distance d, which is the number of positions of priority $\Omega(v)$ that have been seen between u and v. After this move, the play ends when d further positions of

priority $\Omega(v)$ have been seen, unless this priority is "released" by a higher priority.

For finite plays we have the winning condition that a player wins if his opponent cannot move. For infinite plays, the winner is determined according to the parity condition, i.e. Player $\exists$ wins a play π if the highest priority seen infinitely often in π is even, otherwise Player $\forall$ wins.

Definition 4.13. The arena $\mathcal{G} := (V, E, V_\exists, V_\forall, B, \Omega)$ of a backtracking game is a directed graph (V, E), with a partition $V = V_\exists \cup V_\forall$ into positions of Player $\exists$ and positions of Player $\forall$, a subset $B \subseteq V$ of backtrack positions and a map $\Omega : V \to \{0, \ldots, k-1\}$ that assigns to each node a priority.

Proposition 4.14 ([5])**.** The following basic properties about backtracking games are known.

1. Backtracking games are determined, i.e. in every backtracking game, one of the two players has a winning strategy
2. Backtracking games in general do not admit finite memory strategies.
3. Deciding the winner of a backtracking game even with only two priorities is hard for NP and co-NP.
4. Deciding the winner of a backtracking game in general is PSPACE-hard.

However, as yet no upper bound for the complexity of backtracking games is known.

Backtracking games can be used as model checking games for inflationary fixed-point logics, e.g. for every MIC-formula φ and every transition system $\mathcal{T}$ one can construct in polynomial time a backtracking game that is won by Player $\exists$ if, and only if, $\mathcal{T} \models \varphi$. Here, the backtracking distance plays the role of the stack being used to determine a stage of the fixed-point induction containing the current state of the transition system. The rule that higher priorities reset the distance for all lower priorities corresponds to the usual idea that regenerating an outer fixed point restarts the induction on the inner fixed points.

Unlike the stair parity games we constructed in Sec. 4.3, it seems unlikely that the winner of backtracking games is definable in MIC, even for the very simple fragment of backtracking games that suffice for a game-theoretical framework for MIC model checking. The reason is that while in a push-down game, the possible stack contents are represented in the game graph explicitly, the backtracking distance is an "external" concept and counting the distance must be done in the logic itself. Therefore it seems unlikely that MIC suffices for this. In [5], it was shown, however, that the winner of a restricted class of backtracking games can be defined in inflationary fixed-point logic.

5 Outlook

Clearly, MIC and FLC are not the only (modal) fixed-point logics that extend the modal μ-calculus semantically. Another modal fixed-point logic of high expressivity is *Higher-Order Fixed-Point Logic* (HFL) [20]. It incorporates into $\mathcal{L}_\mu$ a simply typed λ-calculus. Its ground type is that of a *predicate* and its only type constructor is the function arrow. Syntactically, HFL extends $\mathcal{L}_\mu$ by function abstraction ($\lambda X.\varphi$) and application ($\varphi\ \psi$).

Not surprisingly, HFL subsumes FLC. In fact, every (sub-)formula in FLC is, semantically, a predicate transformer, i.e. an object of a function type. This way, FLC is embedded into a very low level of HFL, namely HFL1 – the *First-Order Fragment of HFL*. Here, first order does not refer to predicate logic but to the degree of function typing that is allowed in subformulas of that logic. HFL0, the fragment of HFL restricted to formulas in which every subformula is a predicate, is exactly $\mathcal{L}_\mu$– syntactically already.

The type level hierarchy in HFL is strict, and it comes with increasing model checking complexity: it is k-EXPTIME-complete for level k of that hierarchy [3], and this holds already for the data complexity of each fragment. Consequently, model checking full HFL is non-elementary.

HFL, and in particular HFL1, is also interesting as a specification language for non-regular properties. It can, for instance, define assume-guarantee properties [20]. Furthermore, it can define structural properties that we are unable to express in FLC or MIC like that of being bisimilar to a (possibly infinite) word model. Even though we have not formally defined HFL1, we can present the formula for this property because it is very neat, and it can be read with a little bit of understanding of functional programming.

$$\neg\Big(\ \big(\ \mu X^{\mathtt{Pr}\to\mathtt{Pr}\to\mathtt{Pr}}.\lambda A^{\mathtt{Pr}}.\lambda B^{\mathtt{Pr}}.(A\wedge B)\vee(X\ \Diamond A\ \Diamond B)\ \big)\ \langle a\rangle\mathtt{tt}\ \langle b\rangle\mathtt{tt}\ \Big)$$

The superscripts are type annotations. The least fixed-point formula can be seen as a recursively defined function that takes two predicates and checks whether or not their conjunction holds. If not, it calls itself recursively with the two arguments preceeded by $\Diamond$-operators. Applied to the two initial arguments, it checks successively, whether there are two paths of length $1, 2, \ldots$ ending in an a-, resp. b-transition.

We have not included a thorough presentation of HFL (or just HFL1) here, mainly because there is no interesting known game-theoretic characterization of its model checking problem. It can be solved by a reduction to a reachability game using fixed-point elimination [3], but it is not known whether or not there is an extension of parity games to capture this. The example above suggests that stair parity games do not suffice since two stacks would be needed to contain the $\Diamond$-operators for the two different paths.

We conclude with a positive remark: while FLC is trivially embeddable into HFL1, and MIC and FLC seem incomparable, it is reasonable to ask whether HFL1 is a superlogic of both of them. On finite models, MIC can indeed be embedded into HFL. This is because the *computation* of an inflationary fixed point can be carried out by a function of first-order type. However, since this is modeled iteratively, this translation fails in stages beyond ω. Hence, it may not work on infinite models. It remains to be seen whether MIC can be embedded into HFL over arbitrary models.

References

[1] R. Alur, K. Etessami, and P. Madhusudan. A temporal logic of nested calls and returns. In *Proc. 10th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems, TACAS'04*, volume 2988 of *LNCS*, pages 467–481. Springer, 2004.

[2] A. Arnold. The modal μ-calculus alternation hierarchy is strict on binary trees. *RAIRO - Theoretical Informatics and Applications*, 33:329–339, 1999.

[3] R. Axelsson, M. Lange, and R. Somla. The complexity of model checking higher-order fixpoint logic. *Logical Methods in Computer Science*, 3(2:7):1–33, 2007.

[4] A. Dawar, E. Grädel, and S. Kreutzer. Inflationary fixed points in modal logics. *ACM Transactions on Computational Logic*, 5(2):282–315, 2004.

[5] A. Dawar, E. Grädel, and S. Kreutzer. Backtracking games and inflationary fixed points. *Theoretical Computer Science*, 350(2-3), 2006. ICALP 2004 selected paper issue.

[6] S. Dziembowski, M. Jurdziński, and D. Niwiński. On the expression complexity of the modal μ-calculus model checking. Unpublished manuscript, 1996.

[7] E. A. Emerson. Uniform inevitability is tree automaton ineffable. *Information Processing Letters*, 24(2):77–79, 1987.

[8] O. H. Ibarra, T. Jiang, and H. Wang. A characterization of exponential-time languages by alternating context-free grammars. *TCS*, 99(2):301–315, 1992.

[9] D. Janin and I. Walukiewicz. On the expressive completeness of the propositional μ-calculus with respect to monadic second order logic. In

Proc. 7th Conf. on Concurrency Theory, CONCUR'96, volume 1119 of *LNCS*, pages 263–277. Springer, 1996.

[10] M. Lange. Alternating context-free languages and linear time μ-calculus with sequential composition. In *Proc. 9th Workshop on Expressiveness in Concurrency, EXPRESS'02*, volume 68.2 of *ENTCS*, pages 71–87. Elsevier, 2002.

[11] M. Lange. The alternation hierarchy in fixpoint logic with chop is strict too. *Information and Computation*, 204(9):1346–1367, 2006.

[12] M. Lange. Three notes on the complexity of model checking fixpoint logic with chop. *R.A.I.R.O. – Theoretical Informatics and Applications*, 2006. (To appear).

[13] M. Lange and R. Somla. Propositional dynamic logic of context-free programs and fixpoint logic with chop. *Information Processing Letters*, 100(2):72–75, 2006.

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

[15] Ch. Löding, P. Madhusudan, and O. Serre. Visibly pushdown games. In *Proc. 24th Int. Conf. on Foundations of Software Technology and Theoretical Computer Science, FSTTCS'04*, volume 3328 of *LNCS*, pages 408–420. Springer, 2004.

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

[17] A. Okhotin. Boolean grammars. *Information and Computation*, 194(1):19–48, 2004.

[18] C. Stirling. Local model checking games. In *Proc. 6th Conf. on Concurrency Theory, CONCUR'95*, volume 962 of *LNCS*, pages 1–11. Springer, 1995.

[19] A. Tarski. A lattice-theoretical fixpoint theorem and its application. *Pacific Journal of Mathematics*, 5:285–309, 1955.

[20] M. Viswanathan and R. Viswanathan. A higher order modal fixed point logic. In *Proc. 15th Int. Conf. on Concurrency Theory, CONCUR'04*, volume 3170 of *LNCS*, pages 512–528. Springer, 2004.

[21] I. Walukiewicz. Pushdown processes: Games and model-checking. *Information and Computation*, 164(2):234–263, 2001.