A Modal Fixpoint Logic with Chop

Markus Müller-Olm

Department of Computer Science
University of Dortmund
44221 Dortmund, Germany
mmo@ls5.informatik.uni-dortmund.de

Abstract. We study a logic called FLC (Fixpoint Logic with Chop) that extends the modal mu-calculus by a chop-operator and termination formulae. For this purpose formulae are interpreted by predicate transformers instead of predicates. We show that any context-free process can be characterized by an FLC-formula up to bisimulation or simulation. Moreover, we establish the following results: FLC is strictly more expressive than the modal mu-calculus; it is decidable for finite-state processes but undecidable for context-free processes; satisfiability and validity are undecidable; FLC does not have the finite-model property.

1 Introduction

Imperative programming languages typically offer a sequential composition operator which allows the straightforward specification of behavior proceeding in successive phases. Similar operators are provided by interval temporal logics, where they are called *chop*-operators. Important examples are Moszkowski's Interval Temporal Logic ITL [13] and the Duration Calculus DC [17]. As far as we know, however, no point-based temporal logic and, in particular, no branching-time logic with a chop operator has been proposed up to now. Indeed, at first glance there seems to be no natural way for explaining the meaning of sequentially composed formulae $\phi_1 ; \phi_2$ in the setting of point-based temporal or modal logic, as there is no natural notion of where interpretation of ϕ_1 stops and interpretation of ϕ_2 starts.

In this paper we present a logic called FLC (Fixpoint Logic with Chop) that extends the modal mu-calculus [8], a popular point-based branching-time fixpoint logic, by a chop operator ; and *termination formulae* term. For this purpose we utilize a 'second-order' interpretation of formulae. While (closed) formulae of usual temporal logics are interpreted by sets of states, i.e. represent *predicates*, we interpret formulae by mappings from states to states, i.e. by *predicate transformers*. A similar idea has been used by Burkart and Steffen [3] in a model checking procedure for modal mu-calculus formulae and context-free processes. However, while we use a second-order interpretation of formulae, they rely on a second-order interpretation of states as property transformers.

It turns out that FLC is strictly more expressive than the modal mu-calculus but is still decidable for finite-state processes. Consequently, FLC-based model

© Springer-Verlag Berlin Heidelberg 1999

checking is, in our opinion, an interesting alternative to modal mu-calculus based model checking as it enables to verify non-regular properties. The chop-operator also enables a straightforward specification of phased behavior. Other results shown in this paper are that FLC is undecidable for context-free processes, that satisfiability (and thus also validity) is undecidable, and that the logic does not have the finite model property. These results are inferred from the existence of formulae characterizing context-free processes up to bisimulation and simulation.

The remainder of this paper is structured as follows. The next section recalls bisimulation, simulation and context-free processes. In Section 3 we introduce the logic FLC and show that it conservatively extends the modal mu-calculus. Section 4 shows that context-free processes can be characterized up to bisimulation and simulation by single FLC-formulae. These facts, besides being of interest in their own, provide the main means for establishing the results on expressiveness and decidability, which are presented in Section 5. The paper finishes with a discussion of the practical utility of FLC.

2 Preliminaries

Processes, Bisimulation, and Simulation. A commonly used basic operational model of processes is that of rooted labeled transition systems. Assume for the remainder of this paper given a finite set Act of actions. Then a *labeled transition system* (over Act) is a structure $T = (S, Act, \rightarrow)$, where S is a set of *states*, and $\rightarrow \subseteq S \times Act \times S$ is a *transition relation.* We write $s \xrightarrow{a} s'$ for $(s, a, s') \in \rightarrow$. A *process* is a pair $P = (T, s_0)$ consisting of a labeled transition system $T = (S, Act, \rightarrow)$ and an *initial state* (or *root*) $s_0 \in S$. A process is called *finite-state* if the underlying state set S is finite.

Transition systems provide a rather fine-grained model of processes. Therefore, various equivalences and preorders have been studied in the literature that identify or order processes on the basis of their behavior. Classic examples are strong bisimulation [15, 11] denoted by $\sim$ and simulation denoted by $\preceq$.

For two given processes $P = ((S, Act, \rightarrow_P), s_0)$ and $Q = ((T, Act, \rightarrow_Q), t_0)$ both bisimulation $\sim$ and simulation $\preceq$ are first defined as relations between the state sets S and T. These definitions are then lifted to the processes themselves. As relations between S and T they can be characterized as the greatest fixpoints $\nu F_\sim$ and $\nu F_\preceq$ of certain monotonic functionals $F_\sim$ and $F_\preceq$. These functionals operate on the complete lattice of relations $R \subseteq S \times T$ ordered by set inclusion and are defined by

$$\begin{aligned} F_\sim(R) \overset{\text{def}}{=} \{(s,t) \mid \; & \forall a, s' : s \xrightarrow{a}_P s' \Rightarrow \exists t' : t \xrightarrow{a}_Q t' \wedge (s', t') \in R \\ & \wedge \forall a, t' : t \xrightarrow{a}_Q t' \Rightarrow \exists s' : s \xrightarrow{a}_P s' \wedge (s', t') \in R\} \end{aligned}$$

and $F_\preceq(R) \overset{\text{def}}{=} \{(s,t) \mid \forall a, s' : s \xrightarrow{a}_P s' \Rightarrow \exists t' : t \xrightarrow{a}_Q t' \wedge (s', t') \in R\}$. The processes P and Q are called *bisimilar* if $s_0 \sim t_0$. Similarly, Q is said to *simulate* P if $s_0 \preceq t_0$. By abuse of notation we denote these relationships by $P \sim Q$ and $P \preceq Q$ and view $\sim$ and $\preceq$ also as relations between processes.

$$\begin{tikzcd}
{} \arrow[r] & A \arrow[r, "a"] \arrow[d, "c"] & AB \arrow[r, "a"] \arrow[d, "c"] & AB^2 \arrow[r, "a"] \arrow[d, "c"] & AB^3 \arrow[r, "a"] \arrow[d, "c"] & \cdots \\
& \varepsilon & B \arrow[l, "b"'] & B^2 \arrow[l, "b"'] & B^3 \arrow[l, "b"'] & \cdots \arrow[l, "b"']
\end{tikzcd}$$

Fig. 1. A context-free process

Context-Free Processes. Context-free processes, also called BPA (basic process algebra) processes [1], are a certain type of finitely generated infinite-state processes. Their name derives from the fact that they are induced by leftmost derivations of context-free grammars in Greibach normal form, where the terminal symbols are interpreted as actions and the non-terminals induce the state. Greibach normal form means that all rules have the form $A ::= a\alpha$, where A is a non-terminal symbol, a a terminal symbol, and α a string of non-terminals. Formally, context-free processes can be defined as an instance of Rewrite Transition Systems as introduced by Caucal [4].

A *context-free process rewrite system* (over Act) is a triple $R = (V, Act, \Delta)$ consisting of a finite set V of *process variables*, the assumed finite set Act of *actions*, and a finite set $\Delta \subseteq V \times Act \times V^*$ of *rules*. The labeled transition system induced by $R = (V, Act, \Delta)$, called a *context-free transition system*, is $T_R = (V^*, Act, \rightarrow)$, where $\rightarrow \subseteq V^* \times A \times V^*$ is the smallest relation obeying the prefix rewrite rule

$$\text{PRE} \quad \frac{(A, a, \alpha) \in \Delta}{A\beta \xrightarrow{a} \alpha\beta} \quad .$$

Note that the states in a context-free transition system are words of process variables of the underlying context-free process rewrite system. A *context-free process* is a pair (T_R, α_0), consisting of a context-free transition system T_R and an initial state $\alpha_0 \in V^*$. As an example we picture in Fig. 1 the context-free process (T_R, A) where $R = (V, Act, \Delta)$, $V = \{A, B\}$, $Act = \{a, b, c\}$, and $\Delta = \{(A, a, AB), (A, c, \varepsilon), (B, b, \varepsilon)\}$.

The following two results are crucial for the remainder of this paper: firstly, there are context-free processes that are not bisimilar to any finite-state process (the process in Fig. 1 is an example) and, secondly, simulation between context-free processes is undecidable [6]. The reader interested in learning more about context-free processes and other classes of Rewrite Transition Systems is pointed to the surveys [12] and [2] and the many references there.

3 The Logic FLC

In the remainder of this paper the letter X ranges over an infinite set Var of *variables*, a over the assumed finite action set Act and p over an assumed finite set $Prop$ of *atomic propositions*. We assume that $Prop$ contains the propositions true and false.

The modal mu-calculus [8] is a small, yet expressive process logic that has been used as underlying logic in a number of model checkers. Modal mu-calculus formulae in positive normal form are constructed according to the grammar

$$\phi ::= p \mid [a]\phi \mid \langle a\rangle\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid X \mid \mu X\,.\,\phi \mid \nu X\,.\,\phi\ \ .$$

In the modal mu-calculus the modal operators $[a]$ and $\langle a\rangle$ do not have the status of formulae but can only be used in combination with already constructed formulae ϕ to form composed formulae $[a]\phi$ and $\langle a\rangle\phi$. We now define FLC (*Fixpoint Logic with Chop*), an extension of the modal mu-calculus that gives the modal operators the status of formulae. More importantly, FLC provides a *chop operator* ;, which intuitively represents sequential composition of behavior, and a *termination formula* term, which intuitively requires the behavior of the sequential successor formula.

We consider again formulae in positive form which are now constructed according to the following grammar:

$$\phi ::= p \mid [a] \mid \langle a\rangle \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid X \mid \mu X\,.\,\phi \mid \nu X\,.\,\phi \mid \mathsf{term} \mid \phi_1\ ;\ \phi_2\ \ .$$

As in the modal mu-calculus, the two *fixpoint operators* μX and νX bind the respective variable X and we will apply the usual terminology of free and bound variables in a formula, closed formula etc. Moreover, we write for a finite set M of formulae $\bigwedge M$ and $\bigvee M$ for the conjunction and disjunction of the formulae in M. As usual, we agree that $\bigwedge\emptyset = \mathsf{true}$ and $\bigvee\emptyset = \mathsf{false}$.

Both the modal mu-calculus as well as FLC are basically interpreted over a given labeled transition system $T = (S, \mathit{Act}, \rightarrow)$. Furthermore, an *interpretation* $I \in (\mathit{Prop} \rightarrow 2^S)$ is assumed, which assigns to each atomic proposition the set of states for which it is valid. We assume that interpretations always interpret true and false in the standard way, i.e. such that $I(\mathsf{true}) = S$ and $I(\mathsf{false}) = \emptyset$.

In the modal mu-calculus the meaning of a closed formula essentially is a subset of the state set S, i.e. a *predicate* on states. In order to explain the meaning of the new types of formulae, we interpret FLC-formulae by monotonic predicate transformers. A *(monotonic) predicate transformer* is simply a mapping $f : 2^S \rightarrow 2^S$ which is monotonic w.r.t. the inclusion ordering on 2^S. It follows from well-known results of lattice theory that the set of monotonic predicate transformers, which we denote by MTrans_T, together with the pointwise extension $\sqsubseteq$ of the inclusion ordering on 2^S defined by

$$f \sqsubseteq f' \ \text{ iff } \ f(x) \subseteq f'(x) \text{ for all } x \subseteq S$$

is a complete lattice. We denote the join and meet operations by $\sqcup$ and $\sqcap$.

It is customary to refer to *environments*, in order to explain the meaning of open formulas. In the modal mu-calculus, environments are partial mappings of type $\rho : \mathit{Var} \overset{\text{part.}}{\rightarrow} 2^S$; they interpret (at least) the free variables of the formula in question by a set of states. In FLC we interpret free variables by predicate transformers. Thus, we use environments of type $\delta : \mathit{Var} \overset{\text{part.}}{\rightarrow} \mathit{MTrans}_T$. The predicate transformer assigned to an FLC-formula ϕ, denoted by $\mathcal{C}_T^I(\phi)(\delta)$, is

$$\begin{aligned}
\mathcal{C}_T^I(p)(\delta)(x) &= I(p)\\
\mathcal{C}_T^I([a])(\delta)(x) &= \{s \mid \forall s' : s \xrightarrow{a} s' \;\Rightarrow\; s' \in x\}\\
\mathcal{C}_T^I(\langle a\rangle)(\delta)(x) &= \{s \mid \exists s' : s \xrightarrow{a} s' \;\wedge\; s' \in x\}\\
\mathcal{C}_T^I(\phi_1 \wedge \phi_2)(\delta)(x) &= \mathcal{C}_T^I(\phi_1)(\delta)(x) \cap \mathcal{C}_T^I(\phi_2)(\delta)(x)\\
\mathcal{C}_T^I(\phi_1 \vee \phi_2)(\delta)(x) &= \mathcal{C}_T^I(\phi_1)(\delta)(x) \cup \mathcal{C}_T^I(\phi_2)(\delta)(x)\\
\mathcal{C}_T^I(X)(\delta) &= \delta(X)\\
\mathcal{C}_T^I(\mu X \,.\, \phi)(\delta) &= \sqcap\{f \in \mathit{MTrans}_T \mid \mathcal{C}_T^I(\phi)(\delta[X \mapsto f]) \sqsubseteq f\}\\
\mathcal{C}_T^I(\nu X \,.\, \phi)(\delta) &= \sqcup\{f \in \mathit{MTrans}_T \mid \mathcal{C}_T^I(\phi)(\delta[X \mapsto f]) \sqsupseteq f\}\\
\mathcal{C}_T^I(\mathsf{term})(\delta)(x) &= x\\
\mathcal{C}_T^I(\phi_1 \,;\, \phi_2)(\delta) &= \mathcal{C}_T^I(\phi_1)(\delta) \circ \mathcal{C}_T^I(\phi_2)(\delta)
\end{aligned}$$

Fig. 2. Semantics of FLC

inductively defined in Fig. 2. The similar definition of the predicate $\mathcal{M}_T^I(\phi)(\rho)$ assigned to a modal mu-calculus formulae ϕ is omitted due to lack of space. It can be found in many papers on the modal mu-calculus.

Note that the fixpoint formulae of FLC are interpreted by the corresponding fixpoints in the set of predicate transformers and not in the set of predicates as in the modal mu-calculus. Also note that the chop operator is interpreted by functional composition and that term denotes the identity predicate transformer. Thus, term is the neutral element of ;.

As the meaning of a closed formula ϕ does not depend on the environment, we sometimes write just $\mathcal{C}_T^I(\phi)$ $(\mathcal{M}_T^I(\phi))$ for $\mathcal{C}_T^I(\phi)(\delta)$ $(\mathcal{M}_T^I(\phi)(\rho))$, where δ (ρ) is an arbitrary environment. We also omit the indices T and I if they are clear from the context.

The set of states *satisfying* a given closed formula ϕ is $\mathcal{C}(\phi)(S)$. A process $P = (T, s_0)$ is said to satisfy ϕ if its initial state s_0 satisfies ϕ. It might appear somewhat arbitrary that the predicate transformer $\mathcal{C}(\phi) : 2^S \to 2^S$ is applied to the full state set S in the definition of satisfaction. As far as expressiveness is concerned, however, the choice of a specific set x to which $\mathcal{C}(\phi)$ is applied is largely arbitrary, as long as x can be described by a closed FLC formula ϕ_x: assume $x = \mathcal{C}(\phi_x)(S)$; then $\mathcal{C}(\phi)(x)$ equals $\mathcal{C}(\phi \,;\, \phi_x)(S)$. As Lemma 1 below shows, sets x expressible in this way include at least all state sets that can be described by a modal mu-calculus formula (i.e. all modal mu-calculus definable properties). The formula $\phi_{DL} \overset{\text{def}}{=} \bigwedge_{a \in Act}[a] \,;\, \mathsf{false}$, for instance, characterizes the set of deadlocked states.

Any modal mu-calculus formula ϕ can straightforwardly be translated to FLC: just replace all sub-formulas of the form $[a]\psi$ or $\langle a\rangle\psi$ by $[a] \,;\, \psi$ or $\langle a\rangle \,;\, \psi$, respectively. We call the resulting FLC-formula $\mathcal{T}(\phi)$. A rather straightforward structural induction shows that the interpretation of $\mathcal{T}(\phi)$ is just the constant predicate transformer mapping any state set to the interpretation of the original modal mu-calculus formula.

Lemma 1. *Let ϕ be a modal mu-calculus formula and $\rho : Var \overset{\text{part.}}{\to} 2^S$ a modal mu-calculus environment. Let δ be the environment defined by $\mathsf{dom}\,\delta = \mathsf{dom}\,\rho$ and $\delta(X) = \lambda y\,.\,\rho(X)$ for $X \in \mathsf{dom}\,\rho$. (Note that δ assigns constant predicate transformers to the variables.) Then $\mathcal{C}(\mathcal{T}(\phi))(\delta) = \lambda y\,.\,M(\phi)(\rho)$.*

As a consequence, FLC is at least as expressive as the modal mu-calculus.

Corollary 1. *Suppose ϕ is a closed modal mu-calculus formula and P is a process. Then P satisfies ϕ (in the sense of the mu-calculus) iff P satisfies $\mathcal{T}(\phi)$ (in the sense of FLC).*

Equation systems. A (closed) *equation systems* of FLC-formula is a set $E = \{X_i = \phi_i \mid 1 \leq i \leq n\}$ consisting of $n \geq 0$ equations $X_i = \phi_i$, where $X_1, \ldots, X_n$ are mutually distinct variables and $\phi_1, \ldots, \phi_n$ are FLC-formulae having at most $X_1, \ldots, X_n$ as free variables. An environment $\delta : \{X_1, \ldots, X_n\} \to MTrans$ is a *solution* of equation system E, if $\delta(X_i) = \mathcal{C}(\phi_i)(\delta)$ for $i = 1, \ldots, n$. By the Knaster-Tarski fixpoint theorem every equation system has a largest solution as the corresponding functional on environments is easily seen to be monotonic. We denote the largest solution of E by νE.

While it proves convenient to refer to equation systems, they do not increase the expressive power. Any predicate transformer that can be obtained as a component of the largest solution of an equation system E can just as well be characterized by a single formula. In order to show this, Gauß elimination [10] can be applied to the equation system (see e.g. [14]).

Proposition 1. *Let E be a closed equation system and X a variable bound in E. Then there is a closed FLC-formula ϕ such that $\mathcal{C}(\phi) = (\nu E)(X)$.*

4 Characteristic Formulae for Context-Free Processes

The goal of this section is to show that any context-free process can be characterized up to bisimulation or simulation by an FLC-formula.[1] As a stepping stone, we construct equation systems that capture the contribution of the single process variables to bisimulation and simulation. Characteristic formulae for various other (bi-)simulation-like relations, in particular the weak versions, can be constructed along this line too.

In the following, we assume given a context-free process rewrite system $R = (V, Act, \Delta)$ and agree on the following variable conventions: the letters A and $B, B_1, B_2, \ldots$ range over V, a ranges over Act, and α and β range over V^*. For notational convenience we use the process variables $A \in V$ also as variables of the logic.

We consider the three equation systems $E_\sim = \{A = \phi_{\sim A} \mid A \in V\}$, $E_\preceq = \{A = \phi_{\preceq A} \mid A \in V\}$, and $E_\succeq = \{A = \phi_{\succeq A} \mid A \in V\}$. Analogously to the

[1] For the simulation case we shall actually construct two formulae. One of them characterizes the set of processes that are simulated by the process in question and the other the set of processes that simulate it.

finite-state case [16, 14], the formulae $\phi_{\sim A}$, $\phi_{\preceq A}$, and $\phi_{\succeq A}$ mirror the conditions in the definition of bisimulation and simulation and are defined by

$$\begin{aligned}
\phi_{\sim A} &\overset{\text{def}}{=} \phi_{\succeq A} \wedge \phi_{\preceq A} \ , \\
\phi_{\preceq A} &\overset{\text{def}}{=} \bigwedge_{a \in Act} [a] \ ; \bigvee_{(A,a,B_1 \cdots B_l) \in \Delta} B_1 \ ; \ldots ; B_l \ , \text{ and} \\
\phi_{\succeq A} &\overset{\text{def}}{=} \bigwedge_{a \in Act} \ \bigwedge_{(A,a,B_1 \cdots B_l) \in \Delta} \langle a \rangle \ ; B_1 \ ; \ldots ; B_l \ .
\end{aligned}$$

Now, suppose given an arbitrary transition system $T = (S, Act, \rightarrow)$ and an arbitrary interpretation $I : Prop \rightarrow 2^S$. (The specific interpretation does not matter as only the atomic propositions true and false appear in the characteristic equation systems.) Let $\delta_\sim = \nu E_\sim : V \rightarrow (2^S \rightarrow 2^S)$ be the largest solution of $E_\sim$ on T. The following lemma intuitively shows that the A-component of this solution represents the contribution of the process variable A to bisimulation.

Lemma 2. *$\delta_\sim(A)(\{s \in S \mid s \sim \beta\}) = \{s \in S \mid s \sim A\beta\}$ for all $A \in V$, $\beta \in V^*$.*

The '$\supseteq$'-direction can be proved by a fixpoint induction for $\delta_\sim$ and the '$\subseteq$'-direction by a fixpoint induction for $\sim = \nu F_\sim$. Combined with Proposition 1, Lemma 2 shows that there is a closed formula $\varphi_{\sim A}$ for each $A \in V$ such that for all $\beta \in V^*$:

$$\mathcal{C}_T(\varphi_{\sim A})(\{s \in S \mid s \sim \beta\}) = \{s \in S \mid s \sim A\beta\} \ . \tag{1}$$

These formulae $\varphi_{\sim A}$ can now be used to construct characteristic formulae for context-free processes with underlying process rewrite system R.

Theorem 1 (Characteristic formulae). *For each context-free process P there is a (closed) FLC-formula $\psi_{\sim P}$ such that, for any process Q, Q satisfies $\psi_{\sim P}$ iff $Q \sim P$.*

Proof. Let $P = (T_R, B_1 \cdots B_l)$ and let $\psi_{\sim P}$ be the formula $\varphi_{\sim B_1} \ ; \ldots ; \varphi_{\sim B_l} \ ; \phi_{DL}$, where ϕ_{DL} is the formula characterizing the set of deadlocked states from Section 3.

Suppose $Q = ((S, Act, \rightarrow), s_0)$ is an arbitrary process. Clearly, a state $s \in S$ is bisimilar to the state ε in T_R if and only if it satisfies ϕ_{DL}. It follows by repeated application of (1) that, for $i = 1, \ldots, l$, a state $s \in S$ satisfies $\varphi_{\sim B_i} \ ; \ldots ; \varphi_{\sim B_l} \ ; \phi_{DL}$ if and only if $s \sim B_i \cdots B_l$. Thus, Q is bisimilar to P if and only if it satisfies $\psi_{\sim P}$. $\square$

An analogue of Lemma 2 for $E_\preceq$ and $E_\succeq$ ensures the existence of closed formulae $\varphi_{\preceq A}$ and $\varphi_{\succeq A}$ such that $\mathcal{C}_T(\varphi_{\preceq A})(\{s \in S \mid s \preceq \beta\}) = \{s \in S \mid s \preceq A\beta\}$ and $\mathcal{C}_T(\varphi_{\sim A})(\{s \in S \mid s \succeq \beta\}) = \{s \in S \mid s \succeq A\beta\}$ for arbitrary A and β. These formulae are used to establish the final theorem of this section.

Theorem 2 (Characteristic formulae for simulation). *For each context-free process P there are (closed) FLC-formulae $\psi_{\preceq P}$ and $\psi_{\succeq P}$ such that, for any process Q, Q satisfies $\psi_{\preceq P}$ iff $Q \preceq P$, and Q satisfies $\psi_{\succeq P}$ iff $Q \succeq P$.*

Proof. Let $P = (T_R, B_1 \cdots B_l)$ and $Q = ((S, Act, \rightarrow), s_0)$.

A state $s \in S$ is simulated by ε if and only if it satisfies ϕ_{DL}. Thus, $\psi_{\preceq P}$ can be chosen as the formulae $\varphi_{\preceq B_1}$; ... ; $\varphi_{\preceq B_l}$; ϕ_{DL}.

On the other hand, every state $s \in S$ simulates ε. In other words, a state simulates ε if and only if it satisfies the formulae true. Thus, $\psi_{\succeq P}$ can be chosen as the formulae $\varphi_{\succeq B_1}$; ... ; $\varphi_{\succeq B_l}$; true.[2] □

5 Decidability and Expressiveness Issues

Clearly, FLC is decidable for finite-state processes: given a finite-state process $P = (T, s_0)$, an interpretation I, and an FLC-formula ϕ, $\mathcal{C}_T^I(\phi)$ can effectively be computed inductively over ϕ. The usual approximation of fixpoints terminates as $MTrans_T$ is finite.

Theorem 3. *FLC is decidable for finite-state processes.*

However, FLC is not decidable for context-free processes. This is a consequence of the existence of characteristic formulae for simulation. A decision procedure for FLC could namely be used to decide simulation between context-free processes, which is – as mentioned in Section 2 – undecidable: given two context-free processes P and Q one would just have to check, whether Q satisfies $\psi_{\succeq P}$ in order to decide, whether $P \preceq Q$.

Theorem 4. *FLC is undecidable for context-free processes.*

There is an interesting duality between the decidability of FLC for finite-state processes and the decidability of the modal mu-calculus for context-free (and even push-down) processes [3]. Both scenarios relate an inherently 'regular' structure with a structure of at least 'context-free strength'. While the former is concerned with the at least 'context-free' logic FLC and 'regular' finite-state processes, the latter relates the 'regular' modal mu-calculus (recall that the mu-calculus can be translated to monadic second order logic, which closely corresponds to finite automata) with context-free processes.

The existence of characteristic formulae for simulation also implies that satisfiability (and hence validity) of FLC is undecidable: assume given two context-free processes P and Q. It is easy to see that Q simulates P if and only if the formula $\psi_{\succeq P} \wedge \psi_{\preceq Q}$ is satisfiable. Thus, decidability of satisfiability would again imply decidability of simulation between context-free processes.

Theorem 5. *Satisfiability and validity of FLC are undecidable.*

An interesting consequence of the existence of characteristic formulae for bisimulation is that FLC does not enjoy the finite-model property:[3] choose a

[2] The final true could be omitted due to our definition of satisfaction.

[3] A modal logic has the finite-model property, if any satisfiable formula has a finite model.

$$\begin{tikzcd} 0 \arrow[r, "a"] & 1 \arrow[r, "b"] & 2 \arrow[r, "a"] & 3 \arrow[r, "b"] & 4 \arrow[r, "b"] & 5 \arrow[r, "c"] & 6 \end{tikzcd}$$

Fig. 3. A linear, finite process

context-free process P that is not bisimilar to any finite-state process and take its characteristic formulae $\psi_{\sim P}$. Then this formulae is satisfiable (namely by P itself). But it cannot be satisfied by a finite-state process, as this finite-state process would then be bisimilar to the context-free process which yields a contradiction.

Theorem 6. *FLC does not enjoy the finite-model property.*

The modal mu-calculus on the other hand does enjoy the finite-model property [9]. Hence, in general, context-free processes cannot have characteristic modal mu-calculus formulae: let P again be a context-free process that is not bisimilar to a finite-state process and assume there would be a modal mu-calculus formula ϕ characterizing P up to bisimulation. Then – by the finite-model property – there would be a finite-state process Q satisfying ϕ. But this would mean that P and Q are bisimilar, which contradicts the choice of P.

As a consequence, FLC is strictly more expressive than the modal mu-calculus. However, if this increase of expressiveness would not show through on finite-state processes, it would be useless, as far as automatic model checking is concerned.

Fortunately, we can show that FLC already is more expressive on the class of finite-state processes, and even on a small subclass, that of *finite linear processes*. These are processes corresponding to finite words over Act. Formally, the process corresponding to a word $w = w_0 \cdots w_k \in Act^*$ is $P_w = ((\{0, \ldots, k\}, Act, \rightarrow), 0)$, where $\rightarrow = \{(i, w_i, i+1) \mid 0 \leq i < k\}$. As an example, the process corresponding to the word *ababbc* is pictured in Fig. 3. The class of finite linear processes is $\{P_w \mid w \in Act^*\}$ and subclasses of it can straightforwardly be identified with sets of words over Act. The modal mu-calculus can be translated to monadic second-order logic. Therefore, the class of finite linear models of a modal mu-calculus formula ϕ corresponds to a regular set of words. The class of finite linear models of the FLC-formula $(\mu X \,.\, (\mathsf{term} \vee \langle a \rangle X \langle b \rangle))\ ;\ \phi_{DL}$, however, corresponds to the set $\{a^n b^n \mid n = 0, 1, \ldots\}$, which is well-known not to be regular [7].

Theorem 7. *FLC is strictly more expressive than the modal mu-calculus, even on finite linear processes, and therefore also on finite-state processes.*

It is interesting to note that FLC can even characterize certain non-context-free sets of finite linear processes due to the presence of conjunction: let, for arbitrary actions a and b, ϕ_{ab} be the formula $\mu X \,.\, (\mathsf{term} \vee (\langle a \rangle\ ;\ X\ ;\ \langle b \rangle))$ and ψ_a be the formula $\mu X \,.\, (\mathsf{term} \vee \langle a \rangle\ ;\ X)$. Then the finite linear models of

$$(\phi_{ab}\ ;\ \phi_c\ ;\ \phi_{DL}) \wedge (\phi_a\ ;\ \phi_{bc}\ ;\ \phi_{DL})$$

correspond to the set $\{a^n b^n c^n \mid n = 0, 1, \ldots\}$, which is context-sensitive but not context-free. A more thorough study of the expressiveness of FLC is left for future research.

6 Conclusion

We have proposed a modal logic FLC with fixpoints, a chop-operator, and termination formulae. The basic idea has been to interpret formulae by predicate transformers instead of predicates and to take fixpoint construction over predicate transformers as well. As a stepping stone in the technical development we have shown that FLC allows to characterize context-free processes up to bisimulation and simulation. FLC is strictly more expressive than the modal mu-calculus but is still decidable for finite-state processes.

Like the modal mu-calculus, FLC is perhaps not so much suited as a direct vehicle for specification. Rather it provides an expressive core logic, into which various other logics can be translated. An FLC-based model checking system could handle non-regular specification formalisms that are beyond the reach of modal mu-calculus based model checkers. An interesting example of such a formalism from a practical point of view are the timing diagrams studied by K. Fisler in [5]. It is a topic of future research whether they can actually be embedded into FLC.

A simple global model checking algorithm for FLC and finite-state processes can straightforwardly be constructed from the usual iterative computation of fixpoints. This procedure in general has an exponentially larger storage requirement compared to a straightforward global modal mu-calculus model checker: we have to store a mapping $2^S \to \mathbb{B}$ per state and formula (where $\mathbb{B}$ denotes the set of the Boolean values true and false) instead of just a single Boolean value.[4] Also the time complexity of FLC seems to be much higher than that of modal mu-calculus as $(2^S \to 2^S)$ has exponentially longer chains than 2^S such that fixpoint computation can require exponentially more iterations. Thus, at first glance model checking FLC seems to be impractical. (We currently know that model checking with a fixed formula is at least PSPACE-hard.)

However, the exponential blow up can be avoided for FLC-formulae corresponding to modal mu-calculus formulae. The idea is to represent the above mentioned mappings of type $2^S \to \mathbb{B}$ by binary decision diagrams (BDDs). As a consequence of Lemma 1 these mappings are constant for all FLC-formulae corresponding to modal mu-calculus formulae. It is, moreover, easy to see that the intermediate functions occurring during fixpoint iteration are constant too and correspond to the Boolean values that would be observed in a mu-calculus model checking procedure. Therefore, only a linear penalty arises for both space and time when model checking FLC-formula corresponding to modal mu-calculus formulae because constant BDDs can be represented in constant space. In this sense the increased expressiveness of FLC is obtained for free: the exponential

[4] A collection consisting of one of those mappings for each state in S represents a mapping $2^S \to 2^S$, i.e. a predicate transformer, which is the meaning of a formula.

blow-up can only occur in cases that cannot be handled by a modal mu-calculus model checker at all!

The above comparison applies to straight-forward global model checking. If and how more elaborate global and local mu-calculus model checking procedures can be adapted to FLC remains to be seen. Other topics for future research are a more thorough study of the complexity and expressiveness of FLC, in particular its relationship to context-free and context-sensitive languages and, last not least, the implementation and empirical evaluation of an FLC-based model checker. It is, moreover, interesting to study, whether the idea of a 'second-order' interpretation of formulae by predicate transformers can advantageously be applied to other logics.

References

1. J. A. Bergstra and J. W. Klop. Algebra of communicating processes with abstraction. *Theoretical Computer Science*, 37:77–121, 1985.
2. O. Burkart and J. Esparza. More infinite results. *ENTCS*, 6, 1997. URL: `http://www.elsevier.nl/locate/entcs/volume6.html`.
3. O. Burkart and B. Steffen. Model checking the full modal mu-calculus for infinite sequential processes. In *ICALP '97*, LNCS 1256, 419–429. Springer-Verlag, 1997.
4. D. Caucal. On the regular structure of prefix rewriting. *Theoretical Computer Science*, 106:61–86, 1992.
5. K. Fisler. Containment of regular languages in non-regular timing diagram languages is decidable. In *CAV'97*, LNCS 1254. Springer-Verlag, 1997.
6. J. F. Groote and H. Hüttel. Undecidable equivalences for basic process algebra. *Information and Computation*, 115(2):354–371, 1994.
7. J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, 1979.
8. D. Kozen. Results on the propositional mu-calculus. *Theoretical Computer Science*, 27:333–354, 1983.
9. D. Kozen. A finite model theorem for the propositional mu-calculus. *Studia Logica*, 47:233–241, 1988.
10. A. Mader. Modal mu-calculus, model checking and Gauss elimination. In *TACAS'95*, LNCS 1019, 72–88. Springer-Verlag, 1995.
11. R. Milner. *Communication and Concurrency.* Prentice Hall, 1989.
12. F. Moller. Infinite results. In *CONCUR'96*, LNCS 1119, 195–216. Springer-Verlag, 1996.
13. B. Moszkowski. A temporal logic for multi-level reasoning about hardware. *IEEE Computer*, 18(2):10–19, 1985.
14. M. Müller-Olm. Derivation of characteristic formulae. *ENTCS*, 18, 1998. URL: `http://www.elsevier.nl/locate/entcs/volume18.html`.
15. D. M. R. Park. Concurrency and automata on infinite sequences. In LNCS 154, 561–572. Springer-Verlag, 1981.
16. B. Steffen and A. Ingólfsdóttir. Characteristic formulae for processes with divergence. *Information and Computation*, 110(1):149–163, 1994.
17. Zhou Chaochen, C. A. R. Hoare, and A. P. Ravn. A calculus of durations. *Information Processing Letters*, 40(5):269–276, 1991.