Temporal Logics for Non-Regular Properties: Model Checking

Martin Lange

Institut für Informatik, Ludwig-Maximilians-Universität München, Germany
Martin.Lange@ifi.lmu.de

Abstract

The modal μ-calculus captures exactly the bisimulation-invariant regular tree languages. Hence, properties expressed in formulas of its fragments like LTL or CTL for instance are only regular. We present temporal logics whose expressive power reaches beyond that of the modal μ-calculus and survey known result about these with a focus on their model checking problems — due to the undecidability of satisfiability.

1 Introduction

Temporal logics or, more generally, modal logic with recursion mechanism are used in computer science for program verification [1,22]. Regardless of whether verification is done in terms of model or satisfiability checking, the focus has been on logics that are fragments of or can be embedded into Kozen's modal μ-calculus $\mathcal{L}_\mu$: LTL [19], CTL [5], CTL* [6], or even PDL [7], etc.

There are equivalence preserving translations forth and back between formulas of the modal μ-calculus and various types of finite tree automata or Monadic Second Order Logic [10,11,20]. Hence, all properties expressed in the aforementioned logics are regular in the sense that the set of tree or word models of these formulas form an ω-regular language.

In fact, the expressive power of these four logics for example is strictly below that of full ω-regularity: for instance LTL defines only the star-free properties on ω-words [8,21], and on infinite trees, CTL is weaker than CTL* which coincides with the bisimulation invariant fragment of Monadic Path Logic [17]. All of them, including PDL, can be embedded into the first level of the alternation hierarchy within the modal μ-calculus [2].

Nevertheless, much attention is being payed to these weak logics when it comes to automatic verification. This is (partially) explained by two observations: (1) Increasing expressive power naturally comes with increasing computational complexity and vice-versa. In automatic verification, where

input sizes are typically very large, weak logics may be the only choice. (2) Properties expressed in stronger logics of regular properties than those, i.e. in higher levels of $\mathcal{L}_\mu$'s alternation hierarchy are usually examples of the form: something must hold infinitely often unless something else holds infinitely unless something else ... (to be read right-associatively).

This does not mean that considering even larger classes of properties will result in even more pathological examples. In fact, there are properties that are inexpressible in $\mathcal{L}_\mu$ but which are interesting for the verification of transmission protocols for example: "never more `out`- than `in`-actions", unlimited counting, repetition of sequences of actions, etc.

In Section 2 we will briefly describe four temporal logics that are capable of expressing non-regular properties. As mentioned above, higher expressive power requires higher computational complexity. The model checking problems for these logics are, in this order, complete for PTIME, PSPACE, EXPTIME, and non-elementary. The price to pay w.r.t. satisfiability is even higher: all these logics are highly undecidable. Intuitively, in order to achieve non-regular effects one has to be able to model context-freeness. Furthermore, sensible specification logics feature a $\wedge$-construct. But the intersection problem for context-free languages is undecidable.

Undecidability also restricts the model checking problem to finite structures. Checking non-regular properties on finite structures is not an oxymoron. Note that on a structure of fixed size, mutli-modal logic even suffices to express every possible μ-calculus definable property. But the formula would depend on the structure and it is desirable to have properties formalised independently of the underlying structure to be tested. The same holds for non-regular properties. Such logics provide independence of the underlying models and in same cases even succinctness over logics for regular properties.

2 Temporal Logics for Non-Regular Properties

2.1 Transition Systems

Let $\mathcal{P} = \{p, q, \ldots\}$ be a finite set of atomic proposition, and let $\Sigma = \{a, b, \ldots\}$ be a finite set of atomic action names. A transition system is a tuple $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} \mid a \in \Sigma\}, s_0, L)$ where $\mathcal{S}$ is the set of *states*, $\xrightarrow{a}$ for any $a \in \Sigma$ is a binary relation on states called the *transitions*, $s_0 \in \mathcal{S}$ is a designated *starting state*, and $L : \mathcal{S} \to 2^{\mathcal{P}}$ labels the states with propositions.

The four logics that are presented in the following are all interpreted over transition systems.

2.2 Non-Regular PDL

Propositional Dynamic Logic, as introduced by Fisher and Ladner [7] building on a proposal by Pratt is multi-modal logic over an infinite set of transition relations that form a Kleene Algebra. It was originally introduced to describe

properties of *regular* programs, i.e. programs built from atomic ones using the regular operations *union*, *sequential composition* and *Kleene star*.

Harel, Pnueli and Stavi extended this to *Propositional Dynamic Logic of Non-Regular Programs* (PDL[CFG]) [9] by allowing programs to be built from atomic ones using the full power of context-free grammars.

Definition 2.1 Formulas of PDL[CFG] are given by the following grammar.

$$\varphi \ ::= \ q \mid \varphi \vee \varphi \mid \neg\varphi \mid \langle G \rangle \varphi$$

where $q \in \mathcal{P}$ and G is a context-free grammar over the alphabet Σ. As usual, we write $L(G)$ to denote the language generated by G.

The semantics of PDL[CFG] is defined as follows.

$$\begin{aligned} \mathcal{T}, s \models q &\text{ iff } q \in L(s) \\ \mathcal{T}, s \models \varphi \vee \psi &\text{ iff } \mathcal{T}, s \models \varphi \text{ or } \mathcal{T}, s \models \psi \\ \mathcal{T}, s \models \neg\varphi &\text{ iff } \mathcal{T}, s \not\models \varphi \\ \mathcal{T}, s \models \langle G \rangle \varphi &\text{ iff } \exists w \in L(G), \exists t \in \mathcal{S}, \text{ s.t. } s \xrightarrow{w} t \end{aligned}$$

where the transition relations are lifted to words in Σ^* in a natural way:

$$\begin{aligned} s \xrightarrow{\epsilon} t &\text{ iff } s = t \\ s \xrightarrow{aw} t &\text{ iff } \exists u, \text{ s.t. } s \xrightarrow{a} u \text{ and } u \xrightarrow{w} t \end{aligned}$$

Example 2.2 Consider the context-free grammar G over the alphabet $\Sigma = \{a, b\}$ given by the production rule

$$S \ \rightarrow \ b \mid aSS$$

Note that $L(G) = \{w \mid \forall u, v \in \Sigma^* : w = uv \Rightarrow |u|_a \leq |u|_b\}$. This is easily seen to be a non-regular language. Moreover, $L(G)$ describes the undesired runs of a buffer if, for example, $b = \texttt{out}$ and $a = \texttt{in}$. Hence, the PDL[CFG] formula $[G]\texttt{ff} := \neg\langle G \rangle(q \vee \neg q)$ specifies that on all paths in a transition system the number of b-transitions never exceeds the number of a-transitions.

Theorem 2.3 [12] *The model checking problem for PDL[CFG] is PTIME-complete.*

2.3 The Modal Iteration Calculus

Definition 2.4 Let $(V, \leq)$ be a complete lattice with bottom element $\bot$ and suprema $\sqcup$, and let $f : V \to V$ be any function an V, not necessarily monotone. For limit ordinals λ and arbitrary ordinals α define

$$f^0 := \bot \ , \quad f^{\alpha+1} := f^\alpha \sqcup f(f^\alpha) \ , \quad f^\lambda := \bigsqcup_{\alpha<\lambda} f^\alpha$$

Then the trans-finite sequence $f^0, f^1, \ldots, f^\omega, f^{\omega+1}, \ldots$ trivially forms in increasing chain of elements in V and eventually there is an α s.t. $f^\alpha = f^{\alpha+1}$. This is called the *inflationary fixpoint* of f, denoted $\mathtt{inf} f$, and it is easy to see that it is indeed a fixpoint of f: $f(\mathtt{inf} f) = \mathtt{inf} f$.

Inflationary fixpoints are as interesting from a computational point of view as least fixpoints are. They can serve as quantifiers whose definition already yields a method for computing them.

Inspired by the (perhaps surprising) result that *First-Order Logic with Least Fixpoints* LFP is equi-expressive to *First-Order Logic with Inflationary Fixpoints* IFP, Dawar, Grädel and Kreutzer defined the *Modal Iteration Calculus* MIC which extends the modal μ-calculus with inflationary fixpoint constructs [3].

Definition 2.5 Let $\mathcal{V} = \{X, Y, \ldots\}$ be a set of second-order variables. Formulas of MIC are given by the following grammar.

$$\varphi \;:=\; q \mid X \mid \varphi \vee \varphi \mid \neg\varphi \mid \langle a \rangle \varphi \mid \mathtt{inf}(X_1 \leftarrow \varphi_1, \ldots, X_n \leftarrow \varphi_n)$$

where $q \in \mathcal{P}$, $a \in \Sigma$, and $X, X_1, \ldots, X_n \in \mathcal{V}$. The semantics of a MIC formula is defined using environments ρ that interpret the free variables in a formula by a subset of $\mathcal{S}$, e.g.

$$\begin{aligned}
&[\![\langle a \rangle \varphi]\!]_\rho \;:=\; \{s \in \mathcal{S} \mid \exists t \in \mathcal{S}, s \xrightarrow{a} t \text{ and } t \in [\![\varphi]\!]_\rho\} \\
&[\![\mathtt{inf}(X_1 \leftarrow \varphi_1, \ldots, X_n \leftarrow \varphi_n)]\!]_\rho \;:= \\
&\qquad \pi_1\big(\mathtt{inf}(\lambda(S_1, \ldots, S_n).([\![\varphi_n]\!]_{\rho[S_1/X_1,\ldots,S_n/X_n]}, \ldots, [\![\varphi_n]\!]_{\rho[S_1/X_1,\ldots,S_n/X_n]})\big)
\end{aligned}$$

etc., where π_1 projects a tuple onto its first component.

A fragment of MIC worth mentioning is 1MIC. It is obtained by replacing the syntax rule for arbitrary inflationary fixpoint by a rule for non-simultaneous fixpoint quantifiers only: $\mathtt{inf} X \leftarrow \varphi$.

Example 2.6 Take the MIC formula

$$\mathtt{inf}(X \leftarrow q \vee (\langle - \rangle \mathtt{tt} \wedge [-](X \wedge \neg Y)), Y \leftarrow X \wedge \neg q)$$

This expresses *uniform inevitability*: the formula is satisfied in a state s of a transition system iff there is a $k \in \mathbb{N}$ s.t. on all paths beginning in s, q holds after k steps. No finite tree automaton recognises this language [4].

Theorem 2.7 [3] *The model checking problem for MIC is PSPACE-complete.*

2.4 Fixpoint Logic with Chop

Müller-Olm's *Fixpoint Logic with Chop* (FLC) extends the modal μ-calculus with a sequential composition operator [18]. This makes it possible to achieve context-free, and because of the presence of intersection, even certain context-sensitive effects.

Definition 2.8 Let $\mathcal{V} = \{X, Y, \ldots\}$ be a set of third-order variables for functions from sets of states to sets of states. Formulas of FLC are given by the following grammar.

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

where $q \in \mathcal{P}$, $a \in \Sigma$ and $X \in \mathcal{V}$. Note that the semantics of a modal μ-calculus formula is a predicate, a subset of the state space. In order to cope with sequential composition, the semantics of FLC lifts the modal μ-calculus semantics straight-forwardly to the space of monotone *predicate transformers*. For details we refer to [18]. A game-based characterisation of the FLC semantics has been given in [13].

Example 2.9 The following FLC formula expresses "all finite paths are of the form ww for some $w \in \Sigma^*$".

$$\varphi \;:=\; \bigwedge_{(a,b)\in\Sigma^2, a\neq b} \psi_a; \psi_b; \langle - \rangle; \mathtt{tt} \qquad \text{where} \quad \psi_x \;:=\; \nu X.[x] \wedge [-]; X; [-]$$

where $\mathtt{tt} := q \vee \neg q$ for some q. This is based on the usual trick which shows that the complement of the non-context-free language $\{ww \mid w \in \Sigma^*\}$ is context-free.

Theorem 2.10 [16,14] *The model checking problem for FLC is EXPTIME-complete.*

2.5 Higher-Order Fixpoint Logic

The idea of obtaining more expressiveness by employing higher-order constructs has been followed consequently by Viswanathan and Viswanathan in the introduction of *Higher Order Fixpoint Logic* (HFL) [23]. It is a hybrid of the modal μ-calculus and a simply typed λ-calculus.

Types of HFL formulas are given by the grammar

$$\tau \;::=\; \mathrm{Pr} \mid \tau \to \tau$$

where the atomic type Pr stands for predicates, and function types are understood canonically. Variables in HFL formulas occur either only positively, or only negatively, or without any constraint. Thus, a *variance* is a $\sigma \in \{+, -, ?\}$. The syntax of HFL is then given by the following grammar.

$$\varphi \;::=\; q \mid \varphi \vee \varphi \mid \neg\varphi \mid X \mid \langle a \rangle \varphi \mid \varphi\ \varphi \mid \lambda X^{\sigma}.\varphi \mid \mu X.\varphi$$

Typing rules ensure the well-formedness of formulas. For a detailed definition of HFL including its straight-forward typing rules see the introductory paper [23]. We also refer to this for the exact definition of the semantics which is given through liftings of predicates to monotone higher order functions on

some set of states of a transition system. Monotonicity is guaranteed by the typing rules: fixpoint variables may only occur positively in their defining fixpoint body.

Example 2.11 An example of a class of properties that can be formalised in HFL has been given by Viswanathan and Viswanathan []: *assume-guarantee properties*. A state satisfies the formula $\nu X.\varphi \triangleright \nu Y.\psi$ iff for all $\alpha \in \mathbb{O}rd$: if s satisfies $\nu^{\alpha} X.\varphi$ then s also satisfies $\nu^{\alpha+1} Y.\psi$. In other words, if we assume φ to hold α many times then we must guarantee ψ at least $\alpha + 1$ many times. This property is expressed by the HFL formula

$$(\nu Z.\lambda X.\lambda Y.(\neg X \vee Y) \wedge Z\ \varphi\ \psi)\ \mathtt{tt}\ \psi$$

where the greatest fixpoint quantifier ν is abbreviated as usual through least fixpoints and negation.

Theorem 2.12 [15] *The model checking problem for HFL is non-elementary but elementary for every HFL^k.*

References

[1] Clarke, E. M., E. A. Emerson and A. P. Sistla, *Automatic verification of finite state concurrent systems using temporal logic specifications*, in: *Proc. 10th Symp. on Principles of Programming Languages, POPL'83* (1983), pp. 117–126.

[2] Dam, M., *CTL* and ECTL* as fragments of the modal μ-calculus*, TCS **126** (1994), pp. 77–96.

[3] Dawar, A., E. Grädel and S. Kreutzer, *Inflationary fixed points in modal logic*, in: L. Fribourg, editor, *Proc. 15th Workshop on Computer Science Logic, CSL'01*, LNCS (2001), pp. 277–291.

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

[5] Emerson, E. A. and J. Y. Halpern, *Decision procedures and expressiveness in the temporal logic of branching time*, Journal of Computer and System Sciences **30** (1985), pp. 1–24.

[6] Emerson, E. A. and J. Y. Halpern, *"Sometimes" and "not never" revisited: On branching versus linear time temporal logic*, Journal of the ACM **33** (1986), pp. 151–178.

[7] Fischer, M. J. and R. E. Ladner, *Propositional dynamic logic of regular programs*, Journal of Computer and System Sciences **18** (1979), pp. 194–211.

[8] Gabbay, D., A. Pnueli, S. Shelah and J. Stavi, *The temporal analysis of fairness*, in: *Proc. 7th Symp. on Principles of Programming Languages, POPL'80* (1980), pp. 163–173.

[9] Harel, D., A. Pnueli and J. Stavi, *Propositional dynamic logic of nonregular programs*, Journal of Computer and System Sciences **26** (1983), pp. 222–243.

[10] Janin, D. and I. Walukiewicz, *Automata for the μ-calculus and related results*, in: J. Wiedermann and P. Hájek, editors, *Proc. 20th Symp. on Math. Foundations of Computer Science, MFCS'95*, LNCS **969** (1995).

[11] Kaivola, R., *On modal μ-calculus and Büchi tree automata*, Information Processing Letters **54** (1995), pp. 17–22.

[12] Lange, M., *Model checking propositional dynamic logic with all extras*, Journal of Applied Logic **??**, pp. ??–??, accepted for publication.

[13] Lange, M., *Local model checking games for fixed point logic with chop*, in: L. Brim, P. Jančar, M. Křetínský and A. Kučera, editors, *Proc. 13th Conf. on Concurrency Theory, CONCUR'02*, LNCS **2421** (2002), pp. 240–254.

[14] Lange, M., *Three notes on the complexity of model checking fixpoint logic with chop* (2005), (submitted).

[15] Lange, M. and R. Somla, *The complexity of model checking higher order fixpoint logic*, in: J. Jedrzejowicz and A. Szepietowski, editors, *Proc. 30th Int. Symp. on Math. Foundations of Computer Science, MFCS'05*, LNCS **3618** (2005), pp. 640–651.

[16] Lange, M. and C. Stirling, *Model checking fixed point logic with chop*, in: M. Nielsen and U. H. Engberg, editors, *Proc. 5th Conf. on Foundations of Software Science and Computation Structures, FOSSACS'02*, LNCS **2303** (2002), pp. 250–263.

[17] Moller, F. and A. Rabinovich, *Counting on CTL*: On the expressive power of monadic path logic*, Information and Computation **184** (2003), pp. 147–159.

[18] Müller-Olm, M., *A modal fixpoint logic with chop*, in: C. Meinel and S. Tison, editors, *Proc. 16th Symp. on Theoretical Aspects of Computer Science, STACS'99*, LNCS **1563** (1999), pp. 510–520.

[19] Pnueli, A., *The temporal logic of programs*, in: *Proc. 18th Symp. on Foundations of Computer Science, FOCS'77* (1977), pp. 46–57.

[20] Rabin, M. O., *Decidability of second-order theories and automata on infinite trees*, Trans. of Amer. Math. Soc. **141** (1969), pp. 1–35.

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

[22] Vardi, M. Y. and P. Wolper, *An automata-theoretic approach to automatic program verification (preliminary report)*, in: *Proc. 1st Symp. on Logic in Computer Science, LICS'86*, IEEE, Washington, DC, 1986 pp. 332–344.

[23] Viswanathan, M. and R. Viswanathan, *A higher order modal fixed point logic*, in: P. Gardner and N. Yoshida, editors, *Proc. 15th Int. Conf. on Concurrency Theory, CONCUR'04*, LNCS **3170** (2004), pp. 512–528.