Specifying Program Properties Using Modal Fixpoint Logics: A Survey of Results

Martin Lange(✉)

University of Kassel, Kassel, Germany
martin.lange@uni-kassel.de

Abstract. The modal μ-calculus is a well-known program specification language with desirable properties like decidability of satisfiability and model checking, axiomatisability etc. Its expressive power is limited by Monadic Second-Order Logic or parity tree automata. Hence, it can only express regular properties.

In this talk I will argue in favour of specification languages whose expressiveness reaches beyond regularity. I will present Viswanathan and Viswanathan's Higher-Order Fixpoint Logic as a natural extension of the modal μ-calculus with highly increased expressive power. We will see how this logic can be used to specify some interesting non-regular properties and then survey results on it with a focus on open questions in this area.

1 The Modal μ-Calculus

The modal μ-calculus $\mathcal{L}_\mu$ [23] is a well-known specification formalism for concurrent, reactive systems. Its formulas are interpreted in states of labeled transition systems. It extends multi-modal logic with restricted second-order quantification in the form of least and greatest fixpoints. This makes it a reasonably expressive temporal logic. It can express properties that are built recursively from basic ones like "*there is a successor s.t. ...*" ($\Diamond$) or "*all successors ...*" ($\Box$) using the usual Boolean operations. Least fixpoints (μ) intuitively correspond to terminating recursion, greatest fixpoints (ν) to not necessarily terminating recursion.

Examples of such recursive properties are "*every path ends in some state without a successor*" or "*there is a path on which infinitely many states satisfy p*". The former is expressed by $\varphi_{\mathsf{end}} := \mu X.\Box X$. A helpful tool for understanding such formulas is the fixpoint unfolding principle $\kappa X.\varphi \equiv \varphi[\kappa X.\varphi/X]$ for $\kappa \in \{\mu, \nu\}$ stating that the set of states satisfying $\kappa X.\varphi$ is indeed a fixpoint of the mapping that takes a set X and returns those satisfying $\varphi(X)$. With this principle we get that

$$\mu X.\Box X \;\equiv\; \Box\mu X.\Box X \;\equiv\; \Box\Box\mu X.\Box X \;\equiv\; \ldots$$

Knowing that a state s that has no successors satisfies $\Box\psi$ for any ψ, one can see that any state that is at the source of finite paths only, satisfies $\mu X.\Box X$.

Md. A. Khan and A. Manuel (Eds.): ICLA 2019, LNCS 11600, pp. 42–51, 2019.
https://doi.org/10.1007/978-3-662-58771-3_5

A second helpful tool for understanding formulas that comes on handy at this point is the characterisation of $\mathcal{L}_\mu$'s model checking problem in terms of parity games [33]. Here, two players play with a token on a state s in a transition system and another on φ's syntax tree in order to find out whether s satisfies φ or not. Verifier chooses disjuncts whenever a disjunction is reached, and successor states whenever a $\Diamond$ is reached, likewise Refuter chooses at conjunctions and $\Box$-formulas. When reaching a fixpoint variable the game simply continues with the defining fixpoint formula. Verifier wins in a situation where the currently selected state blatantly satisfies the currently selected formula. In case of $\mu X.\Box X$ above, only Refuter makes choices by continuously selecting successor states. If he follows a (maximal) finite path $s \dots t$ then this will ultimately end in a situation with t and $\Box X$ being selected, and since t is assumed to have no successors, Verifier wins, indicating that the original formula holds in s.

However, if there is an infinite path starting in s, then Refuter can traverse this and the resulting play of the game is infinite. Then the winner is determined by the type of the outermost fixpoint that gets traversed infinitely often. In the case of $\mu X.\Box X$ there is only one candidate – a least fixpoint – which makes Refuter the winner. As an exercise one may check that the second property named above about the existence of an infinite path is expressed by the formula $\nu X.\mu Y.(p \wedge \Diamond X) \vee \Diamond Y$. The key is to see that the game rules make Verifier select a path, and she can only traverse through the outer greatest fixpoint if this path has infinitely many states satisfying p. Otherwise any play will eventually only traverse through the inner least fixpoint which would make Refuter the winner again.

$\mathcal{L}_\mu$ is well understood in terms of its expressivity and the computational complexity of the standard decision problems associated with a (temporal) logic. We quickly recall the most important results on $\mathcal{L}_\mu$, for further and more detailed overviews see also [4–6] and [14, Chap. 8].

- $\mathcal{L}_\mu$ respects, like multi-modal logic, bisimulation-invariance, i.e. it cannot distinguish bisimilar models. Despite sounding negatively, this is a good property to have since program specification formalisms should not distinguish states of transition systems that exhibit the same temporal behaviour.
- Many standard temporal and other logics can be embedded into $\mathcal{L}_\mu$, for instance CTL and PDL with simple linear translations, but also CTL* (and therefore LTL) with exponential translations [13], [14, Theorem 10.2.7].
- On trees, $\mathcal{L}_\mu$ is equi-expressive to alternating parity tree automata (APT) and, since alternation can be eliminated, therefore also to nondeterministic tree automata [15]. Bearing the first point in mind, this statement is not quite accurate since APT in their usual form are aware of directions in trees and can therefore specify non-bisimulation-invariant properties. To be precise, $\mathcal{L}_\mu$ is in fact only equi-expressive to APT over classes of ranked trees of bounded branching-degree where it has access to a specific successor, not just some. Over the class of all trees, $\mathcal{L}_\mu$ is equi-expressive to so-called symmetric APT [17,38].

There is of course also a well-known connection between tree automata and Monadic Second-Order Logic (MSO) [32]. Even without automata at hand it is easy to see that $\mathcal{L}_\mu$ can be embedded into MSO. The cannot hold as MSO is not bisimulation-invariant. However, it turns out that $\mathcal{L}_\mu$ is as expressive as MSO when restricted to bisimulation-invariant properties [20].

- Satisfiability is decidable and ExpTime-complete. The upper bound is a consequence of the linear translation into APT and an exponential emptiness test there [15]. The lower bound is inherited from PDL for instance [18].
- There are relatively simple sound and complete axiomatic systems for $\mathcal{L}_\mu$ [1,23], but establishing completeness is typically a challenging task [17,36].
- Model checking over finite transition systems is trivially decidable. It is in fact computationally equivalent to the problem of solving a parity game [33,35]. The best lower bound is known to date is P-hardness since $\mathcal{L}_\mu$ can express winning in a reachability game. The currently best known upper bounds – found only recently after a long time of research in this area – are quasi-polynomial [12,21,27].
  Model checking is even decidable over richer classes of infinite transition systems: for pushdown systems it is ExpTime-complete [37], for higher-order recursion schemes it is of non-elementary complexity [31].
- An interesting source of computational and pragmatic complexity in $\mathcal{L}_\mu$ formulas is the *alternation depth* [16,30], measuring the degree to which recursion is defined by entangling least and greatest fixpoints. It is the determining element in the asymptotic complexity of many algorithms, being exponential in it. It is also a major source of obfuscation when trying to understand the property expressed by a given $\mathcal{L}_\mu$ formula. It is therefore interesting to know how much fixpoint alternation is necessary for writing down all definable properties. It turned out that the fixpoint alternation hierarchy is strict [2,7]: for any alternation depth there are definable properties that cannot be specified using this depth only.

A consequence of $\mathcal{L}_\mu$'s connection to APT and MSO is the fact that it can only define *regular* properties. There are, however, many non-regular properties which are more or less interesting, depending on potential application areas. Typical examples include "*all executions of a program terminate at the same moment*", "*no two executions can be distinguished from the outside*", "*there is no underflow in an unbounded buffer*", "*there is a maximal path of length* n^2 *for some* n", etc.

There are a few proposals for modal logics that are capable of expressing non-regular properties, for instance PDL[CFL] [19], FLC [29] and HFL [34]. FLC extends $\mathcal{L}_\mu$, and HFL (vastly) extends FLC. PDL[CFL] is orthogonal to $\mathcal{L}_\mu$ in terms of expressive power but is already captured by FLC. In the following we will turn our attention to *Higher-Order Fixpoint Logic* (HFL), the most expressive among these. We compare it to $\mathcal{L}_\mu$ and its properties as laid out above. We will explain how the increase in expressive power comes at a very high price, not just computationally but also in terms of the number of questions on certain aspects of HFL that remain unanswered to date.

2 Higher-Order Fixpoint Logic

We refrain from giving a detailed definition of the syntax and semantics of the logic HFL. Instead we concentrate on the presentation of those principles that are used there, especially for the semantics. The goal of this exposition is not detailed mathematical completeness but the intuition behind the constructs in a modal fixpoint logic that achieves high expressive power. For a formal definition see [34].

HFL results from a merger between the modal μ-calculus with a simply typed λ-calculus. Its formulas are typed in a simple type system that inductively builds types from a single base type $\bullet$ using three function type constructors:

$$\sigma, \tau ::= \bullet \mid \sigma^{v} \to \tau \qquad v ::= + \mid - \mid 0$$

Formulas of base type are predicates as in $\mathcal{L}_\mu$; formally the type represents the powerset lattice of the set of states of a given transition system. The type $\sigma^v \to \tau$ then represents functions from objects of type σ to objects of type τ which are monotone (if $v = +$), antitone (if $v = -$) resp. unrestricted (if $v = 0$).

Formulas are given by the following grammar.

$$\varphi ::= p \mid X \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \neg\varphi \mid \langle a \rangle \varphi \mid [a]\varphi \mid \mu X^{\tau}.\varphi \mid \nu X^{\tau}.\varphi \mid \lambda X^{\tau}.\varphi \mid \varphi\ \varphi$$

where X is a variable, p is an atomic proposition interpreted by a set of states in a transition system, a is an action interpreted as a set of edges in a transition system, and τ is a type. However, not every object formed in this way is a formula. The type system guarantees well-formedness of formulas; it mainly ensures that

- in an application of the form $\varphi\ \psi$ the formula ψ has some type σ, and φ has a type $\sigma \to \tau$, and
- in a fixpoint formula $\kappa X.\varphi$, the mapping $X \mapsto \varphi(X)$ is monotone in order to guarantee the existence of least and greatest fixpoints.

The *order* of a type is defined via $ord(\bullet) = 0$ and $ord(\sigma^v \to \tau) = \max\{ord(\tau), 1 + ord(\sigma)\}$. The fragment HFL^k, $k \geq 0$, consists of all formulas of type $\bullet$ which use types of order at most k.

Consider the formula

$$\varphi \ := \ \lambda f^{\bullet^0 \to \bullet}.\lambda g^{\bullet^0 \to \bullet}.\lambda X^{\bullet}.f(g(X)) \ .$$

Its type is

$$(\bullet^0 \to \bullet)^+ \to (\bullet^0 \to \bullet)^+ \to \bullet^+ \to \bullet$$

and φ is therefore a formula of order 2.

The semantics of a formula with type τ is a function of type τ in a transition system. Its definition is straightforward given that each type induces a complete lattice of pointwise ordered (monotone/antitone/unrestricted) functions in a transition system. Fixpoint formulas can therefore be given meaning through

the Knaster-Tarski Theorem. Instead of listing the formal definitions here we present some examples of formulas with the aim of giving some intuition on how to specify complex program properties in HFL. The important concepts for this are the fixpoint unfolding principle and β-reduction: $(\lambda X.\varphi)\ \psi \equiv \varphi[\psi/X]$.

We will use the following abbreviations with appropriate type annotation which are left out for brevity here.

$$g \circ f \ :=\ \lambda X.g\ (f\ X), \quad f^i \ :=\ \underbrace{f \circ \ldots \circ f}_{i \text{ times}}, \quad \Diamond \ :=\ \lambda X.\Diamond X$$

Example 1. Consider the formula

$$\varphi_{\mathsf{qpath}} \ :=\ \mu F.\lambda g.\lambda f.(g\ \Box\mathtt{ff}) \vee F\ (g \circ f^2 \circ \Diamond)\ (f \circ \Diamond)$$

Using fixpoint unfolding and β-reduction we see that $\varphi_{\mathsf{qpath}}\ \Diamond\ \Diamond$ unfolds to

$$\Diamond\Box\mathtt{ff} \vee (\varphi_{\mathsf{qpath}}\ \Diamond^4\ \Diamond^2) \ \equiv\ \Diamond\Box\mathtt{ff} \vee \Diamond^4\Box\mathtt{ff} \vee (\varphi_{\mathsf{qpath}}\ \Diamond^9\ \Diamond^3)$$

and so on. In fact, after unfolding n times and β-reducing appropriately we obtain $\bigvee_{i=1}^{n} \Diamond^{i^2}\Box\mathtt{ff} \vee (\varphi_{\mathsf{qpath}}\ \Diamond^{(n+1)^2}\ \Diamond^{n+1})$. This uses the fact that $(n+1)^2 = n^2+2n+1$.

Hence, in HFL^2 it is possible to define the property of having a maximal path of quadratic length.

Example 2. The property of a tree being balanced can be defined in HFL^1 already. Note that being balanced means there is some n such that every path of length n ends in a state without successors, and that no path of shorter length does so. This is defined by

$$\Big(\mu F.\lambda X.X \vee (F\ (\Diamond\mathtt{tt} \wedge \Box X))\Big)\ \Box\mathtt{ff}$$

which, again, can be unfolded and reduced to yield

$$\bigvee_{i \geq 0} \underbrace{\Diamond\mathtt{tt} \wedge \Box(\Diamond\mathtt{tt} \wedge \Box(\ldots \wedge \Diamond\mathtt{tt} \wedge \Box}_{i \text{ times}} \Box\mathtt{ff}))$$

Example 3. A similar construction principle is used in $\varphi_{\mathsf{unb}} := (\nu F.\lambda X.X \wedge (F\ \Diamond X))\ \mathtt{tt}$. It unfolds to $\bigwedge_{i=0} \Diamond^i\mathtt{tt}$ and therefore states that are paths of unbounded length. Note that this is not the same as stating there is an infinite path.

Example 4. Note that the context-free grammar $S \to \mathsf{out} \mid \mathsf{in}\, S\, S$ generates the language of all words that have one more out than *in*'s but no prefix does so. It represents the runs of potentially unbounded buffers that see an underflow. This grammar can immediately be transferred into an HFL^1 formula:

$$\neg\Big(\big(\mu S.\lambda X.\langle\mathsf{out}\rangle X \vee \langle\mathsf{in}\rangle(S\ (S\ X))\big)\ \mathtt{tt}\Big)$$

states that no execution is of a form that falls into this grammar. Hence, it states that all runs of a buffer do not underflow.

2.1 Results on HFL

We survey results on HFL that are known and problems that are still open, comparing this in particular to the situation with $\mathcal{L}_\mu$.

Embeddings. HFL^s ubsumes $\mathcal{L}_\mu$ in the simple sense that $\mathcal{L}_\mu$ is HFL^0, even syntactically. HFL^1 also subsumes the aforementioned FLC [34] with, in turn, subsumes PDL[CFL] [26].

Model Properties. HFL retains bisimulation-invariance [34]. However, HFL^1 already does not possess the finite model property anymore. Consider the formula $\varphi_{\mathsf{unb}} \wedge \varphi_{\mathsf{end}}$. It requires paths of unbounded length to exist but every path to be finite. This is satisfiable but not in a finite model.

Satisfiability. Strongly connected to the loss of the finite model property is the high undecidability of satisfiability checking, even for HFL^1. It is at least Σ^1_1-hard: this is proved originally for PDL[CFL] [19] and then transferred to stronger logics.

So far, no non-trivial fragments of HFL^w ith a decidable satisfiability problem have been found.

Proof Systems. The situation on the proof-theoretic side of a theory of higher-order modal fixpoint logics is even more bleak. It is not known whether there are fragments of HFL^o r even some HFL^k which can be axiomatised in a sound and complete way, not even when giving up on completeness (looking for non-trivial fragments in that case of course).

Model Checking. The model checking problem for HFL over finite transition systems is decidable and, roughly speaking, k-ExpTime-complete for formulas of order k.[1] The upper bound is obtained in a more or less straightforward way by computing the semantics of a formula bottom-up, the lower bound can be obtained using standard reductions for k-ExpTime-complete problems, for instance tiling game problems [3].

Given that HFL has complete model checking problems for every level of the exponential time hierarchy, it is a fair question to ask whether something similar holds for the exponential space hierarchy. The answer is positive: it is possible to identify a syntactic criterion on formulas called *tail recursion* such that the model checking problem for HFL^k formulas restricted in this way becomes $(k-1)$-ExpSpace-complete [10].

There seems to be no chance to extend the decidability result to any meaningful class of infinite-state systems. One can show undecidability of model checking

[1] This does not hold for $k = 0$, i.e. $\mathcal{L}_\mu$. It also requires an assumption of a bound on the number of arguments a function can take. Otherwise the upper bound is one exponential higher.

for FLC, resp. HFL^1 formulas over BPA processes already [29]. It remains to be seen whether there is in fact a – necessarily very small – class of infinite-state transition systems for which HFL^1 model checking is decidable.

On the other hand, there is a connection between model checking higher-order formulas and higher-order model checking: the problems of model checking a $\mathcal{L}_\mu$ formula over a higher-order recursion scheme is computationally equivalent to the problem of model checking an HFL formula over a finite transition system [22]. This can be seen as a trade-off between higher order on the formula side and higher order on the model side. It is worth noting that the translations preserve maximal order.

Automata for HFL. There is a counterpart to HFL in the world of automata. Bruse has been able to come up with an automaton model that captures HFL in the sense that every formula is equivalent to an automaton and vice-versa [8]. The model is called *Alternating Parity Krivine Automata* (APKA) and is an extension of APT that uses the mechanisms of the Krivine machine to handle higher-order functions (using a call-by-name technique). The main combinatorial difficulty in designing such an automaton model is the correct capturing of the interplay of fixpoints in the presence of higher-order features by an appropriate acceptance condition. Bruse has shown [9] that in the case of HFL^1, one can use a neater acceptance condition which is closer to the stair-parity condition [24] used in visibly pushdown games [28].

It remains to be seen whether this neater condition can be extended to fragments beyond first-order functions. We also suspect that Boolean alternation cannot be eliminated from APKA as it can be for APT. There is, however, no proof of this or the contrary.

Fixpoint Alternation. The richness of HFL as opposed to $\mathcal{L}_\mu$ opens up a variety of questions regarding the strictness or collapse of fixpoint alternation hierarchies. Besides the obvious restriction to particular classes of models one can now also ask whether the fixpoint alternation hierarchy in some HFL^k, say $\mathcal{L}_\mu$ for instance, despite being strict in itself, collapses in some HFL^k for $k > 0$. I.e. it is conceivable that one may be able to reduce fixpoint alternation when one is willing to pay with higher function orders. This is indeed true in some case, namely finite models. One can express the Kleene iteration of length at most ω of a greatest fixpoint at order 0 using a least fixpoint at order 1 and an embedded but non-alternating greatest fixpoint of order 0. Hence, over finite models, every $\mathcal{L}_\mu$ formula is equivalent to an alternation-free HFL^1 formula.

One has to admit, though, that fixpoint alternation is not easy to define syntactically. Using β-expansion it is always possible to decouple nested fixpoints so that syntactically they look like they are not dependent on each other. This, however, only shows that the definition of fixpoint alternation that is used for $\mathcal{L}_\mu$, is too coarse for HFL. Bruse has suggested to define fixpoint alternation via the minimal number of priorities used in equivalent APKA. This way he has

managed to show that the fixpoint alternation hierarchy is strict within HFL^1 [9], resembling similar proofs for $\mathcal{L}_\mu$ [2] and FLC [25].

The trick of trading in fixpoint alternation for higher order can be extended slightly beyond order 0 [11]. Here, simulating the Kleene iteration of a greatest fixpoint is more difficult because one has to test two first-order functions for equality, rather than two sets. This would in principle require the enumeration of all possible sets which HFL^2 cannot do due to bisimulation-invariance. It turns out, though, that it suffices to enumerate all modal formulas as possible arguments to such first-order functions.

In summary, results on fixpoint alternation in HFL are sparse. In particular, it is currently open whether general strictness results or, equivalently, strictness over trees, can be extended to order higher than 1. On the other hand, it is equally open whether collapse results based on the trade-in of alternation against higher orders can be extended beyond low orders.

References

1. Afshari, B., Leigh, G.E.: Cut-free completeness for modal mu-calculus. In: Proceedings of the 32nd ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, pp. 1–12. IEEE (2017)
2. Arnold, A.: The modal μ-calculus alternation hierarchy is strict on binary trees. RAIRO Theor. Inform. Appl. **33**, 329–339 (1999)
3. Axelsson, R., Lange, M., Somla, R.: The complexity of model checking higher-order fixpoint logic. Log. Methods Comput. Sci. **3**, 1–33 (2007)
4. Bradfield, J., Stirling, C.: Modal logics and μ-calculi: an introduction. In: Bergstra, J., Ponse, A., Smolka, S. (eds.) Handbook of Process Algebra, pp. 293–330. Elsevier, New York (2001)
5. Bradfield, J., Stirling, C.: Modal mu-calculi. In: Blackburn, P., van Benthem, J., Wolter, F. (eds.) Handbook of Modal Logic: Studies in Logic and Practical Reasoning, vol. 3, pp. 721–756. Elsevier, New York (2007)
6. Bradfield, J., Walukiewicz, I.: The mu-calculus and model checking. In: Clarke, E., Henzinger, T., Veith, H., Bloem, R. (eds.) Handbook of Model Checking, pp. 871–919. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-10575-8_26
7. Bradfield, J.C.: The modal mu-calculus alternation hierarchy is strict. In: Montanari, U., Sassone, V. (eds.) CONCUR 1996. LNCS, vol. 1119, pp. 233–246. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-61604-7_58
8. Bruse, F.: Alternating Parity Krivine Automata. In: Csuhaj-Varjú, E., Dietzfelbinger, M., Ésik, Z. (eds.) MFCS 2014. LNCS, vol. 8634, pp. 111–122. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44522-8_10
9. Bruse, F.: Alternation is strict for higher-order modal fixpoint logic. In: Proceedings of the 7th International Symposium on Games, Automata, Logics and Formal Verification, GandALF 2016. EPTCS, vol. 226, pp. 105–119 (2016)
10. Bruse, F., Lange, M., Lozes, E.: Space-efficient fragments of higher-order fixpoint logic. In: Hague, M., Potapov, I. (eds.) RP 2017. LNCS, vol. 10506, pp. 26–41. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-67089-8_3
11. Bruse, F., Lange, M., Lozes, E.: Collapses of fixpoint alternation hierarchies in low type-levels of higher-order fixpoint logic. In: Proceedings Workshop on Programming and Reasoning on Infinite Structures, PARIS 2014 (2018)

12. Calude, C.S., Jain, S., Khoussainov, B., Li, W., Stephan, F.: Deciding parity games in quasipolynomial time. In: Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, pp. 252–263. ACM (2017)
13. Dam, M.: CTL* and ECTL* as fragments of the modal μ-calculus. TCS **126**(1), 77–96 (1994)
14. Demri, S., Goranko, V., Lange, M.: Temporal Logics in Computer Science. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge (2016)
15. Emerson, E.A., Jutla, C.S.: Tree automata, μ-calculus and determinacy. In: Proceedings of the 32nd Symposium on Foundations of Computer Science, San Juan, pp. 368–377. IEEE (1991)
16. Emerson, E.A., Lei, C.L.: Efficient model checking in fragments of the propositional μ-calculus. In: Symposion on Logic in Computer Science, Washington, D.C., pp. 267–278. IEEE (1986)
17. Enqvist, S., Seifan, F., Venema, Y.: Completeness for the modal μ-calculus: separating the combinatorics from the dynamics. Theor. Comput. Sci. **727**, 37–100 (2018)
18. Fischer, M.J., Ladner, R.E.: Propositional dynamic logic of regular programs. J. Comput. Syst. Sci. **18**(2), 194–211 (1979)
19. Harel, D., Pnueli, A., Stavi, J.: Propositional dynamic logic of nonregular programs. J. Comput. Syst. Sci. **26**(2), 222–243 (1983)
20. Janin, D., Walukiewicz, I.: On the expressive completeness of the propositional mu-calculus with respect to monadic second order logic. In: Montanari, U., Sassone, V. (eds.) CONCUR 1996. LNCS, vol. 1119, pp. 263–277. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-61604-7_60
21. Jurdzinski, M., Lazic, R.: Succinct progress measures for solving parity games. In: Proceedings of the 32nd ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, pp. 1–9. IEEE (2017)
22. Kobayashi, N., Lozes, É., Bruse, F.: On the relationship between higher-order recursion schemes and higher-order fixpoint logic. In Proceedings of POPL 2017, pp. 246–259. ACM (2017)
23. Kozen, D.: Results on the propositional μ-calculus. TCS **27**, 333–354 (1983)
24. Lange, M.: Local model checking games for fixed point logic with chop. In: Brim, L., Křetínský, M., Kučera, A., Jančar, P. (eds.) CONCUR 2002. LNCS, vol. 2421, pp. 240–254. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45694-5_17
25. Lange, M.: The alternation hierarchy in fixpoint logic with chop is strict too. Inf. Comput. **204**(9), 1346–1367 (2006)
26. Lange, M., Somla, R.: Propositional dynamic logic of context-free programs and fixpoint logic with chop. Inf. Process. Lett. **100**(2), 72–75 (2006)
27. Lehtinen, K.: A modal μ perspective on solving parity games in quasi-polynomial time. In: Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, pp. 639–648. ACM (2018)
28. Löding, C., Madhusudan, P., Serre, O.: Visibly pushdown games. In: Lodaya, K., Mahajan, M. (eds.) FSTTCS 2004. LNCS, vol. 3328, pp. 408–420. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30538-5_34
29. Müller-Olm, M.: A modal fixpoint logic with chop. In: Meinel, C., Tison, S. (eds.) STACS 1999. LNCS, vol. 1563, pp. 510–520. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-49116-3_48
30. Niwiński, D.: Fixed point characterization of infinite behavior of finite-state systems. Theor. Comput. Sci. **189**(1–2), 1–69 (1997)

31. Ong, C.-H.L.: On model-checking trees generated by higher-order recursion schemes. In: Proceedings of the 21st IEEE Symposium on Logic in Computer Science, LICS 2006, pp. 81–90. IEEE Computer Society (2006)
32. Rabin, M.O.: Decidability of second-order theories and automata on infinite trees. Trans. Am. Math. Soc. **141**, 1–35 (1969)
33. Stirling, C.: Local model checking games (extended abstract). In: Lee, I., Smolka, S.A. (eds.) CONCUR 1995. LNCS, vol. 962, pp. 1–11. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-60218-6_1
34. Viswanathan, M., Viswanathan, R.: A higher order modal fixed point logic. In: Gardner, P., Yoshida, N. (eds.) CONCUR 2004. LNCS, vol. 3170, pp. 512–528. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28644-8_33
35. Walukiewicz, I.: Monadic second order logic on tree-like structures. In: Puech, C., Reischuk, R. (eds.) STACS 1996. LNCS, vol. 1046, pp. 399–413. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-60922-9_33
36. Walukiewicz, I.: Completeness of Kozen's axiomatisation of the propositional μ-calculus. Inf. Comput. **157**(1–2), 142–182 (2000)
37. Walukiewicz, I.: Pushdown processes: games and model-checking. Inf. Comput. **164**(2), 234–263 (2001)
38. Wilke, T.: Alternating tree automata, parity games, and modal μ-calculus. Bull. Belgian Math. Soc. **8**(2), 359–391 (2001)