Game-Based Local Model Checking for the Coalgebraic μ-Calculus

Daniel Hausmann
Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany
daniel.hausmann@fau.de

Lutz Schröder
Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany
lutz.schroeder@fau.de

Abstract

The coalgebraic μ-calculus is a generic framework for fixpoint logics with varying branching types that subsumes, besides the standard relational μ-calculus, such diverse logics as the graded μ-calculus, the monotone μ-calculus, the probabilistic μ-calculus, and the alternating-time μ-calculus. In the present work, we give a local model checking algorithm for the coalgebraic μ-calculus using a coalgebraic variant of parity games that runs, under mild assumptions on the complexity of the so-called one-step satisfaction problem, in time p^k where p is a polynomial in the formula and model size and where k is the alternation depth of the formula. We show moreover that under the same assumptions, the model checking problem is in NP $\cap$ coNP, improving the complexity in all mentioned non-relational cases. If one-step satisfaction can be solved by means of small finite games, we moreover obtain standard parity games, ensuring quasi-polynomial run time. This applies in particular to the monotone μ-calculus, the alternating-time μ-calculus, and the graded μ-calculus with grades coded in unary.

2012 ACM Subject Classification Theory of computation → Modal and temporal logics; Theory of computation → Verification by model checking

Keywords and phrases Model checking, μ-calculus, coalgebraic logic, graded μ-calculus, probabilistic μ-calculus, parity games

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2019.35

Funding Work forms part of the DFG project *Generic Algorithmic Methods for Modal and Hybrid Logics* (SCHR 1118/5-3).

1 Introduction

One of the most central and established computation tasks in the verification of concurrent software is *model checking*, i.e. to determine whether a given system state satisfies a temporal specification (e.g. [2]). The complexity of model checking is generally fairly sensitive to variations in the logic, more so than satisfiability checking, which for fairly wide ranges of branching time temporal logics (such as PDL, CTL and the μ-calculus) tends to be ExpTime-complete. For instance, CTL allows model checking in polynomial time, while LTL (and more generally CTL*) model checking is PSpace-complete (e.g. [27]). In fact, the input of model checking is naturally split into two parts, the model and the formula, and the complexity analysis is typically phrased in terms of model size and formula size separately.

Model checking for μ-calculus formulae typically proceeds by a reduction to computing winning regions in parity games, and in fact the two problems are linear-time equivalent [11]. In consequence, the current best upper bound for the time complexity of μ-calculus model checking has recently dropped when Calude et al. [4] showed that parity games can be solved in quasi-polynomial time; in fact, further improvement seems possible as the exact complexity of parity game solving remains open.

Besides model checking relational systems, there is a long-standing and growing interest in systems with additional structure, e.g. probabilities, weights, multi-player games, or neighbourhoods. To name just a few concrete examples, the alternating-time μ-calculus [1] is interpreted over concurrent game structures; the (two-valued) probabilistic μ-calculus [6, 21] over Markov chains, and the monotone μ-calculus [12] (the ambient fixpoint logic of concurrent dynamic logic (CPDL) [25] and Parikh's game logic [23]) over (monotone) neighbourhood structures. The graded μ-calculus [20], standardly interpreted over relational structures, has an equivalent and more natural semantics over integer-weighted structures [10]. As a unifying framework for such fixpoint logics beyond relational semantics, the *coalgebraic μ-calculus* [6] has emerged. It works within the paradigm of *universal coalgebra* [26], i.e. encapsulates the system type as a set functor and systems as coalgebras for this functor; the interpretation of modalities is based on *predicate liftings* as used in the broader field of *coalgebraic logic* [8].

Our present contribution is a generic *local* model checking algorithm for the coalgebraic μ-calculus, which can be instantiated easily to concrete μ-calculi including all the ones mentioned above, where by *local* we mean that the algorithm allows establishing satisfaction of a formula in a given state without calculating full extensions of all subformulae across all states. Under mild assumptions on the concrete functor and predicate liftings defining the logic, our algorithm runs in time p^k where p is a polynomial in the size of the formula and the model and k is the alternation depth of the formula; in particular, the algorithm runs in polynomial time on alternation-free coalgebraic μ-calculi, or more generally on formulae of bounded alternation depth. Further analysis shows moreover that under the same assumptions, the model checking problem is in fact in $\mathrm{NP} \cap \mathrm{coNP}$. The algorithm is based on a coalgebraic generalization of parity games, in which some steps consist in calls to a *one-step satisfaction* checker that essentially just implements the modalities. In some cases, notably the monotone μ-calculus, the alternating-time μ-calculus, and the graded μ-calculus, we can replace one-step satisfaction checking with suitable finite *one-step satisfaction games*, obtaining a standard (rather than coalgebraic) parity game. Exploiting the mentioned recent results on parity game solving, we obtain quasi-polynomial runtime in cases where these one-step satisfaction games are sufficiently small; in particular, we show that the monotone μ-calculus, the alternating-time μ-calculus, and the graded μ-calculus with unary coding of grades all admit model checking in quasi-polynomial time, to our knowledge new results.

Related Work. Model checking games for the monotone μ-calculus have been studied by Hansen et al. [15], without complexity analysis. For the alternating-time μ-calculus, Alur et al. [1] describe a model checking procedure that works essentially by fixpoint iteration, and runs in time $\mathcal{O}((m \cdot l)^{k+1})$ where m is the number of transitions, l the length of the formula, and k its alternation depth. An upper bound $\mathrm{UP} \cap \mathrm{coUP}$ for model checking the probabilistic μ-calculus is stated in [21]. The complexity of model checking the graded μ-calculus is mentioned by Ferrante et al. [13] in work otherwise concerned with the more complex problem of *module* checking, giving an upper bound ExpTime for numbers in both systems and formulae coded in binary. We improve this bound to $\mathrm{NP} \cap \mathrm{coNP}$ under binary coding, and to quasi-polynomial time under unary coding, and moreover give a deterministic algorithm that is exponential only in the alternation depth.

Cîrstea et al. [9] provide an automata-based model checking procedure for quantitative linear time logics over systems with weights in a partial commutative semiring, using coalgebraic methods. Satisfiability checking in the coalgebraic μ-calculus has been shown to be in ExpTime [6, 17]. For purposes of the present paper, the most relevant piece of related work is Hasuo et al.'s [16] model checking algorithm for the coalgebraic μ-calculus, which

is based on fixpoint computation using progress measures in a highly general setting. The bound on the run time stated in [16] has roughly the form p^r where p is a polynomial in the number of states of the model and the size of the formula understood as the number of nodes in its parse tree, and r is the number of least fixpoint operators. The bound is thus independent of the size of the transition structure of models and the actual representation size of the formula (which may in general contain, e.g., integer or rational numbers); this can clearly hold only under strong assumptions on both the system type and the syntax of formulae, mentioned implicitly in [16] on p. 728. In particular, the overall representation size of models must be polynomially bounded in the number of states. The analyis in [16] therefore does not apply to our main examples; e.g. a concurrent game structure with just one state can have an unbounded number of transitions; integer-weighted transition systems can involve unboundedly large integer numbers; and a monotone neighbourhood frame can have exponentially more (even minimal) neighbourhoods than states. Another important difference is that we make the intermediate game theoretic constructions in the algorithm explicit, while game constructions are eliminated in favour of a direct implementation of progress measures in [16]. Immediate benefits include exponential dependence only on alternation depth instead of number of least fixpoint operators, materializing an improvement conjectured by Hasuo et al.; the local nature of our model checking algorithm; and quasi-polynomial runtime for the above-mentioned cases admitting small one-step satisfaction games. As a long-term benefit, we expect algorithmic improvements paralleling the development in standard parity games also for our coalgebraic parity games, and hence for the complexity of coalgebraic μ-calculus model checking in general.

2 The Coalgebraic μ-Calculus

We proceed to recall basic definitions and examples in universal coalgebra [26] and the coalgebraic μ-calculus [6].

The abstraction principle underlying universal coalgebra is to encapsulate system types as functors, for our present purposes on the category of sets. Such a functor $T : \mathsf{Set} \to \mathsf{Set}$ maps every set X to a set $T(X)$, and every map $f : X \to Y$ to a map $Tf : T(X) \to T(Y)$, preserving identities and composition. We think of $T(X)$ as a type of structured collections over X; a basic example is the covariant powerset functor $\mathcal{P}$, which assigns to each set its powerset and acts on maps by taking forward image. Systems of the intended type are then cast as T-*coalgebras* (C, ξ) (or just ξ) consisting of a set C of *states* and a *transition map* $\xi : C \to T(C)$, thought of as assigning to each state $x \in C$ a structured collection $\xi(x) \in T(C)$ of successors. E.g. a $\mathcal{P}$-coalgebra $\xi : C \to \mathcal{P}(C)$ assigns to each state a set of successors; that is, $\mathcal{P}$-coalgebras are transition systems. We will see additional examples later. A coalgebra is *finite* if its state set is finite.

Following the paradigm of *coalgebraic logic* [8], we fix a set Λ of modal operators (in principle of any finite arity but restricted to unary modalities in the technical development for the sake of readability; our proofs generalize by essentially writing more indices), which we interpret over T-coalgebras for a functor T as *predicate liftings*, i.e. natural transformations

$$[\![\heartsuit]\!]_X : 2^X \to 2^{T(X)} \qquad \text{for } \heartsuit \in \Lambda.$$

Here, the index X, omitted when clear from the context, ranges over all sets; 2^X denotes the set of maps $X \to 2$ into the two-element set $2 = \{\bot, \top\}$, isomorphic to the powerset of X (i.e. 2^- is the *contravariant powerset functor*); and naturality means that $[\![\heartsuit]\!]_X(f^{-1}[A]) =$

$(Tf)^{-1}[[\![\heartsuit]\!]_Y(A)]$ for $f : X \to Y$ and $A \in 2^Y$. Thus, the predicate lifing $[\![\heartsuit]\!]$ indeed lifts predicates on a base set X to predicates on the set $T(X)$. Two standard examples for $T = \mathcal{P}$ are the predicate liftings for the standard $\Box$ and $\Diamond$ modalities, given by

$$[\![\Box]\!]_X(A) = \{B \in \mathcal{P}(X) \mid B \subseteq A\} \quad \text{and} \quad [\![\Diamond]\!]_X(A) = \{B \in \mathcal{P}(X) \mid A \cap B \neq \emptyset\}.$$

Since we mean to form fixpoint logics, we need to require that every $[\![\heartsuit]\!]$ is *monotone*, that is, $A \subseteq B \subseteq X$ implies $[\![\heartsuit]\!]_X(A) \subseteq [\![\heartsuit]\!]_X(B)$. To support negation, we assume moreover that Λ is closed under *duals*, i.e. for each $\heartsuit \in \Lambda$ we have $\overline{\heartsuit} \in \Lambda$ such that $[\![\overline{\heartsuit}]\!]_X(A) = T(X) \setminus [\![\heartsuit]\!]_X(X \setminus A)$, chosen so that $\overline{\overline{\heartsuit}} = \heartsuit$ (e.g. $\overline{\Box} = \Diamond$, $\overline{\Diamond} = \Box$).

To introduce the syntax of the *coalgebraic μ-calculus*, we fix a set Var of *fixpoint variables*. We let the meta-variable η range over the standard fixpoint operators μ (least fixpoint), ν (greatest fixpoint). The set of *formulae* ϕ, ψ is then defined by the grammar

$$\psi, \phi := \top \mid \bot \mid \psi \vee \phi \mid \psi \wedge \phi \mid \heartsuit\psi \mid X \mid \eta X.\psi \qquad (\heartsuit \in \Lambda, X \in \mathsf{Var}).$$

Note that the grammar does not include propositional atoms; these can, if desired, be treated as nullary modalities (Example 1). Negation is not included explicitly but can be defined, as usual, by taking negation normal forms. Fixpoint operators bind their variables, giving rise to the usual notions of bound and free variables; we denote the set of free variables of a formula ψ by $\mathsf{FV}(\psi)$. A formula ψ is *closed* if $\mathsf{FV}(\psi) = \emptyset$. Given a T-coalgebra $\xi : C \to T(C)$ and a valuation $\sigma : \mathsf{Var} \to \mathcal{P}(C)$, the extension

$$[\![\phi]\!]_\sigma \subseteq C$$

of a formula ϕ is defined recursively by the expected clauses for the propositional operators ($[\![\top]\!]_\sigma = C$; $[\![\bot]\!]_\sigma = \emptyset$; $[\![\phi \wedge \psi]\!]_\sigma = [\![\phi]\!]_\sigma \cap [\![\psi]\!]_\sigma$; $[\![\phi \vee \psi]\!]_\sigma = [\![\phi]\!]_\sigma \cup [\![\psi]\!]_\sigma$); by $[\![X]\!]_\sigma = \sigma(X)$ and

$$[\![\heartsuit\psi]\!]_\sigma = \xi^{-1}[[\![\heartsuit]\!]([\![\psi]\!]_\sigma)];$$

and by $[\![\mu X.\psi]\!]_\sigma = \mathsf{LFP}[\![\psi]\!]^X_\sigma$, $[\![\nu X.\psi]\!]_\sigma = \mathsf{GFP}[\![\psi]\!]^X_\sigma$ where the map $[\![\psi]\!]^X_\sigma : \mathcal{P}(C) \to \mathcal{P}(C)$ is defined by $[\![\psi]\!]^X_\sigma(A) = [\![\psi]\!]_{\sigma[X \mapsto A]}$ for $A \subseteq C$, with $(\sigma[X \mapsto A])(X) = A$ and $(\sigma[X \mapsto A])(Y) = \sigma(Y)$ for $X \neq Y$, and LFP and GFP take least and greatest fixpoints of monotone functions on $\mathcal{P}(C)$, respectively; monotonicity of $[\![\psi]\!]^X_\sigma$ is clearly an invariant of the recursive definition.

The *alternation depth* $\mathsf{ad}(\eta X.\psi)$ of a fixpoint $\eta X.\psi$ is the depth of alternating nesting of such fixpoints in ψ that depend on X; we assign *odd* numbers to least fixpoints and *even* numbers to greatest fixpoints. E.g. for $\psi = \nu X.\phi$ and $\phi = \mu Y.(p \wedge \heartsuit X) \vee \heartsuit Y$, we have $\mathsf{ad}(\psi) = 2$, $\mathsf{ad}(\phi) = 1$. For a detailed definition of alternation depth, see e.g. [22].

▶ **Example 1.** We proceed to see some standard examples of coalgebraic semantics, see [28, 6, 29] for more details.

1. *Relational μ-calculus:* As indicated above, we obtain a version of the standard relational μ-calculus [19] without propositional atoms by taking $T = \mathcal{P}$ and $\Lambda = \{\Box, \Diamond\}$ with predicate liftings as described previously. We can add a set At of propositional atoms $p, q, \ldots$ by regarding them as nullary modalities, and interpret $p \in \mathsf{At}$ over the extended functor $T = \mathcal{P}(\mathsf{At}) \times \mathcal{P}$ by the nullary predicate lifting $[\![p]\!]_X = \{(P, B) \in \mathcal{P}(\mathsf{At}) \times \mathcal{P}(X) \mid p \in P\}$. E.g. the formula $\nu X.\, \mu Y.\, ((p \wedge \Box X) \vee (q \wedge \Box Y))$ says that on every path from the current state, p holds everywhere except possibly on finite segments where q holds. Similarly, we can introduce a set Act of actions, and index modalites over actions $a \in \mathsf{Act}$. We interpret the indexed modalities $\Box_a, \Diamond_a$ over the functor $T = \mathcal{P}^{\mathsf{Act}}$, e.g. the box by $[\![\Box_a]\!]_X(A) = \{f \in \mathcal{P}(X)^{\mathsf{Act}} \mid f(a) \subseteq A\}$. We can similarly add propositional atoms and actions to all examples that follow.

2. *Graded μ-calculus:* The graded μ-calculus [20] has modalities $\langle b\rangle$, $[b]$, indexed over $b \in \mathbb{N}$, read "in more than b successors" and "in all but at most b successors", respectively. These can be interpreted over relational structures but it is technically more convenient to use *multigraphs*, i.e. transition systems with edge weights (*multiplicities*) in $\mathbb{N} \cup \{\infty\}$, which are coalgebras for the multiset functor $\mathcal{B}$. The latter maps a set X to the set $\mathcal{B}(X) = (\mathbb{N} \cup \{\infty\})^X$ of maps $X \to (\mathbb{N} \cup \{\infty\})$; we treat elements $\beta \in \mathcal{B}(X)$ as $(\mathbb{N} \cup \{\infty\})$-valued discrete measures on X, and in particular write $\beta(A) = \sum_{x\in A} \beta(x)$ for $A \subseteq X$. For a map $f : X \to Y$, the map $\mathcal{B}(f) : \mathcal{B}(X) \to \mathcal{B}(Y)$ is then given by $\mathcal{B}(f)(\beta)(y) = \beta(f^{-1}[\{y\}])$. Over $\mathcal{B}$-coalgebras, we interpret $\langle b\rangle$ and $[b]$ by the mutually dual predicate liftings

$$[\![\langle b\rangle]\!]_X(A) = \{\beta \in \mathcal{B}(X) \mid \beta(A) > b\} \quad \text{and} \quad [\![[b]]\!]_X(A) = \{\beta \in \mathcal{B}(X) \mid \beta(X \setminus A) \leq b\}.$$

   E.g. the formula $\nu X.\, (\phi \wedge \Diamond_1 X)$ says that the current state is the root of an infinite tree with branching degree at least 2 (counting multiplicities) on which ϕ holds everywhere.
3. *Probabilistic μ-calculus:* Let $\mathcal{D}$ denote the *discrete distribution functor*, defined on sets by $\mathcal{D}(X) = \{\beta : X \to [0, 1] \mid \sum_{x\in X} \beta(x) = 1\}$. That is, $\mathcal{D}(X)$ is the set of discrete probability distributions on X; coalgebras for $\mathcal{D}$ are just Markov chains. Similarly as for the graded μ-calculus, we take modalities $[p]$, $\langle p\rangle$ indexed over $p \in [0, 1] \cap \mathbb{Q}$, interpreted over $\mathcal{D}$ by $[\![\langle p\rangle]\!]_X(A) = \{\beta \in \mathcal{D}(X) \mid \beta(A) > p\}$ and $[\![[p]]\!]_X(A) = \{\beta \in \mathcal{D}(X) \mid \beta(X \setminus A) \leq p\}$ (using the same measure-theoretic notation as in the previous item). The arising coalgebraic μ-calculus is the *probabilistic μ-calculus* [6, 21].
4. *Monotone μ-calculus:* The *monotone neighbourhood functor* $\mathcal{M}$ maps a set X to the set $\mathcal{M}(X) = \{\mathfrak{A} \in 2^{(2^X)} \mid \mathfrak{A} \text{ upwards closed}\}$ of set systems over X that are upwards closed under subset inclusion (i.e. $A \in \mathfrak{A}$ and $A \subseteq B$ imply $B \in \mathfrak{A}$). Coalgebras for $\mathcal{M}$ are *monotone neighbourhood frames* in the sense of Scott-Montague semantics [5]. We take $\Lambda = \{\Box, \Diamond\}$ and interpret $\Box$ over $\mathcal{M}$ by the predicate lifting

$$[\![\Box]\!]_X(A) = \{\mathfrak{A} \in \mathcal{M}(X) \mid A \in \mathfrak{A}\} = \{\mathfrak{A} \in \mathcal{M}(X) \mid \exists B \in \mathfrak{A}.\, B \subseteq A\},$$

   and $\Diamond$ by the corresponding dual lifting, $[\![\Diamond]\!]_X(A) = \{\mathfrak{A} \in \mathcal{M}(X) \mid (X \setminus A) \notin \mathfrak{A}\} = \{\mathfrak{A} \in \mathcal{M}(X) \mid \forall B \in \mathfrak{A}.\, B \cap A \neq \emptyset\}$. The arising coalgebraic μ-calculus is known as the *monotone μ-calculus* [12]. When we add propositional atoms and actions, and replace $\mathcal{M}$ with its subfunctor $\mathcal{M}_s$ defined by $\mathcal{M}_s(X) = \{\mathfrak{A} \in \mathcal{M}(X) \mid \emptyset \notin \mathfrak{A} \ni X\}$, whose coalgebras are *serial* monotone neighbourhood frames, we arrive at the ambient fixpoint logic of *concurrent dynamic logic* [25] and Parikh's *game logic* [23]. In game logic, actions are understood as atomic games of Angel vs. Demon, and we read $\Box_a\phi$ as "Angel has a strategy to enforce ϕ in game a". Game logic is then mainly concerned with composite games, formed by the control operators of dynamic logic and additional ones; the semantics can be encoded into fixpoint definitions. For instance, the formula $\nu X.\, p \wedge \Box_a X$ says that Angel can enforce p in the composite game where a is played repeatedly, with Demon deciding when to stop.
5. *Alternating-time μ-calculus:* Fix a set $N = \{1, \dots, n\}$ of *agents*. Using alternative notation from *coalition logic* [24], we present the *alternating-time μ-calculus (AMC)* [1] by modalities $[D]$, $\langle D\rangle$ indexed over *coalitions* $D \subseteq N$, read "D can enforce" and "D cannot prevent", respectively. We define a functor $\mathcal{G}$ by

$$\mathcal{G}(X) = \{(k_1, \dots, k_n, f) \mid k_1, \dots, k_n \in \mathbb{N} \setminus \{0\}, f : \big(\textstyle\prod_{i\in N}[k_i]\big) \to X\}$$

   where we write $[k] = \{1, \dots, k\}$. We understand $(k_1, \dots, k_n, f) \in \mathcal{G}(X)$ as a one-step concurrent game with k_i available moves for agent $i \in N$, and outcomes in X determined by the *outcome function* f from a joint choice of moves by all the agents. For $D \subseteq N$,

we write $S_D = \prod_{i\in D}[k_i]$. Given joint choices $s_D \in S_D$, $s_{N\setminus D} \in S_{N\setminus D}$ of moves for D and $N \setminus D$ respectively, we write $(s_D, s_{N\setminus D}) \in s_N$ for the joint move of all agents induced in the evident way. In this notation, we interpret the modalities $[D]$ over $\mathcal{G}$ by the predicate lifting

$$[\![[D]]\!]_X(A) = \{(k_1, \dots, k_n, f) \in \mathcal{G}(X) \mid \exists s_D \in S_D. \forall s_{N\setminus D} \in S_{N\setminus D}. f(s_D, s_{N\setminus D}) \in A\},$$

and the modalities $\langle D\rangle$ by dualization. This captures exactly the semantics of the AMC: $\mathcal{G}$-coalgebras are precisely *concurrent game structures* [1], i.e. assign a one-step concurrent game to each state, and $[D]\phi$ says that the agents in D have a joint move such that whatever the agents in $N \setminus D$ do, the next state will satisfy ϕ. E.g. $\mu X.\, p \vee [D]X$ says that coalition D can eventually enforce that p is satisfied (a property expressible already in alternating-time temporal logic ATL [1]).

We *fix the data* T, Λ *and a predicate lifting* $[\![\heartsuit]\!]$ *for each* $\heartsuit \in \Lambda$ *for the rest of the paper.* We assume given a suitable representation size for the modalities in Λ; this is relevant in particular when Λ is infinite, e.g. for the graded and the probabilistic μ-calculus. We generally assume that numbers are coded in binary, except in the treatment of the graded μ-calculus via one-step satisfaction games in Section 5.

We write $\mathsf{size}(\psi)$ for the ensuing representation size of a formula ψ, counting the representation size for each modality and 1 for other connectives and variables. Similarly, we assume given a representation of elements of $T(X)$ for finite sets X, with associated representation size $\mathsf{size}(s)$ for elements $s \in T(X)$. Again, we generally assume that numbers in s (e.g. in probablity distributions) are encoded in binary, except in Section 5. Moreover, we represent monotone neighbourhood systems $\mathfrak{A}$ by set systems $\mathfrak{A}_0 \subseteq \mathfrak{A}$ that generate $\mathfrak{A}$ by upwards closure; otherwise, representations are essentially obvious.

The *size* $\mathsf{size}(\xi)$ of a finite coalgebra $\xi : C \to T(C)$ is then defined as $\sum_{x\in C}(1+\mathsf{size}(\xi(x)))$. We also fix the input for the model checking algorithm: First, we fix a T-coalgebra (C, ξ) with C finite; second, we fix a closed target formula χ, assuming w.l.o.g. that χ is *clean*, i.e. that every fixpoint variable is bound by at most one fixpoint operator in χ. For a variable $X \in V$ that is bound in χ, we then write $\theta(X)$ to denote *the* formula $\eta X.\psi$ that is a subformula of χ. Let $\mathsf{Cl}(\chi)$ be the *closure* (that is, the set of subformulae) of χ. We have $|\mathsf{Cl}(\chi)| \leq |\chi| \leq |\mathsf{size}(\chi)|$, where $|\chi|$ denotes the number of operators or variables in χ (ignoring representation sizes). We put $k = \max\{\mathsf{ad}(\eta X.\psi) \mid \eta X.\psi \in \mathsf{Cl}(\chi)\}$.

3 Local Model Checking for the Coalgebraic μ-Calculus

We proceed to introduce and analyse our model checking algorithm which essentially is a more general version of the fixpoint iteration algorithm for standard parity games [3]. The algorithm will interface with the functor via the following computational problem:

▶ **Definition 2** (One-step satisfaction problem)**.** Let T be a functor and let C be a set. The *one-step satisfaction problem* for inputs $s \in T(C)$, $\heartsuit \in \Lambda$ and $U \subseteq C$ consists in deciding whether $s \in [\![\heartsuit]\!]U$. We denote the time it takes to solve the problem for input $s, \heartsuit, U$ by $t(\mathsf{size}(s), \mathsf{size}(\heartsuit), |C|)$, having $|U| \leq |C|$.

We discuss the one-step satisfaction problem for some of the logics from Example 1:

▶ **Example 3.**

1. In the relational case, with $T = \mathcal{P}$, $\Lambda = \{\Diamond, \Box\}$, to check one-step satisfaction for $s \in \mathcal{P}(C), \Diamond, U \subseteq C$, we have to check whether s intersects with U; and to check one-step satisfaction for $s, \Box, U$, we have to check whether s is a subset of U. So $t(\mathsf{size}(s), \mathsf{size}(\heartsuit), |C|) \leq |C|$ for $\heartsuit \in \Lambda$, assuming that containment in sets can be checked in constant time.
2. In the graded case, with $T = \mathcal{B}$, $\Lambda = \{\langle b\rangle, [b] \mid b \in \mathbb{N}\}$, to check one-step satisfaction for $s \in \mathcal{B}(C), \langle b\rangle, U \subseteq C$, we have to check whether $\sum_{u\in U} s(u) > b$; to check one-step satisfaction for $s, [b], U$, we have to check whether $\sum_{u\in(C\setminus U)} s(u) \leq b$. Hence $t(\mathsf{size}(s), \mathsf{size}(\heartsuit), |C|) \leq \mathsf{size}(\heartsuit) \cdot |C|$ for $\heartsuit \in \Lambda$; this holds even if grades are coded in binary since binary numbers can be added and compared in linear time.
3. In the probabilistic case, with $T = \mathcal{D}$, $\Lambda = \{\langle p\rangle, [p] \mid p \in \mathbb{Q} \cap [0, 1]\}$, to check one-step satisfaction for $s \in \mathcal{D}(C), \langle p\rangle, U \subseteq C$, we have to check whether $\sum_{u\in U} s(u) > p$; to check one-step satisfaction for $s, [p], U$, we have to check whether $\sum_{u\in(C\setminus U)} s(u) \leq p$. Hence $t(\mathsf{size}(s), \mathsf{size}(\heartsuit), |C|) \in \mathcal{O}((((|C| \cdot \mathsf{size}(s))^2) \cdot |C|)$ for $\heartsuit \in \Lambda$. This holds even if probabilities are coded in binary since quotients of o-bit binary numbers can be added in time $\mathcal{O}(o^2)$; there are at most $|C|$ summation steps, and since each summation step increases the number of bits of the denominator of the summand by at most $\mathsf{size}(s)$, each summation step can be done in time $(|C| \cdot \mathsf{size}(s))^2$.

Since our model checking algorithm can be seen as an algorithm for solving a coalgebraic variant of parity games, we now recall some basic notions of parity games (see e.g. [14]).

▶ **Definition 4** (Parity games). A *parity game* (V, E, α) consists of a set of *nodes* V, a set of *moves* $E \subseteq V \times V$ and a *priority function* $\alpha : V \to \mathbb{N}$. Furthermore, each node belongs to exactly one of the players Eloise or Abelard (where we denote Eloise's nodes by $V_\exists$ and Abelard's nodes by $V_\forall$). A *play* $\rho = v_0, v_1, \ldots \subseteq V^* \cup V^\omega$ is a (finite or infinite) sequence of nodes such that for all $i \geq 0$ such that ρ contains at least $i + 1$ nodes, we have $(v_i, v_{i+1}) \in E$. We say that an infinite play $\rho = v_0, v_1, \ldots$ is *even*, if the largest priority that occurs infinitely often in it is even (formally, if $\max\{\alpha(v) \mid \forall j.\, (v_j = v) \Rightarrow \exists j' > j.\, (v_{j'} = v)$ and $\exists j.\, v_j = v\}$ is an even number), and *odd* otherwise; finite plays are required to end in nodes that have no outgoing move. Player Eloise wins all even plays and finite plays that end in an Abelard-node; player Abelard wins all other plays. The *size* of a parity game (V, E, α) is $|V|$. A *(history-free)* Eloise-*strategy* $s : V_\exists \rightharpoonup V$ is a partial function that assigns moves $s(x)$ to Eloise-nodes $x \in \mathsf{dom}(s)$. A play $\rho = v_0, v_1, \ldots$ *follows* a strategy s if for all $i \geq 0$ such that $v_i \in V_\exists$, $v_{i+1} = s(v_i)$. An Eloise-strategy *wins* a node $v \in V$ if Eloise wins all plays that start at v and follow s. We have a dual notion of Abelard-strategies; *solving* a parity game consists in computing the *winning regions* of the two players, that is, the sets of states that they win.

A crucial property of parity games is that they are *history-free determined* [14], that is, that every node in a parity game is won by exactly one of the two players and then there is a history-free strategy for the respective player that wins the node.

One natural way to solve a parity game is by *fixpoint iteration* [3] (which essentially computes a particular *progress measure annotation* [18] of the game). Algorithms that use this method repeatedly apply a function (denoted by Ψ in Section 3.1 in [3] and sometimes also referred to as *is_mother*, see e.g. Section 4.2 in [20]) that evaluates the allowed moves at each node to compute the set of nodes that are won by player Eloise *under the assumption* that sets of nodes computed by previous iterations of the function are also won by her. Intuitively, the algorithm keeps a set of currently allowed nodes Y_i for each priority i in memory; one iteration of the function *is_mother* then computes the set of nodes *is_mother*$(Y_0, \ldots, Y_k)$

which have some priority j and which either belong to Eloise and have *some* move to a node from Y_j or belong to Abelard and *only* have moves to nodes from Y_j. The repeated application of this function is defined in terms of a nested fixpoint (referred to as Φ_d in [3], Section 3.1). The winning region for Abelard is computed as the dual fixpoint of the complementary function $\overline{is\_mother}$. Our model checking algorithm proceeds in a very similar fashion, but crucially uses instances of the one-step satisfaction problem to evaluate modal nodes. We proceed to define our variants of the *is_mother* and $\overline{is\_mother}$ functions which we call f and g for brevity in this work.

▶ **Definition 5.** For $V = \mathsf{Cl}(\chi) \times C$, we define the *tracing function* $h : V \to \mathcal{P}(V)$ by putting

$$h(\psi, x) = \begin{cases} \emptyset & \text{if } \psi = \bot \text{ or } \psi = \top \\ \{(\psi_1, x), (\psi_2, x)\} & \text{if } \psi = \psi_1 \vee \psi_2 \text{ or } \psi = \psi_1 \wedge \psi_2 \\ \{(\psi_1, x)\} & \text{if } \psi = \eta X.\psi_1 \\ \{(\theta(X), x)\} & \text{if } \psi = X \\ \{(\psi_1, y) \mid y \in C\} & \text{if } \psi = \heartsuit\psi_1 \end{cases}$$

A non-modal formula ψ can be traced to the formula ϕ at x if and only if $(\phi, x) \in h(\psi, x)$, so e.g. a conjunction $\psi_1 \wedge \psi_2$ can be traced to ψ_1 and to ψ_2, all at some state x. A modal formula $\heartsuit\psi_1$ can be traced from x to the formula ψ_1 at any state y (that is, we have $(\psi_1, y) \in h(\heartsuit\psi_1, x)$ for all $y \in C$); for instance, $\Diamond\psi$ can be traced from some state x to ψ at any state $y \in C$. Hence $h(\psi, x)$ computes the set of nodes that are relevant for the (one-step) satisfaction of ψ at x. For a given set $G \subseteq V$, we then define $k+1$-ary functions $f, g : (\mathcal{P}(G))^{k+1} \to \mathcal{P}(G)$ by putting, for $\mathbf{Y} = (Y_0, \dots, Y_k) \in (\mathcal{P}(G))^{k+1}$,

$$\begin{aligned} f(\mathbf{Y}) =& \{(\top, x) \mid (\top, x) \in G\} \cup \{(\heartsuit\psi, x) \in G \mid \xi(x) \in [\![\heartsuit]\!]\{y \mid (\psi, y) \in Y_0\}\} \cup \\ & \{(\psi \vee \phi, x) \in G \mid h(\psi \vee \phi, x) \cap Y_0 \neq \emptyset\} \cup \{(\psi \wedge \phi, x) \in G \mid h(\psi \wedge \phi, x) \subseteq Y_0\} \cup \\ & \{(\eta X.\psi, x) \in G \mid h(\eta X.\psi, x) \subseteq Y_0\} \cup \{(X, x) \mid h(X, x) \subseteq Y_{\mathsf{ad}(\theta(X))}\} \\ g(\mathbf{Y}) =& \{(\bot, x) \mid (\bot, x) \in G\} \cup \{(\heartsuit\psi, x) \in G \mid \xi(x) \notin [\![\heartsuit]\!]\{y \mid (\psi, y) \in (G \setminus Y_0)\}\} \cup \\ & \{(\psi \vee \phi, x) \in G \mid h(\psi \vee \phi, x) \subseteq Y_0\} \cup \{(\psi \wedge \phi, x) \in G \mid h(\psi \wedge \phi, x) \cap Y_0 \neq \emptyset\} \cup \\ & \{(\eta X.\psi, x) \in G \mid h(\eta X.\psi, x) \subseteq Y_0\} \cup \{(X, x) \in G \mid h(X, x) \subseteq Y_{\mathsf{ad}(\theta(X))}\}. \end{aligned}$$

Finally, we put

$$\mathbf{E}_G = \eta_k Y_k. \dots . \eta_1 Y_1. \eta_0 Y_0. f(\mathbf{Y}) \quad \text{and} \quad \mathbf{A}_G = \overline{\eta_k} Y_k. \dots . \overline{\eta_1} Y_1. \overline{\eta_0} Y_0. g(\mathbf{Y}),$$

where η_i is GFP if i is even and LFP otherwise and where $\overline{\mathsf{LFP}} = \mathsf{GFP}$ and $\overline{\mathsf{GFP}} = \mathsf{LFP}$.

For instance, we have $(X, x) \in f(\mathbf{Y})$ if $(\theta(X), x) \in Y_p$ where $p = \mathsf{ad}(\theta(X))$, that is, when passing the fixpoint variable X, priority p occurs in the game. We also have e.g. $(\psi \vee \phi, x) \in f(\mathbf{Y})$ if $(\psi, x) \in Y_0$ or $(\phi, x) \in Y_0$ but $(\psi \vee \phi, x) \in g(\mathbf{Y})$ if $(\psi, x) \in Y_0$ *and* $(\phi, x) \in Y_0$; the latter constraint comes with the intuition that g is used to derive *non*-satisfaction of formulas. Checking whether some pair $(\heartsuit\psi, x)$ is contained in $f(\mathbf{Y})$ or $g(\mathbf{Y})$ is an instance of the one-step satisfaction problem with input $\xi(x)$,$\heartsuit$,$\{y \mid (\psi, y) \in U\}$ for some $U \subseteq V$; since $\mathsf{size}(\xi(x)) \leq \mathsf{size}(C)$, $\mathsf{size}(\heartsuit) \leq \mathsf{size}(\chi)$ and $\{y \mid (\psi, y) \in U\} \subseteq C$, the instance can be solved in time $t(\mathsf{size}(C), \mathsf{size}(\chi), |C|)$.

The sets $\mathbf{E}_G$ and $\mathbf{A}_G$ can be seen as winning regions for the two players Eloise and Abelard in a coalgebraic parity game with set of nodes $G \subseteq V$ that is very similar to a model checking parity game but uses instances of the one-step satisfaction problem instead of modal moves.

Intuitively, Eloise has, for all nodes $(\psi, x) \in \mathbf{E}_G$ and for all disjunctions that are encountered when checking satisfaction of ψ at x, a choice such that the combination of her choices guarantees that no trace of ψ through ξ that starts at x unfolds some least fixpoint infinitely often without also unfolding a surrounding fixpoint infinitely often.

We now introduce our local model checking algorithm, which iteratively adds nodes from V to a growing set of nodes G and computes the sets $\mathbf{E}_G$ and $\mathbf{A}_G$ in optional intermediate model checking steps. The algorithm hence inherently supports *local model checking* [30] since the coalgebraic model checking game is constructed step by step and the partially constructed game can be solved on-the-fly, that is, at any time while building up the game, terminating as soon as one of the players has a strategy that wins the initial node in the game played over G. Since the model sizes in model checking tend to be exponential in the sizes of formulas, on-the-fly solving seems to be a valuable capability as it allows to cut down the search space of the algorithm; a concrete implementation of the algorithm and an evaluation of this aspect remains a task for future work though.

Algorithm 1: Local model checking.

To decide whether $x \in [\![\chi]\!]$, initialize $U = \{(\chi, x)\}$, $G = \emptyset$.

1. Expansion: pick *some* $(\psi, y) \in U$, add it to G and remove it from U, and add those pairs from $h(\psi, y)$ that are not already contained in G to U.
2. (Optional) Check: compute $\mathbf{E}_G$ and/or $\mathbf{A}_G$. If $(\chi, x) \in \mathbf{E}_G$, then return "yes", if $(\chi, x) \in \mathbf{A}_G$, then return "no".
3. Loop: if $U \neq \emptyset$, then continue with step 1).
4. Final Check: compute $\mathbf{E}_G$. If $(\chi, x) \in \mathbf{E}_G$, then return "yes", otherwise return "no".

In the final checking step, all nodes that are reachable from (χ, x) using h have been added to G so that $\mathbf{E}_G = G \setminus \mathbf{A}_G$, i.e. every node – including (χ, x) – is won by exactly one of the players; in the optional intermediate checking steps however, there may be nodes for which no player has a winning strategy yet.

The correctness statement then reads as follows.

▶ **Lemma 6.** *We have $(\chi, x) \in \mathbf{E}_V$ if and only if $x \in [\![\chi]\!]$.*

Proof sketch. Let $(\chi, x) \in \mathbf{E}_V$. We use lexicalically ordered vectors of natural numbers as a measure for the computation of the nested fixpoint $\mathbf{E}_V$ (similar in spirit to coalgebraic progress measures [16]) and then show $x \in [\![\chi]\!]$ by nested induction and coinduction, using the measure to ensure termination of the inductive parts of the proof. For the converse direction, let $x \in [\![\chi]\!]$. Again we use lexicalically ordered vectors of natural numbers as a measure, but this time for the satisfaction of fixpoint formulas in models. The proof of $x \in \mathbf{E}_V$ then is again by nested induction and coinduction, using the latter vectors as termination measure for the inductive parts. ◀

▶ **Lemma 7.** *The algorithm runs in time $\mathcal{O}(|V| \cdot t(\mathsf{size}(C), \mathsf{size}(\chi), |C|) \cdot |V|^{k+1})$.*

Proof. The runtime of the algorithm is dominated by the time it takes to compute $\mathbf{E}_V$. We have to compute a $k+1$-nested fixpoint which can be done by fixpoint iteration, that is, by computing $f(\mathbf{Y})$ for some $\mathbf{Y}$ at most $|V|^{k+1}$ times. A single computation of $f(\mathbf{Y})$ can be implemented to run in time $|V| \cdot t(\mathsf{size}(C), \mathsf{size}(\chi), |C|)$ since we have to solve the one-step satisfaction problem at most for all $(\heartsuit\psi, x) \in V$, that is, at most $|V|$ times. ◀

This yields the following bounds in concrete instances:

▶ **Corollary 8.** *We obtain the following upper time bounds for the model checking problems of the respective μ-calculi:*
1. *for the graded μ-calculus: $\mathcal{O}((|\chi| \cdot |C|^2 \cdot \mathsf{size}(\chi) \cdot (|\chi| \cdot |C|)^{k+1})$;*
2. *for the probabilistic μ-calculus: $\mathcal{O}((|\chi| \cdot |C|^4 \cdot (\mathsf{size}(C))^2 \cdot (|\chi| \cdot |C|)^{k+1})$;*

Proof. Immediate from Lemma 7 by the observations from Example 3. ◀

4 Coalgebraic Model Checking in NP ∩ coNP

As a side result, we now show that if the one-step satisfaction problem of a coalgebraic logic is in P, then the model checking problem of the μ-calculus over this logic is in NP ∩ coNP. We derive this result independently of our local model checking algorithm, with the help of the exponentially sized evaluation games for the coalgebraic μ-calculus from Definition 3.5. in [7], which we briefly recall below; the authors of [7] use a slightly different but equivalent formulation of the games that uses the Fischer-Ladner closure instead of $\mathsf{Cl}(\chi)$.

▶ **Definition 9** (Evaluation games, [7])**.** The *evaluation game* for χ is a parity game $\mathcal{E}_\chi = (W, E, \alpha)$ with set of nodes $W = V \cup (\{\heartsuit\psi \in \mathsf{Cl}(\chi)\} \times \mathcal{P}(C))$, set of moves $E \subseteq W \times W$ and priority function $\alpha : W \to \mathbb{N}$. For nodes $(\psi, x) \in V$ such that ψ is not a modal operator, we put $E(\psi, x) = h(\psi, x)$; for $\heartsuit\psi \in \mathsf{Cl}(\chi)$, we put

$$E(\heartsuit\psi, x) = \{(\heartsuit\psi, U) \mid U \subseteq C, \xi(x) \in [\![\heartsuit]\!](U)\},$$

and for $U \subseteq C$, we put $E(\heartsuit\psi, U) = \{(\psi, y) \mid y \in U\}$. Nodes $(\bot, x)$, $(\psi_1 \vee \psi_2, x)$ and $(\heartsuit\psi, x)$ belong to player Eloise and nodes $(\top, x)$, $(\psi_1 \wedge \psi_2, x)$ and $(\heartsuit\psi, U)$ belong to player Abelard; the ownership of nodes $(\eta X.\, \psi, x)$ and (X, x) is irrelevant since such nodes have exactly one outgoing move. All nodes $(\psi, x) \in V$ such that ψ is not a fixpoint variable and all nodes $(\psi, U) \in \mathsf{Cl}(\chi) \times \mathcal{P}(C)$ have priority 0; nodes $(X, x) \in V$ have priority $\alpha(X, x) = \mathsf{ad}(\theta(X))$.

For modal nodes $(\heartsuit\psi, x)$, Eloise has to pick a set U of states in such a way that $\xi(x)$ is contained in the predicate on $T(C)$ obtained by lifting U with $[\![\heartsuit]\!]$ (that is, $\xi(x) \in [\![\heartsuit]\!](U)$); player Abelard in turn can challenge, for all $y \in U$, whether ψ is satisfied at y, that is, he can move from $(\heartsuit\psi, U)$ to all nodes (y, ψ) with $y \in U$. As we have $V \leq |\chi| \cdot |C|$ and hence $W \leq |\chi|(|C| + 2^{|C|})$, the sizes of evaluation games are exponential in the number of states of the input models.

Under the stated assumptions on the one-step satisfaction problem, our algorithm can be understood as a method to solve these exponential-sized parity games in time p^k where p is a polynomial in the formula size and the model size, by avoiding a full unfolding of the game.

▶ **Lemma 10.** *We have $(\chi, x) \in \mathbf{E}_V$ if and only if* Eloise *wins the node (χ, x) in $\mathcal{E}_\chi$.*

Proof. Directly from Lemma 6 and Theorem 3.6 in [7]. ◀

We now obtain the announced criterion for model checking in NP ∩ coNP:

▶ **Theorem 11.** *If the one-step satisfaction problem for a coalgebraic logic is in* P*, then the model checking problem for the μ-calculus over this logic is in* NP ∩ coNP*.*

Proof sketch. We recall that the coalgebraic μ-calculus is closed under negation so that it suffices to show containment in NP. A history-free Eloise-strategy turns an exponential-sized evaluation game into a polynomial-sized graph since it picks, for each modal node, exactly

one of the exponentially many moves that are available to Eloise. We nondeterministically guess whether the input formula is satisfied at the input state or not; depending on this guess, we then guess a history-free winning strategy in the evaluation game for the input formula or its negation, and verify that it indeed is a winning strategy for the respective game. This verification can be done in polynomial time since the guessed strategy turns the game into a graph of polynomial size and since the one-step satisfaction problem (which has to be solved once for each node in the graph) can be solved in polynomial time by assumption. ◀

5 Small model checking games

As we have recalled in Definition 9, the one-step satisfaction problem can also be encoded in terms of the evalution games from [7]; this particular encoding however involves guessing sets of states and hence leads to games of exponential size. We now introduce the notions of one-step satisfaction arenas and games, which we use as an alternative game-based means to decide one-step satisfaction of modal operators; for certain logics, including the monotone μ-calculus, the graded μ-calculus with grades coded in unary, and the alternating-time μ-calculus, this enables the construction of polynomial-size parity games for model checking, which can be fed directly to parity game solvers and profit automatically from advances in parity game solving.

▶ **Definition 12** (One-step satisfaction arenas)**.** Recall that T is a functor and Λ a set of modal operators. Let C be a set. A *one-step satisfaction arena* $\mathsf{A}_{\heartsuit,t}$ for $\heartsuit \in \Lambda$ and $t \in T(C)$ consists of a set $V_{\heartsuit,t}$ of nodes that is made up of

- the *initial node* $(\heartsuit, t)$,
- a set $I_{\heartsuit,t}$ of *inner nodes*,
- the set C of *exit nodes*,

and an *acyclic* set $E_{\heartsuit,t} \subseteq V_{\heartsuit,t} \times V_{\heartsuit,t}$ of moves such that $E_{\heartsuit,t}(x) = \emptyset$ for all exit nodes $x \in C$. Additionally, the initial and the inner nodes belong to exactly one of the players Eloise or Abelard.

▶ **Example 13.**

1. For $T = \mathcal{P}$, one-step satisfaction arenas have depth one and hence do not have inner nodes. The initial node $(\Diamond, t)$ belongs to Eloise and she can move to any state $y \in t$; formally, we put $E_{\Diamond,t}(\Diamond, t) = \{y \mid y \in t\}$.
2. For the (serial) monotone μ-calculus, one-step satisfaction arenas have depth two. As set of inner nodes, we choose $I_{\Diamond,t} = t$; the initial node $(\Diamond, t)$ belongs to player Eloise and all nodes from $I_{\Diamond,t}$ belong to player Abelard. The moves are defined by putting $E_{\Diamond,t}(\Diamond, t) = \{A \mid A \in t\}$ and $E_{\Diamond,t}(A) = \{y \mid y \in A\}$. The one-step satisfaction arenas thus have $|I_{\Diamond,t}| = |t| \in \mathcal{O}(\mathsf{size}(C)) \subseteq 2^{\mathcal{O}(|C|)}$ inner nodes.
3. For graded logic, the one-step satisfaction arena for $\langle b \rangle$ and $t \in \mathcal{B}(C)$ has the set $I_{\langle b \rangle,t} = \{1, \dots, |C| + 1\} \times \{0, \dots, b\} \times \{0, 1\}$ as inner nodes and there is a referee move from the initial node $(\langle b \rangle, t)$ to $(1, 0, 0) \in I_{\langle b \rangle,t}$. We assume a linear ordering on C and let v_n denote the n-th element in the according sequence of states. Nodes $(n, c, 0)$ belong to player Eloise and nodes $(n, c, 1)$ to player Abelard. We have moves

$$E_{\langle b \rangle,t}(n, c, 0) = \{(n, \min(b + 1, c + t(v_n)), 1), (n + 1, c, 0)\}$$
$$E_{\langle b \rangle,t}(n, c, 1) = \{v_n, (n + 1, c, 0)\},$$

where we assume $n \leq |C|$ in the first clause. Nodes $(|C| + 1, c, 0)$ have no successors and they belong to player Abelard if $c > b$ and to player Eloise if $c \leq b$. Note how we stop counting when the counter c reaches $b + 1$ by taking $\min(b + 1, c + t(v_n))$ as new counter.

The one-step satisfaction arena for $(\langle b\rangle, t)$ contains $|I_{\langle b\rangle,t}| = 2(|C|+1)(b+1) \in \mathcal{O}(|C|\cdot b)$ inner nodes. The estimate on $|I_{\langle b\rangle,t}|$ is thus linear in the size of $\langle b\rangle$ if grades are coded in unary, and exponential if grades are coded in binary.

4. For probabilistic logic, we proceed analogously to the graded case; however, the obtained one-step arenas are of exponential size, even when probabilities are coded in unary. We define the one-step satisfaction arena for $\langle p\rangle$ and t to have the set $I_{\langle p\rangle,t} = \{1,\dots,|C|+1\} \times P_t \times \{0,1\}$ as inner nodes, where $P_t = \{q \in \mathbb{Q} \mid q \leq p, \exists U \subseteq C.\ \sum_{u\in U} t(u) = q\} \cup \{*\}$ (that is, P_t is the set of probabilities $q \leq p$ that can be encountered when summing up any combination of probabilities that t assigns to states). There is a referee move from the initial node $(\langle p\rangle, t)$ to $(1,0,0) \in I_{\langle p\rangle,t}$. Again, we assume a linear ordering on C and let v_n denote the n-th element in the according sequence of states. Nodes $(n,c,0)$ belong to player Eloise and nodes $(n,c,1)$ to player Abelard. We have moves

$$E_{\langle p\rangle,t}(n,c,0) = \{(n, c\oplus t(v_n), 1), (n+1,c,0)\} \qquad E_{\langle p\rangle,t}(n,c,1) = \{v_n, (n+1,c,0)\},$$

where we assume $n \leq |C|$ in the first clause. Here, $c \oplus t(v_n) = c + t(v_n)$ if $c \neq *$ and $c + t(v_n) \leq p$ and $c \oplus t(v_n) = *$ if $c = *$ or $c + t(v_n) > p$, that is, we stop counting once a probability greater than p has been reached. Nodes $(|C|+1, c, 0)$ have no successors and they belong to player Abelard if $c = *$ and to player Eloise if $c \neq *$. The one-step satisfaction arena for $(\langle p\rangle, t)$ contains $|I_{\langle p\rangle,t}| = 2(|C|+1)|P_t| \in \mathcal{O}(|C|\cdot|P_t|)$ inner nodes; we have $|P_t| \leq 2^{|C|}+1$.

5. For alternating-time logic, let $N = \{1,\dots,n\}$ be the set of agents. For the one-step satisfaction arena for $([D], t)$ with $D \subseteq N$, $t = (k_1,\dots,k_n,f) \in \mathcal{G}(C)$, and $f : (\prod_{i\in N}[k_i]) \to C$, we put $I_{[D],t} = S_D$. The arena has the initial node $([D],t)$ that belongs to player Eloise while all inner nodes $s_D \in S_D$ belong to player Abelard. The moves are defined by

$$E_{[D],t}([D],t) = \{s_D \mid s_D \in S_D\} \qquad E_{[D],t}(s_D) = \{f(s_D, s_{N\setminus D}) \mid s_{N\setminus D} \in S_{N\setminus D}\}.$$

The one-step satisfaction arena for $([D],t)$ has $|I_{[D],t}| = |S_D| \in \mathcal{O}(\mathsf{size}(C))$ inner nodes.

6. Also, we can always take the general one-step arena with initial node $(\heartsuit, t)$, belonging to Eloise, set of inner nodes $I_{\heartsuit,t} = \{U \subseteq C \mid t \in [\![\heartsuit]\!](U)\}$ (having $|I_{\heartsuit,t}| \leq 2^{|C|}$), all belonging to Abelard, and with moves $E_{\heartsuit,t}(\heartsuit,t) = \{U \subseteq C \mid t \in [\![\heartsuit]\!](U)\}$ and $E_{\heartsuit,t}(U) = \{y \mid y \in U\}$.

In all examples, except the last one, the one-step satisfaction arenas for dual modal operators are obtained by simply switching the ownership of nodes.

▶ **Definition 14** (One-step games)**.** A *one-step game* $(\mathsf{A}_{\heartsuit,t}, U)$ consists of a one-step satisfaction arena $\mathsf{A}_{\heartsuit,t}$ for $\heartsuit$ and t together with a set $U \subseteq C$ of states, encoding a winning condition; player Eloise *wins* the game $(\mathsf{A}_{\heartsuit,t}, U)$ if she has a strategy s such that all plays in $\mathsf{A}_{\heartsuit,t}$ that start at $(\heartsuit,t)$ and that are played according to s end in exit nodes $c \in U$ or in inner nodes that belong to Abelard. A one-step satisfaction arena $\mathsf{A}_{\heartsuit,t}$ for $\heartsuit$ and t is *one-step sound and complete* if for all $U \subseteq C$, we have $t \in [\![\heartsuit]\!]U$ if and only if Eloise wins the one-step game $(\mathsf{A}_{\heartsuit,t}, U)$.

▶ **Example 15.** The one-step satisfaction arenas from Example 13 all are one-step sound and complete:

1. In the relational case, let $t \in \mathcal{P}(C)$ and $U \subseteq C$. We have $t \in [\![\Diamond]\!]U$ if and only if $t \cap U \neq \emptyset$ which in turn is the case if and only if there is some $y \in t$ such that $y \in U$. Since Eloise can move from $(\Diamond, t)$ to y if and only if $y \in t$, the last statement is true if and only if Eloise wins the game $(\mathsf{A}_{\Diamond,t}, U)$.

2. For the (serial) mononotone μ-calculus, let $t \in \mathcal{P}(\mathcal{P}(C))$ and $U \subseteq C$. We have $t \in [\![\Diamond]\!]U$ if and only if there is some $A \in t$ such that $A \subseteq U$. The latter is the case if and only if there is some $A \in t$ (so that Eloise can move from $(\Diamond, t)$ to A) such that for all $y \in A$, we have $y \in U$ and hence an according Abelard-move from B to the exit node y. The move from $(\Diamond, t)$ to A thus constitutes a winning strategy for Eloise.
3. For graded logic, assume input $\langle b \rangle$ where $b \in \mathbb{N}$ and $t \in \mathcal{B}(C)$ and let $U \subseteq C$. We have $t \in [\![\langle b \rangle]\!]U$ if and only if $\sum_{a \in U} t(a) > b$. So let $\sum_{a \in U} t(a) > b$. The function s defined by $s(n, c, 0) = (n, \min(b+1, c + t(v_n), 1)$ if $v_n \in U$ and $s(n, c, 0) = (n+1, c, 0)$ if $v_n \notin U$ is a strategy that ensures that all plays of the game end at exit nodes $y \in U$ or at the inner node $s(|C|+1, \sum_{a \in U} t(a), 0)$ which belongs to Abelard. The former is the case since s uses exactly nodes $v_n \in U$ to increase the counter so that Abelard can only reach exit nodes from U. The latter is the case since there is just one play that reaches an ending node $s(|C|+1, c, 0)$ for some c and during this play, s moves in such a way that the counter sums up the $t(a)$ for all $a \in U$. For the converse direction, let Eloise have a winning strategy in the game $(\mathsf{A}_{\langle b \rangle, t}, U)$. Using this strategy, Eloise only uses states $v_n \in U$ to increase the counter; there is just a single final inner node of the shape $(|C|+1, c, 0)$ to which the strategy can lead and this node belongs to Abelard by assumption so that we have $c > b$. There is some set $A \subseteq U$ such that if the play that leads to this final node, the counter sums up all $t(a)$ for $a \in A$. Thus we have we have $c = t(A) \leq t(U)$ and hence $t(U) > b$, as required.
4. For probabilistic logic, the proof is analogous to the proof for graded logic.
5. For alternating-time logic, assume input $[D]$ where $D \subseteq \{1, \dots, n\}$ and $t \in \mathcal{G}(C)$ where $t = (k_1, \dots, k_n, f)$ and let $U \subseteq C$. We have $t \in [\![[D]]\!]U$ if and only if there is some $s_D \in S_D$ such that for all $s_{N \setminus D} \in S_{N \setminus D}$, we have $f(s_D, s_{N \setminus D}) \in U$. So assume that the latter is the case. The move from $([D], t)$ to s_D constitutes a winning strategy for Eloise since for all Abelard-moves from s_D to some exit node $f(s_D, s_{N \setminus D})$, we have $f(s_D, s_{N \setminus D}) \in U$ by assumption. For the converse direction, let there be a winning strategy for Eloise in the game $(\mathsf{A}_{[D], t}, U)$ that moves from $([D], t)$ to some s_D such that for all Abelard-moves from s_D to some $f(s_D, s_{N \setminus D})$, we have $f(s_D, s_{N \setminus D}) \in U$. Then we have $\exists s_D \in S_D . \forall s_{N \setminus D} \in S_{N \setminus D} . f(s_D, s_{N \setminus D}) \in U$.
6. The general one-step arenas from Example 13.6 are easily seen to be one-step sound and complete. However, when using these one-step arenas, the model checking games from Definition 16 below are essentially just the evaluation games recalled in Definition 9.

In the following, we assume that one-step sound and complete one-step satisfaction arenas are available for all $\heartsuit$ and t and that the sets of inner nodes of all these arenas are disjoint. We continue to define our modular model checking game by plugging the one-step satisfaction arenas into the modal steps of a standard parity model checking game:

▶ **Definition 16** (Coalgebraic model checking games)**.** The *model checking game* $\mathcal{G}_\chi = (W, E, \alpha)$ for χ is a parity game that is played over the set of nodes

$$V' = V \cup \bigcup_{\heartsuit\psi \in \mathsf{Cl}(\chi), x \in C} (\{\psi\} \times V_{\heartsuit, \xi(x)}).$$

The set of moves $E \subseteq W \times W$ is defined by putting $E(\psi, x) = h(\psi, x)$ if $(\psi, x) \in V$ and ψ is not a modal operator; for $(\heartsuit\psi, x) \in V$, we put $E(\heartsuit\psi, x) = \{(\psi, (\heartsuit, \xi(x)))\}$, that is, there is a referee move from such nodes to the initial node of the one-step arena for $\heartsuit$ and $\xi(x)$, paired with the formula ψ. For nodes $(\psi, v) \in \{\psi\} \times V_{\heartsuit, \xi(x)}$ such that $v \in V_{\heartsuit, \xi(x)}$ is not an exit node, we put $E(\psi, v) = \{\psi\} \times E_{\heartsuit, \xi(x)}(v)$, that is, at such nodes, the moves are inherited from

the arena $\mathsf{A}_{\heartsuit,\xi(x)}$ and take place only in the second component. Nodes $(\top, x)$ and $(\psi_1 \wedge \psi_2, x)$ belong to player Abelard, nodes $(\bot, x)$ and $(\psi_1 \vee \psi_2, x)$ belong to player Eloise. The ownership of nodes $(\eta X.\psi, x)$, (X, x) and $(\heartsuit\psi, x)$ is irrelevant. Nodes $(\psi, v) \in \{\psi\} \times V_{\heartsuit,\xi(x)}$ such that $v \in V_{\heartsuit,\xi(x)}$ is not an exit node are owned by the player that owns v in the arena $\mathsf{A}_{\heartsuit,\xi(x)}$. All nodes, including the nodes from the one-step satisfaction arenas, have priority 0, with the exception of nodes $(X, x) \in V$ which have priority $\alpha(X, x) = \mathsf{ad}(\theta(X))$.

The game $\mathcal{G}_\chi$ is a parity game with k priorities and $|V| + \sum_{\heartsuit\psi \in \mathsf{Cl}(\chi), x \in C} |I_{\heartsuit,\xi(x)}|$ nodes.

▶ **Theorem 17.** *We have $x \in [\![\chi]\!]$ if and only if* Eloise *wins the node (χ, x) in $\mathcal{G}_\chi$.*

Proof. We have that Eloise wins the node (χ, x) in $\mathcal{G}_\chi$ if and only if Eloise wins the node (χ, x) in the evaluation game (recalling Definition 9): the only difference between $\mathcal{G}_\chi$ and the evaluation game for χ is the modal step. If Eloise wins a node $(\heartsuit\psi, x)$ in the evaluation game, then there is a set U such that $\xi(x) \in [\![\heartsuit]\!](U)$ and Eloise wins all nodes (ψ, y) such that $y \in U$. It suffices to show that Eloise can win the one-step satisfaction game $(\mathsf{A}_{\heartsuit,\xi(x)}, U)$; this however is the case since the one-step arena $\mathsf{A}_{\heartsuit,\xi(x)}$ for $\heartsuit$ and $\xi(x)$ is one-step complete. Conversely, let Eloise win a node $(\heartsuit\psi, x)$ in $\mathcal{G}_\chi$. Then there is some set $U \subseteq C$ of exit nodes such that Eloise wins all nodes from $\{\psi\} \times U$ as well as the one-step game $(\mathsf{A}_{\heartsuit,\xi(x)}, U)$. By one-step soundness of the one-step arena $\mathsf{A}_{\heartsuit,\xi(x)}$, we have $\xi(x) \in [\![\heartsuit]\!](U)$ so that in the evaluation game, Eloise can move from $(\heartsuit\psi, x)$ to $(\heartsuit\psi, U)$ and for each Abelard-move from $(\heartsuit\psi, U)$ to (ψ, y) such that $y \in U$, Eloise wins (ψ, y). Lemmas 10 and 6 – or alternatively, Theorem 3.6 in [7] – finish the proof. ◀

Using parity games, we can decide the model checking problem in quasi-polynomial time if the one-step satisfaction arenas are of at most quasi-polynomial size.

▶ **Corollary 18.** *Let $q(\mathsf{size}(\chi), \mathsf{size}(C))$ denote the maximal number of inner nodes of the one-step satisfaction arenas that are used when constructing the model checking game for χ and C. Model checking C against χ can be done in time $\mathcal{O}(((|\chi| \cdot |C|) \cdot q(\mathsf{size}(\chi), \mathsf{size}(C)))^{\log(k)+6})$.*

Proof. The model checking game $\mathcal{G}_\chi$ consists of at most $|\chi| \cdot |C|$ one-step satisfaction games that are plugged into one parity game of size $|\chi| \cdot |C|$, resulting in a parity game of size $\mathcal{O}((|\chi| \cdot |C|) \cdot q(\mathsf{size}(\chi), \mathsf{size}(C)))$ and with k priorities. Using the methods from [4], this game can be solved in time $\mathcal{O}(((|\chi| \cdot |C|) \cdot q(\mathsf{size}(\chi), \mathsf{size}(C)))^{\log(k)+6})$. ◀

▶ **Corollary 19.** *We obtain the following quasi-polynomial upper time bounds for the model checking problems of the respective μ-calculi:*

1. *for the relational μ-calculus: $\mathcal{O}((|\chi| \cdot |C|)^{\log(k)+6})$;*
2. *for the (serial) monotone μ-calculus: $\mathcal{O}((|\chi| \cdot |C| \cdot \mathsf{size}(C))^{\log(k)+6})$;*
3. *for the graded μ-calculus with grades coded in unary: $\mathcal{O}((|\chi| \cdot |C|^2 \cdot \mathsf{size}(\chi))^{\log(k)+6})$;*
4. *for the alternating-time μ-calculus: $\mathcal{O}((|\chi| \cdot |C| \cdot \mathsf{size}(C))^{\log(k)+6})$.*

Proof. Immediate from Corollary 18 by the observations from Example 13. ◀

6 Conclusion

We have presented an algorithm to solve the model checking problem for coalgebraic μ-calculi in time p^k, where p is a polynomial in the input formula size and the input model size and where k is the alternation depth of the input formula; while the algorithm is correct for all instances of the coalgebraic μ-calculus, the runtime analysis relies on the assumption that the one-step satisfaction problem of the base logic is in P. This holds in many instance

logics, including the graded μ-calculus and the probabilistic μ-calculus, even if grades and probabilities are coded in binary. We also have shown that under the same assumption on the one-step satisfaction problem, the model checking problem of a coalgebraic μ-calculus is in $\mathrm{NP} \cap \mathrm{coNP}$. Our approach relies centrally on a coalgebraic variant of parity games, whose algorithmics we will develop further in future research. For certain logics, including the graded μ-calculus with grades coded in unary, the alternating-time μ-calculus and the monotone μ-calculus, we have shown how to construct model checking games as *standard* parity games of polynomial size; this enables the efficient use of standard parity game solvers in model checking, and also transfers any future progress on solving parity games directly to model checking for such logics.

References

1 Rajeev Alur, Thomas Henzinger, and Orna Kupferman. Alternating-time temporal logic. *J. ACM*, 49:672–713, 2002.

2 Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008.

3 Florian Bruse, Michael Falk, and Martin Lange. The Fixpoint-Iteration Algorithm for Parity Games. In *Games, Automata, Logics and Formal Verification, GandALF 2014*, volume 161 of *EPTCS*, pages 116–130, 2014.

4 Cristian Calude, Sanjay Jain, Bakhadyr Khoussainov, Wei Li, and Frank Stephan. Deciding parity games in quasipolynomial time. In *Theory of Computing, STOC 2017*, pages 252–263. ACM, 2017.

5 Brian F. Chellas. *Modal Logic*. Cambridge University Press, 1980.

6 Corina Cîrstea, Clemens Kupke, and Dirk Pattinson. EXPTIME tableaux for the coalgebraic μ-calculus. In *Computer Science Logic, CSL 2009*, volume 5771 of *LNCS*, pages 179–193. Springer, 2009.

7 Corina Cîrstea, Clemens Kupke, and Dirk Pattinson. EXPTIME tableaux for the coalgebraic μ-calculus. *Log. Meth. Comput. Sci.*, 7, 2011.

8 Corina Cîrstea, Alexander Kurz, Dirk Pattinson, Lutz Schröder, and Yde Venema. Modal logics are coalgebraic. *Comput. J.*, 54:31–41, 2011.

9 Corina Cîrstea, Shunsuke Shimizu, and Ichiro Hasuo. Parity Automata for Quantitative Linear Time Logics. In *Algebra and Coalgebra in Computer Science, CALCO 2017*, volume 72 of *LIPIcs*, pages 7:1–7:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. URL: `http://www.dagstuhl.de/dagpub/978-3-95977-033-0`.

10 Giovanna D'Agostino and Albert Visser. Finality regained: A coalgebraic study of Scott-sets and multisets. *Arch. Math. Logic*, 41:267–298, 2002.

11 E. Allen Emerson, Charanjit Jutla, and A. Prasad Sistla. On model checking for the μ-calculus and its fragments. *Theor. Comput. Sci.*, 258:491–522, 2001. URL: `http://dx.nodoi.org/10.1016/S0304-3975(00)00034-7`.

12 Sebastian Enqvist, Fatemeh Seifan, and Yde Venema. Monadic Second-Order Logic and Bisimulation Invariance for Coalgebras. In *Logic in Computer Science, LICS 2015*. IEEE, 2015.

13 Alessandro Ferrante, Aniello Murano, and Mimmo Parente. Enriched μ-Calculi Module Checking. *Log. Methods Comput. Sci.*, 4(3), 2008.

14 Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research*, volume 2500 of *LNCS*. Springer, 2002. `doi:10.1007/3-540-36387-4`.

15 Helle Hvid Hansen, Clemens Kupke, Johannes Marti, and Yde Venema. Parity Games and Automata for Game Logic. In *Dynamic Logic. New Trends and Applications, DALI 2017*, volume 10669 of *LNCS*, pages 115–132. Springer, 2018.

16 Ichiro Hasuo, Shunsuke Shimizu, and Corina Cîrstea. Lattice-theoretic Progress Measures and Coalgebraic Model Checking. In *Principles of Programming Languages, POPL 2016*, pages 718–732. ACM, 2016.

17 Daniel Hausmann and Lutz Schröder. Optimal Satisfiability Checking for Arithmetic μ-Calculi. In *Foundations of Software Science and Computation Structures, FOSSACS 2019*, volume 11425 of *LNCS*, pages 277–294. Springer, 2019.

18 Marcin Jurdziński. Small Progress Measures for Solving Parity Games. In Horst Reichel and Sophie Tison, editors, *Symposium on Theoretical Aspects of Computer Science, STACS 2000*, pages 290–301, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

19 Dexter Kozen. Results on the propositional μ-calculus. *Theor. Comput. Sci.*, 27:333–354, 1983.

20 Orna Kupferman, Ulrike Sattler, and Moshe Vardi. The Complexity of the Graded μ-Calculus. In *Automated Deduction, CADE 02*, volume 2392 of *LNCS*, pages 423–437. Springer, 2002.

21 Wanwei Liu, Lei Song, Ji Wang, and Lijun Zhang. A Simple Probabilistic Extension of Modal Mu-calculus. In *International Joint Conference on Artificial Intelligence, IJCAI 2015*, pages 882–888. AAAI Press, 2015.

22 Damian Niwinski. On Fixed-Point Clones (Extended Abstract). In *Automata, Languages and Programming, ICALP 1986*, volume 226 of *LNCS*, pages 464–473. Springer, 1986.

23 Rohit Parikh. The logic of games and its applications. *Ann. Discr. Math.*, 24:111–140, 1985.

24 Marc Pauly. A Modal Logic for Coalitional Power in Games. *J. Logic Comput.*, 12:149–166, 2002.

25 David Peleg. Concurrent dynamic logic. *J. ACM*, 34:450–479, 1987. URL: `http://nodoi.acm.org/10.1145/23005.23008`.

26 Jan Rutten. Universal Coalgebra: A Theory of Systems. *Theor. Comput. Sci.*, 249:3–80, 2000.

27 Philippe Schnoebelen. The Complexity of Temporal Logic Model Checking. In *Advances in Modal Logic, AiML 2002*, pages 393–436. College Publications, 2003.

28 Lutz Schröder and Dirk Pattinson. Strong completeness of coalgebraic modal logics. In *Theoretical Aspects of Computer Science, STACS 09*, pages 673–684. Schloss Dagstuhl – Leibniz-Zentrum für Informatik; Dagstuhl, Germany, 2009.

29 Lutz Schröder and Yde Venema. Completeness of Flat Coalgebraic Fixpoint Logics. *ACM Trans. Comput. Log.*, 19(1):4:1–4:34, 2018.

30 Colin Stirling and David Walker. Local model checking in the modal mu-calculus. *Theoretical Computer Science*, 89(1):161–177, 1991.