COOL 2 – A Generic Reasoner for Modal Fixpoint Logics (System Description)

Oliver Görlitz[1], Daniel Hausmann[0000−0002−0935−8602]2 ⋆, Merlin Humml[0000−0002−2251−8519]1 ⋆⋆, Dirk Pattinson[0000−0002−5832−6666]3, Simon Prucker[0009−0000−2317−5565]1, and Lutz Schröder[0000−0002−3146−5906]1 ⋆⋆⋆

[1] Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany
[2] Gothenburg University, Sweden
[3] Australian National University, Canberra, Australia

Abstract. There is a wide range of modal logics whose semantics goes beyond relational structures, and instead involves, e.g., probabilities, multi-player games, weights, or neighbourhood structures. Coalgebraic logic serves as a unifying semantic and algorithmic framework for such logics. It provides uniform reasoning algorithms that are easily instantiated to particular, concretely given logics. The *COOL 2* reasoner provides an implementation of such generic algorithms for coalgebraic modal fixpoint logics. As concrete instances, we obtain in particular reasoners for the aconjunctive and alternation-free fragments of the graded μ-calculus and the alternating-time μ-calculus. We evaluate the tool on standard benchmark sets for fixpoint-free graded modal logic and alternating-time temporal logic (ATL), as well as on a dedicated set of benchmarks for the graded μ-calculus.

1 Introduction

Modal and temporal logics are established tools in the specification and verification of systems. While many such logics are interpreted over relational transition systems, the semantics of quite a number of important logics goes beyond the relational setup, involving, for instance, probabilities [20,30], concurrent games as in alternating-time logics [1,37], monotone neighbourhoods structures as in game logic [35] and concurrent dynamic logic [38], or integer transition weights as in the multigraph semantics [5] of the graded μ-calculus [25]. *Coalgebraic logic* [4] provides a uniform semantic and algorithmic framework for these logics, based on the paradigm of *universal coalgebra* [40]. It provides reasoning algorithms

⋆ Supported by the ERC Consolidator grant D-SynMA (No. 772459)
⋆⋆ Supported by Deutsche Forschungsgemeinschaft (DFG) as part of the Research Training Group 2475 (grant number 393541319/GRK2475/1-2019) and the project 'RAND' (grant number 377333057).
⋆⋆⋆ Supported by Deutsche Forschungsgemeinschaft (DFG) under project MI 717/7-1

of optimal complexity at various levels of expressiveness, up to the coalgebraic μ-calculus [3, 21–23]. These algorithms are parametric in the transition type of systems (weighted, probabilistic, game-based etc.) as well as in suitable choices of modalities specific to the given system type. Their instantiation to specific logics requires providing either a set of next-step modal tableau rules satisfying a suitable completeness criterion [45] or, more generally, a plug-in algorithm that determines satisfiability for an extremely simple *one-step logic* that describes the interaction between modalities, and consists of (conjunctions of) modal operators applied to variables only [29].

The *COalgebraic Ontology Logic solver (COOL)* provides reasoning support for coalgebraic logics based on these generic algorithms. The first version of the tool [15] provided reasoning support for fixpoint-free coalgebraic hybrid logic with global assumptions, using a global caching principle [13]. In the present paper, we present *COOL 2*, which provides reasoning support for coalgebraic fixpoint logics, specifically for both the aconjunctive fragment and the alternation-free fragment of the coalgebraic μ-calculus. By instantiation, we obtain in particular the first implemented reasoners for the graded μ-calculus [26] (for which a set of coalgebraic modal tableau rules has been described in the literature [45]; however, this rule set has later turned out to be incomplete, cf. Remark 2.3) and the alternating-time μ-calculus [1]. We describe the structure of the tool including implementational details, and present evaluation results, focusing on the graded μ-calculus and alternating-time temporal logic (ATL). Additional details on the evaluation can be found in the appendix.

Related Work: We have already mentioned work in coalgebraic logic on which COOL is based [3,13,21–23,45]. COOL is conceptually a successor of the *Coalgebraic Logic Satisfiability Solver (CoLoSS)* [2] but does not share any of its code. CoLoSS implements fixpoint-free logics, and is entirely unoptimised. The first version of COOL [15] has been evaluated on fixpoint-free next-step logics.

COOL does cover also various relational modal logics, for which there are numerous specialised reasoners, including highly optimised description logic reasoners such as FaCT++ [49], Pellet [47], RACER [18], and HermiT [12]. As these systems do not support fixpoint logics, a comparison would be of limited value. In previous work, COOL has been evaluated on various relational fixpoint logics, and has been shown to perform favourably on Computation Tree Logic [23] (in comparison to reasoners featured in a previous systematic evaluation [14]), as well as on the aconjunctive fragment of the modal μ-calculus [22] (in comparison to MLSolver [11]). A reasoner for (next-step) graded modal logic has been evaluated against various description logic reasoners [48], using however the above-mentioned incomplete set of modal tableau rules.

For the same reasons, we refrain from evaluating COOL 2 against reasoners for coalition logic, i.e. the fixpoint-free fragment of the alternating-time μ-calculus, such as CLProver [33]. The only implemented reasoner for any fragment of the alternating-time μ-calculus that does include fixpoints still appears to be the tableau reasoner TATL for alternating-time temporal logic [6, 7]. TATL has been compared to COOL on random formulas in previous work [23].

2 Satisfiability in the Coalgebraic μ-Calculus

COOL 2 is a satisfiability checker for the coalgebraic μ-calculus [3], that is, for the extension of coalgebraic modal logic with extremal fixpoint operators. Formulas of this logic are interpreted over coalgebras, where the semantics of modal operators is defined by means of so-called *predicate liftings* [45]; we recapitulate examples of system types and modalities subsumed by this paradigm in Example 2.1.

Syntax: Formulas are built relative to a set Var of fixpoint variables and a *modal similarity type* Λ, that is, a set of modal operators with assigned finite arities that is closed under duals, with $\overline{\heartsuit} \in \Lambda$ denoting the dual of $\heartsuit \in \Lambda$. Formulas $\psi, \phi, \ldots$ of the *coalgebraic μ-calculus* over Λ are given by the grammar

$$\psi, \phi := \bot \mid \top \mid \psi \wedge \phi \mid \psi \vee \phi \mid \heartsuit(\psi_1, \ldots, \psi_n) \mid X \mid \mu X.\, \psi \mid \nu X.\, \psi,$$

where $\heartsuit \in \Lambda$ has arity n and $X \in \mathsf{Var}$. A formula χ is *aconjunctive* if for every conjunction $\psi \wedge \phi$ that is a subformula of χ, at most one of the formulas ψ and ϕ contains a free fixpoint variable X that is bound by a least fixpoint operator μX. While the logic does not contain negation as an explicit operator, full negation can be defined as usual; e.g. we have $\neg\heartsuit\psi = \overline{\heartsuit}\neg\psi$ and $\neg\mu X.\, \psi = \nu X.\, \neg\psi[\neg X/X]$, using $\neg\neg X = X$.

Both the theoretical satisfiability checking algorithm and its implementation in COOL 2 operate on the *Fischer-Ladner closure* [21, 24, 27] of the target formula. The *alternation depth* (e.g. [21, 29, 34]) of a formula is the maximum depth of dependent alternating nestings of least and greatest fixpoints within the formula. Formulas with alternation depth 1 are *alternation-free*.

Semantics: Formulas are interpreted over F-coalgebras, that is, structures

$$(C, \xi : C \to FC),$$

where $F\colon \mathsf{Set} \to \mathsf{Set}$ is a functor determining the branching type of the systems at hand; thus $\xi(x) \in FC$ encodes the transitions from $x \in C$, structured according to F. Modalities $\heartsuit \in \Lambda$ of arity n are interpreted as *predicate liftings*, that is, families of maps $[\![\heartsuit]\!]_U\colon (2^U)^n \to 2^{FU}$ (for $U \in \mathsf{Set}$) that assign predicates on FU to n-tuples of predicates on U, subject to a *naturality* condition [36, 43]. On a coalgebra (C, ξ), the semantics of formulas is defined inductively in the usual way for the propositional operators and fixpoints, and by $[\![\heartsuit(\psi_1, \ldots, \psi_n)]\!] = \xi^{-1}[[\![\heartsuit]\!]_C([\![\psi_1]\!], \ldots, [\![\psi_n]\!])]$ for modalities.

A closed formula ψ is *satisfiable* if there is a coalgebra (C, ξ) and a state $x \in C$ such that $x \in [\![\psi]\!]$. A formula ψ is *valid* if $\neg\psi$ is not satisfiable.

Example 2.1. (1) The standard *modal μ-calculus* [24] is obtained using the functor $F = \mathcal{P}(A) \times \mathcal{P}$, where A is a fixed set of atoms, the similarity type $\Lambda = \{\Diamond, \Box, a, \neg a \mid a \in A\}$, and predicate liftings

$$[\![\Diamond]\!]_C(B) = \{(A, Z) \in 2^A \times 2^C \mid Z \cap B \neq \emptyset\} \qquad [\![a]\!]_C = \{(A, Z) \in 2^A \times 2^C \mid a \in A\}$$
$$[\![\Box]\!]_C(B) = \{(A, Z) \in 2^A \times 2^C \mid Z \subseteq B\} \qquad [\![\neg a]\!]_C = \{(A, Z) \in 2^A \times 2^C \mid a \notin A\}$$

The expressive power of the modal μ-calculus is demonstrated by the formulas

$$\mu X.\, \nu Y.\, (p \wedge \Diamond Y) \vee \Diamond X \qquad\qquad \nu X.\, \mu Y.\, (p \wedge \Diamond X) \vee \Diamond Y.$$

The former is a co-Büchi formula expressing the existence of a path on which p holds forever, from some point on; the latter formula expresses the Büchi property that there is a path on which the atom p is satisfied infinitely often.

(2) The *graded μ-calculus* [26] allows expressing quantitative properties with the help of modal operators $\langle n \rangle$ and $[n]$, $n \in \mathbb{N}$; formulas $\langle n \rangle \psi$ and $[n]\psi$ then have the intuitive meaning that 'there are more than n successor states that satisfy ψ', and 'all but at most n successor states satisfy ψ', respectively. Its coalgebraic interpretation is based on *multigraphs*, which are coalgebras for the multiset functor [5]. A graded variant of the above Büchi property is specified, e.g., by the formula $\nu X.\, \mu Y.\, (p \wedge \langle n \rangle X) \vee \langle n \rangle Y$, which expresses the existence of an infinite $n+1$-ary tree such that the atom p is satisfied infinitely often on every path in the tree.

(3) The *alternating-time μ-calculus* (AMC) [41] extends coalition logic [37] with fixpoints and (modulo syntax) supports modalities $\langle D \rangle$ and $[D]$, where $D \subseteq N$ is a coalition formed by agents from the set $N = \{1, \ldots, n\}$ for some fixed $n \in \mathbb{N}$; formulas $\langle D \rangle \psi$ and $[D]\psi$ then state that 'coalition D has a joint strategy to enforce ψ' and that 'coalition D cannot prevent ψ', respectively. For instance, the formula $\nu X.\, \mu Y.\, \nu Z.\, (p \wedge \langle D \rangle X) \vee (q \wedge \langle D \rangle Y) \vee (\neg q \wedge \langle D \rangle Z)$ expresses that coalition D has a joint multi-step strategy that guarantees that p is visited infinitely often whenever q is visited infinitely often.

Satisfiability Checking: We proceed to recall the satisfiability checking algorithm for the coalgebraic μ-calculus that forms the basis of the implementation within COOL 2. This algorithm adapts the automata-based approach to satisfiability checking for the standard μ-calculus, and generalises the treatment of modal steps by parametrizing over a solver for the *one-step satisfiability* problem of the logic, which concerns satisfiability of formulae with exactly one layer of next-step modalities [21]. It thus avoids the necessity of tractable sets of tableaux rules for modal operators. Under mild assumptions on the complexity of the one-step satisfiability problem of the base logic at hand ('*tractability*'), the algorithm witnesses a, typically optimal, upper bound ExpTime for the complexity of the satisfiability problem; unlike a previous algorithm [4], the algorithm thus has optimal runtime also in cases where no tractable sets of modal tableaux rules are known, such as the graded (or, more generally, Presburger) μ-calculus (further cases of this kind include the probabilistic μ-calculus with polynomial inequalities [21] and the unrestricted form of the *alternating-time μ-calculus with disjunctive explicit strategies* [16]).

The algorithm constructs and solves a parity game that characterises satisfiability of the input formula χ. In this game one player attempts to construct a tableau structure for χ while the opposing player attempts to refute the existence of such a structure. Modal steps in this tableau construction are treated by using instances of the one-step satisfiability problem for the logic at hand, thereby

generalising traditional modal tableau rules. The winning condition of the game is encoded by a non-deterministic parity automaton A_χ, reading infinite words that encode sequences of step-wise formula evaluations (so-called *formula traces*) within a coalgebra; such words encode branches in the constructed tableau structure. Conjunctions give rise to nondeterminism in this automaton, and the parity condition of the automaton is used to accept exactly those words that encode sequences of formula evaluations in which some least fixpoint is unfolded infinitely often. To use the language accepted by A_χ as the winning condition in a parity game, we transform A_χ to an equivalent deterministic parity automaton B_χ. This automaton then is paired with the tableau construction to yield a parity game in which the existential player aims to show the existence of a tableau structure in which all branches are rejected by B_χ, and that is built in such a way that modalities always are jointly one-step satisfiable. To ensure the latter property, the modal moves in the game invoke instances of the one-step satisfiability problem of the base logic. For more details on one-step satisfiability and the overall algorithm, see the appendix as well as [21].

Corollary 2.2 ([21]). *Suppose that the one-step satisfiability problem is tractable. Then the satisfiability problem of the corresponding instance of the coalgebraic μ-calculus is in* ExpTime.

Remark 2.3. As mentioned above, previous algorithms for the coalgebraic μ-calculus (also implemented in COOL 2) rely on complete sets of modal tableau rules, specifically on one-step cutfree complete sets of so-called *one-step rules* [45]; such rules (in their incarnation as tableau rules) have a premiss with exactly one layer of modal operators and a purely propositional conclusion. A typical example is the usual tableau rule for the modal logic K: 'To satisfy $\Box a_1 \wedge \cdots \wedge \Box a_n \wedge \neg\Box a_0$, satisfy $a_1 \wedge \cdots \wedge a_n \wedge \neg a_0$'. It has been shown that the existence of a tractable one-step cutfree complete set of one-step rules implies tractability of one-step satisfiability [29], i.e. the approach via one-step satisfiability is more general.

As indicated in the introduction, a tractable one-step cutfree complete set of one-step rules for graded modal logic has been claimed in the literature [45, 48] but has since turned out to be incomplete; we give a counterexample in the appendix. (A similar rule for Presburger modal logic [28] has also been shown to be in fact incomplete [29].)

3 Implementation

The previous version COOL [15] only implements fixpoint-free (coalgebraic) logics, such as standard modal logic, probabilistic modal logic, or coalition logic. The main novelty of the new version COOL 2, described here, is

– the addition of fixpoint constructs to the previously implemented logics, supporting alternation-free and aconjunctive fragments of the resulting μ-calculi, and implementing on-the-fly solving to allow early termination

– support for treating modal steps both by tableaux rules (when a suitable rule set exists), and by one-step satisfiability checking (in the remaining cases)

In more detail, COOL 2 is written in OCaml and implements the satisfiability checking algorithm described in Section 2, treating modal steps by solving instances of the one-step satisfiability problem[4]. For logics where a suitable set of modal tableau rules is implemented, those are used for the treatment of modal steps, rather than relying on one-step satisfiability (unless the user explicitly chooses otherwise); in these cases, COOL 2 essentially implements the algorithm described in [29]. The current implementation supports the alternation-free and the aconjunctive fragments of the standard μ-calculus (both serial and non-serial), the monotone μ-calculus [19], the alternating-time μ-calculus (i.e. coalition logic with fixpoint operators), and the graded μ-calculus. Tractable tableaux rules are available for all cases except for the graded μ-calculus, for which COOL 2 uses the one-step satisfiability algorithm to decide satisfiability. In particular, COOL 2 is the only existing reasoner for the graded μ-calculus (as well as the only reasoner covering the alternating-time μ-calculus beyond ATL).

The concrete logic used can be selected via a command-line parameter setting up the data structures in COOL 2 accordingly before parsing and checking the syntax of the given formula χ. COOL 2 then builds the determinised automaton B_χ, yielding the parity game described above in a step-wise manner, repeatedly adding nodes in *expansion steps* that explore the game. In the case of simpler alternation-free formulas, the Miyano-Hayashi method [31] is used to construct B_χ, resulting in asymptotically smaller games with a Büchi winning condition; for the more involved aconjunctive formulas, the implementation uses the permutation method for determinisation of limit-deterministic parity automata [9,22]. Nodes in the constructed game are marked as either unexpanded, undecided, unsatisfiable, or satisfiable.

Optional *solving steps* may take place at any point during the construction of B_χ, depending on runtime parameters of COOL 2; these steps compute the winning regions of the partial game that has been constructed so far and accordingly mark nodes as satisfiable or unsatisfiable, if possible. The reasoner terminates as soon as the initial node is marked satisfiable or unsatisfiable. If this does not allow for early termination, the game eventually becomes fully explored, at which point a final (obligatory) solving step for the complete game is guaranteed to mark the initial node, thereby ensuring termination.

We detail the implementation of the two main procedures within COOL 2.

Implementation of Expansion Steps. The propositional expansion steps in the game construction for nodes v are performed using the propositional satisfiability solver MiniSat [8] to compute a word that encodes consistent propositional formula manipulations for v. Afterwards, the successor of v in B_χ under this word is computed and added to the game.

When the one-step satisfiability based algorithm of COOL 2 is used, modal expansion steps for nodes v create new game nodes for each subset κ of the

[4] Sources are available at `https://git8.cs.fau.de/software/cool`

modalities that are to be jointly satisfied at v; this is done by computing the successor of v in B_χ that is reached by manipulating each formula from κ.

When the tableau-based algorithm of COOL 2 is used, the modal expansion step for a node v instead computes all applications of a modal rule matching v and inserts, for each such rule application, and each conjunctive clause κ in the conclusion of the rule application, the new game node that is reached from v in B_χ by manipulating the modalities that constitute κ. Intuitively, using tableau rules reduces the search space by only adding nodes found in the conclusion of some matching rule application.

Any node that is added by some expansion step is initially marked as undecided. Crucially, all expansion steps perform on-the-fly determinisation, that is, given a game node v and a word that encodes a sequence of formula manipulations, the newly added game node is computed using only the information stored in v.

Implementation of Solving Steps. A single solving step computes the winning regions in the parity game that has been constructed up to this point, and marks nodes accordingly. The game solving is done using either the parity game solver PGSolver [10] or a native implementation provided by COOL 2 that solves the game by fixpoint iteration.

If the one-step satisfiability-based algorithm is used, an assigned modal node v is satisfiable if its modalities are jointly one-step satisfiable in those successors of v that are satisfiable themselves. An enumerative representation of the game thus contains existential moves to all subsets Π of subsets of modalities of v that are sufficiently large for one-step satisfaction of the modalities of v, followed by universal moves to nodes induced by any $\kappa \in \Pi$; the full game thus is of doubly-exponential size. This can be avoided by inlining the modal steps, thereby evading the intermediate nodes Π. The winning region can then be computed in single-exponential time by using COOL 2's native fixpoint iteration over a function that computes the two-tiered modal steps in one go.

Decision procedures for the one-step satisfiability problems in the relational and the graded case are implemented in COOL 2 along the lines of the algorithms described in [21, Example 6] (in the graded case, nondeterministic guessing is replaced with a recursive search procedure).

If the algorithm based on modal tableau rules is used, the treatment of modal steps follows the tableaux-based algorithm that is given in [3]. States v are satisfiable if for all rule applications that match v, the conclusion of the application contains a conjunctive clause κ such that the node induced by κ is satisfiable.

COOL 2 also allows the user to specify the desired frequency of optional game solving steps, including the options once and adaptive. With the option once, no intermediate solving takes place so that the game is fully constructed and solved just once, at the very end of the execution. With the option adaptive, intermediate solving takes places, but the frequency of solving reduces as the size of the constructed graph increases; this option implements *on-the-fly* solving and allows for finishing early in cases where a small model or refutation exists.

4 Evaluation

We conduct experiments in order to evaluate the performance of the various algorithms implemented in COOL in comparison with each other, as well as in comparison with other tools (where applicable).[5] Complete definitions of all formula series used in the evaluation as well as additional experimental results can be found in the appendix.

Experiments: In a first experiment, we compare COOL 2 with the established reasoner FaCT++, which supports the description logic $\mathcal{SROIQ}(\mathcal{D})$ (subsuming fixpoint-free graded modal logic), using the following series of formulas from Snell et al. [48].

$$\mathsf{Cardinality}(n) := \langle n-1\rangle\neg p \wedge \langle n-1\rangle p \wedge [n]\neg q \wedge [n]q \qquad \text{(Sat)}$$

$$\mathsf{CardinalityU}(n) := \langle n-1\rangle\neg p \wedge \langle n-1\rangle p \wedge [n]\neg q \wedge [n-1]q \qquad \text{(UnSat)}$$

Intuitively, the satisfiable $\mathsf{Cardinality}(n)$ formulas express that there are at least $2n$ successors and that both q and $\neg q$ are satisfied in at most n successors, each; similarly the unsatisfiable $\mathsf{CardinalityU}(n)$ formulas state that there are at least $2n$ successors, and that q and $\neg q$ hold in at most n and $n-1$ successors, respectively; the latter statements imply that there are at most $2n-1$ successors, yielding a contradiction.

Going beyond next-step formulas, we continue by devising various complex series of graded μ-calculus formulas that involve (nested) fixpoints and express non-trivial properties of graded trees, automata and games.

– We obtain a series of unsatisfiable formulas by requiring the existence of an $n+1$-branching tree in which p holds everywhere while at the same time requiring that this tree contains some state with $n+2$ successors that satisfy p:

$$\mathsf{TreeU}(n) = (\nu X.\, \langle n\rangle(p \wedge X) \wedge [n+1]\neg p) \wedge (\mu Y.\, \langle n+1\rangle p \vee \langle n\rangle(p \wedge Y)) \qquad \text{(UnSat)}$$

– Next we turn our attention to graded formulas involving parity conditions. We devise a series of valid formulas expressing that graded parity automata can be transformed to graded Büchi automata accepting a superlanguage of the original automaton:

$$\mathsf{ParityToBuechi}(n,k) := \mathsf{Parity}(n,k) \rightarrow \mathsf{Buechi}(n,k) \qquad \text{(Valid)}$$

Here, $\mathsf{Parity}(n,k)$ encodes parity acceptance with k priorities and grade n while $\mathsf{Buechi}(n,k)$ expresses Büchi acceptance by a nondeterministic automaton that eventually guesses the maximal priority that occurs infinitely often; the negated formula $\neg\mathsf{ParityToBuechi}(n,k)$ is unsafisfiable.

– Rabin conditions are given by families of pairs $\langle i_j, f_j\rangle_{j\leq k}$ of sets i_j, f_j of states, and express the constraint that there is some $j \leq k$ such that states from i_j (*infinite*) are visited infinitely often and states from f_j (*finite*) are visited

[5] Scripts and executables that allow for reproducing our experiments can be found at DOI 10.5281/zenodo.8042581

only finitely often. We can express Rabin conditions with k pairs (and one-step property ψ), Büchi properties and satisfaction of single Rabin-pairs by formulas $\mathsf{Rabin}(k, \psi)$, $\mathsf{Buechi}(f, \psi)$ and $\mathsf{RabinPair}(i, f, \psi)$, respectively. Then we obtain valid formulas stating that the existence of an $n+1$-branching tree that satisfies the Rabin condition on each path implies that there is a path satisfying a simpler Büchi condition or a single Rabin-pair, respectively:

$$\mathsf{RabinToBuechi}(k, n) := \mathsf{Rabin}(k, \langle n \rangle) \rightarrow \mathsf{Buechi}(i_1 \vee \ldots \vee i_k, \langle 0 \rangle) \quad \text{(Valid)}$$

$$\mathsf{RabinToRPair}(k, n) := \mathsf{Rabin}(k, \langle n \rangle) \rightarrow \bigvee_{1 \leq j \leq k} \mathsf{RabinPair}(i_j, f_j, \langle 0 \rangle) \quad \text{(Valid)}$$

– Coming to games, we specify the winning regions in graded Büchi and Rabin games by formulas $\mathsf{BuechiG}(f, n)$ and $\mathsf{RabinG}(k, n)$, respectively; in such graded games, players are required to have at least n winning moves at their nodes in order to win. The following valid formulas then express that winning strategies in graded Rabin games with k pairs guarantee that some node from $i_1 \cup \ldots \cup i_k$ is visited infinitely often:

$$\mathsf{RabinGame}(k, n) := \mathsf{RabinG}(k, n) \rightarrow \mathsf{BuechiG}(i_1 \vee \ldots \vee i_k, n) \quad \text{(Valid)}$$

In a final experiment on alternating-time formulas, we compare COOL 2 with TATL [6] on the ATL example formulas given in [6] as well as on additional formula series. For instance, we turn the formula $\langle\langle 1 \rangle\rangle Gp \wedge \neg\langle\langle 2 \rangle\rangle F\langle\langle 1 \rangle\rangle Gp$ (written here using ATL syntax) from [6] into a series $\mathsf{Nest}(n)$ with increasing number of nested operators; formulas then alternatingly are satisfiable and unsatisfiable:

$$\chi(0) = p \qquad \chi(i+1) = \neg\langle\langle 2 \rangle\rangle F\langle\langle 1 \rangle\rangle G\chi(i) \qquad \mathsf{Nest}(n) = \langle\langle 1 \rangle\rangle Gp \wedge \chi(n),$$

Results: We conducted all experiments on a virtual machine with four 2,3GHz vCPUs processors and 8GB of RAM. We compare with a 64-bit binary of FaCT++ v1.6.5 and with TATL. We compute all results with a timeout of 60 seconds and average the results over multiple executions. For the execution and measurement we use hyperfine[6]. Below, 'COOL' and 'COOL on-the-fly' refer to invoking COOL 2 with solving rate once and adaptive, respectively.

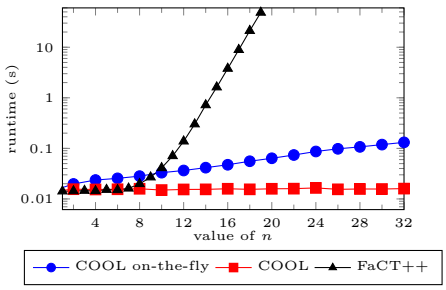

Fig. 1. Runtimes for $\mathsf{Cardinality}(n)$

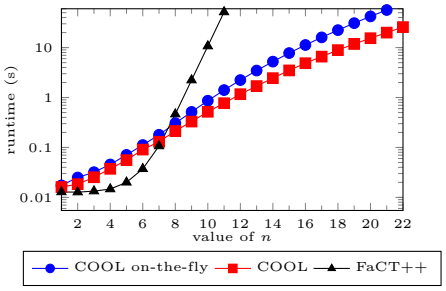

Fig. 2. Runtimes for $\mathsf{CardinalityU}(n)$

[6] https://github.com/sharkdp/hyperfine

Results for the Cardinality and CardinalityU series are shown in Figure 1 and Figure 2, respectively. From $n = 10$ and $n = 8$ onwards, COOL 2 outperforms FaCT++ considerably. An explanation for this could be that FaCT++ appears to treat multiplicities in a naïve way while COOL 2 employs the more efficient one-step satisfiability algorithm.

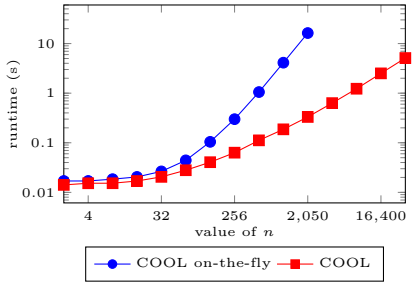

Fig. 3. Runtimes for TreeU(n)

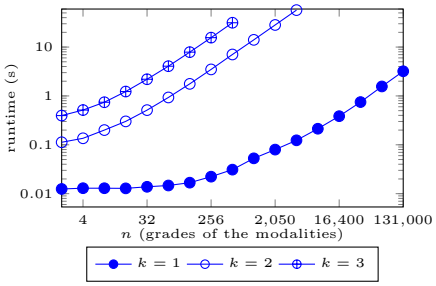

Fig. 4. Runtimes for ¬ParityToBuechi(n, k)

Results for the unsatisfiable tree property are shown in Figure 3. As these formulas contain fixpoint operators, a comparison with FaCT++ is not possible. While COOL 2 is generally capable of handling quite large branching factors, this experiment showcases the drawbacks of on-the-fly solving in the case that a formula cannot be decided early so that repeated attempts of solving the game early lead to overhead computations.

Runtimes for COOL 2 (using on-the-fly solving) on the unsatisfiable series of parity formulas ¬ParityToBuechi(n, k) are shown in Figure 4. The results indicate that increasing the number of priorities k has a much stronger effect on the runtime than increasing multiplicities n in the modalities. This is in accordance with expectations as increasing k leads to much larger determinized automata and resulting satisfiabilty games, while increasing n only complicates the modal steps in the game while leaving the global game structure unchanged.

Results for the Rabin families of formulas are given in the table below, with † indicating a timeout of 60 seconds. COOL 2 is able to handle reasonably large formulas describing Rabin properties of automata and games, with the series for $n = 1$ expressing properties of standard automata (solved using tableau rules), and the series with $n = 2$ properties of graded automata with multiplicity 2 (solved using one-step satisfiability).

In accordance with previous experiments on random ATL formulas of larger sizes in [23], COOL 2 generally outperforms TATL by a large margin, starting from formulas containing at least five modalities or involving nesting of temporal operators; this trend is confirmed by Figure 5 which shows the stepped execution times for the series Nest that alternates between being satisfiable and unsatisfiable

series$\backslash k$	1	2	3
RabinToBuechi$(k,1)$	0.03	0.51	45.25
RabinToBuechi$(k,2)$	0.08	10.56	†
RabinToRPair$(k,1)$	0.03	8.38	†
RabinToRPair$(k,2)$	0.07	†	†
RabinGame$(k,1)$	0.05	1.04	†
RabinGame$(k,2)$	0.31	43.94	†

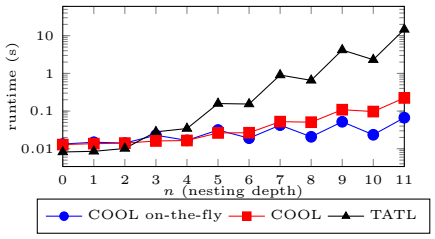

Fig. 5. Runtimes for the ATL series Nest(n)

In summary, COOL 2 shows promising performance in comparison to TATL and FaCT++, as well as for practical applicability. On graded formulas without fixpoints, COOL 2 scales much better than FaCT++ with regard to increasing multiplicities. In the presence of fixpoints, COOL 2 still scales well and can handle multiplicities that should be sufficient for practical use. The formula series ¬ParityToBuechi appears to show the limits of COOL 2 with the current implementation of graded one-step satisfiability checking. Nonetheless, our results indicate that COOL 2 is capable of automatically proving or refuting involved properties of (graded) ω-automata and games in reasonable time.

5 Conclusion

We have described and evaluated the current version COOL 2 of the *CO*algebraic *O*ntology *L*ogic reasoner (COOL). Future development will include the implementation of additional instance logics, such as the probabilistic and graded μ-calculus with linear inequalities, as well as support for the full coalgebraic μ-calculus via on-the-fly determinisation of *unrestricted* Büchi automata, using the Safra-Piterman construction.

We would like to thank Frederik Hennig for finding and correcting a slight mistake in the Rabin-type formulas in an earlier version of this paper; the corresponding runtimes reported in the table above as well as the full formulas in the appendix have been updated accordingly.

References

1. Alur, R., Henzinger, T., Kupferman, O.: Alternating-time temporal logic. J. ACM **49**, 672–713 (2002), https://doi.org/10.1145/585265.585270
2. Calin, G., Myers, R., Pattinson, D., Schröder, L.: CoLoSS: The coalgebraic logic satisfiability solver. In: Methods for Modalities, M4M-5. ENTCS, vol. 231, pp. 41–54. Elsevier (2009), https://doi.org/10.1016/j.entcs.2009.02.028
3. Cîrstea, C., Kupke, C., Pattinson, D.: EXPTIME tableaux for the coalgebraic μ-calculus. Log. Meth. Comput. Sci. **7** (2011), https://doi.org/10.2168/LMCS-7(3:3)2011

4. Cîrstea, C., Kurz, A., Pattinson, D., Schröder, L., Venema, Y.: Modal logics are coalgebraic. Comput. J. **54**, 31–41 (2011), https://doi.org/10.1093/comjnl/bxp004
5. D'Agostino, G., Visser, A.: Finality regained: A coalgebraic study of Scott-sets and multisets. Arch. Math. Logic **41**, 267–298 (2002), https://doi.org/10.1007/s001530100110
6. David, A.: TATL: Implementation of ATL tableau-based decision procedure. In: Automated Reasoning with Analytic Tableaux and Related Methods, TABLEAUX 2013. LNCS, vol. 8123, pp. 97–103. Springer (2013), https://doi.org/10.1007/978-3-642-40537-2_10
7. David, A.: Deciding ATL* satisfiability by tableaux. In: Felty, A.P., Middeldorp, A. (eds.) Automated Deduction - CADE-25. pp. 214–228. Springer International Publishing, Cham (2015), https://doi.org/10.1007/978-3-319-21401-6_14
8. Eén, N., Sörensson, N.: An extensible SAT-solver. In: Giunchiglia, E., Tacchella, A. (eds.) Theory and Applications of Satisfiability Testing, SAT 2003. LNCS, vol. 2919, pp. 502–518. Springer (2003), https://doi.org/10.1007/978-3-540-24605-3_37
9. Esparza, J., Kretínský, J., Raskin, J., Sickert, S.: From linear temporal logic and limit-deterministic Büchi automata to deterministic parity automata. Int. J. Softw. Tools Technol. Transf. **24**(4), 635–659 (2022), https://doi.org/10.1007/s10009-022-00663-1
10. Friedmann, O., Lange, M.: The PGSolver collection of parity game solvers. Tech. rep., LMU Munich (2009)
11. Friedmann, O., Lange, M.: A solver for modal fixpoint logics. In: Methods for Modalities, M4M-6 2009. ENTCS, vol. 262, pp. 99–111 (2010), https://doi.org/10.1016/j.entcs.2010.04.008
12. Glimm, B., Horrocks, I., Motik, B., Stoilos, G., Wang, Z.: Hermit: An OWL 2 reasoner. J. Autom. Reason. **53**(3), 245–269 (2014), https://doi.org/10.1007/s10817-014-9305-1
13. Goré, R., Kupke, C., Pattinson, D., Schröder, L.: Global caching for coalgebraic description logics. In: Automated Reasoning, IJCAR 2010. LNCS, vol. 6173, pp. 46–60. Springer (2010), https://doi.org/10.1007/978-3-642-14203-1_5
14. Goré, R., Thomson, J., Widmann, F.: An experimental comparison of theorem provers for CTL. In: Temporal Representation and Reasoning, TIME 2011. pp. 49–56. IEEE (2011), https://doi.org/10.1109/TIME.2011.16
15. Gorín, D., Pattinson, D., Schröder, L., Widmann, F., Wißmann, T.: COOL – a generic reasoner for coalgebraic hybrid logics (system description). In: Automated Reasoning, IJCAR 2014. LNCS, vol. 8562, pp. 396–402. Springer (2014), https://doi.org/10.1007/978-3-319-08587-6_31
16. Göttlinger, M., Schröder, L., Pattinson, D.: The alternating-time μ-calculus with disjunctive explicit strategies. In: Baier, C., Goubault-Larrecq, J. (eds.) Computer Science Logic, CSL 2021. LIPIcs, vol. 183, pp. 26:1–26:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2021), https://doi.org/10.4230/LIPIcs.CSL.2021.26
17. Grädel, E., Thomas, W., Wilke, T. (eds.): Automata, Logics, and Infinite Games: A Guide to Current Research, LNCS, vol. 2500. Springer (2002), https://doi.org/10.1007/3-540-36387-4
18. Haarslev, V., Möller, R.: RACER system description. In: Goré, R., Leitsch, A., Nipkow, T. (eds.) Automated Reasoning, IJCAR 2001. LNCS, vol. 2083, pp. 701–706. Springer (2001), https://doi.org/10.1007/3-540-45744-5_59

19. Hansen, H.H., Kupke, C., Marti, J., Venema, Y.: Parity games and automata for game logic. In: Dynamic Logic. New Trends and Applications, DALI 2017. LNCS, vol. 10669, pp. 115–132. Springer (2018), https://doi.org/10.1007/978-3-319-73579-5_8
20. Hansson, H., Jonsson, B.: A logic for reasoning about time and reliability. Formal Asp. Comput. **6**, 512–535 (1994), https://doi.org/10.1007/BF01211866
21. Hausmann, D., Schröder, L.: Optimal satisfiability checking for arithmetic μ-calculi. In: Foundations of Software Science and Computation Structures, FOSSACS 2019. LNCS, vol. 11425, pp. 277–294. Springer (2019), https://doi.org/10.1007/978-3-030-17127-8_16
22. Hausmann, D., Schröder, L., Deifel, H.: Permutation games for the weakly aconjunctive μ-calculus. In: Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2018. LNCS, vol. 10806, pp. 361–378. Springer (2018), https://doi.org/10.1007/978-3-319-89963-3_21
23. Hausmann, D., Schröder, L., Egger, C.: Global caching for the alternation-free coalgebraic μ-calculus. In: Concurrency Theory, CONCUR 2016. LIPIcs, vol. 59, pp. 34:1–34:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2016), https://doi.org/10.4230/LIPIcs.CONCUR.2016.34
24. Kozen, D.: Results on the propositional μ-calculus. Theor. Comput. Sci. **27**, 333–354 (1983), https://doi.org/10.1016/0304-3975(82)90125-6
25. Kupferman, O., Piterman, N., Vardi, M.: Fair equivalence relations. In: Verification: Theory and Practice. LNCS, vol. 2772, pp. 702–732. Springer (2003), https://doi.org/10.1007/978-3-540-39910-0_30
26. Kupferman, O., Sattler, U., Vardi, M.: The complexity of the graded μ-calculus. In: Automated Deduction, CADE 2002. LNCS, vol. 2392, pp. 423–437. Springer (2002), https://doi.org/10.1007/3-540-45620-1_34
27. Kupke, C., Marti, J., Venema, Y.: Size measures and alphabetic equivalence in the μ-calculus. In: Baier, C., Fisman, D. (eds.) Logic in Computer Science, LICS 2022. pp. 18:1–18:13. ACM (2022), https://doi.org/10.1145/3531130.3533339
28. Kupke, C., Pattinson, D.: On modal logics of linear inequalities. In: Advances in Modal Logic, AiML 2010. pp. 235–255. College Publications (2010)
29. Kupke, C., Pattinson, D., Schröder, L.: Coalgebraic reasoning with global assumptions in arithmetic modal logics. ACM Trans. Comput. Log. **23**(2), 11:1–11:34 (2022), https://doi.org/10.1145/3501300
30. Larsen, K., Skou, A.: Bisimulation through probabilistic testing. Inform. Comput. **94**, 1–28 (1991), https://doi.org/10.1016/0890-5401(91)90030-6
31. Miyano, S., Hayashi, T.: Alternating finite automata on ω-words. Theor. Comput. Sci. **32**, 321–330 (1984), https://doi.org/10.1016/0304-3975(84)90049-5
32. Myers, R., Pattinson, D., Schröder, L.: Coalgebraic hybrid logic. In: Foundations of Software Science and Computational Structures, FOSSACS 2009. LNCS, vol. 5504, pp. 137–151. Springer (2009), https://doi.org/10.1007/978-3-642-00596-1_11
33. Nalon, C., Zhang, L., Dixon, C., Hustadt, U.: A resolution prover for coalition logic. In: Mogavero, F., Murano, A., Vardi, M.Y. (eds.) Strategic Reasoning, SR 2014. EPTCS, vol. 146, pp. 65–73 (2014), https://doi.org/10.4204/EPTCS.146.9
34. Niwinski, D.: On fixed-point clones (extended abstract). In: Automata, Languages and Programming, ICALP 1986. LNCS, vol. 226, pp. 464–473. Springer (1986), https://doi.org/10.1007/3-540-16761-7_96
35. Parikh, R.: Propositional game logic. In: Foundations of Computer Science, FOCS 1983. IEEE Computer Society (1983), https://doi.org/10.1109/SFCS.1983.47

36. Pattinson, D.: Expressive logics for coalgebras via terminal sequence induction. Notre Dame J. Formal Logic **45**, 19–33 (2004), https://doi.org/10.1305/ndjfl/1094155277
37. Pauly, M.: A modal logic for coalitional power in games. J. Logic Comput. **12**, 149–166 (2002), https://doi.org/10.1093/logcom/12.1.149
38. Peleg, D.: Concurrent dynamic logic. J. ACM **34**, 450–479 (1987), https://doi.org/10.1145/23005.23008
39. Piterman, N.: From nondeterministic Büchi and Streett automata to deterministic parity automata. Log. Meth. Comput. Sci. **3** (2007), https://doi.org/10.2168/LMCS-3(3:5)2007
40. Rutten, J.: Universal coalgebra: A theory of systems. Theor. Comput. Sci. **249**, 3–80 (2000), https://doi.org/10.1016/S0304-3975(00)00056-6
41. Schewe, S.: Synthesis of distributed systems. Ph.D. thesis, Universität des Saarlands (2008)
42. Schröder, L.: A finite model construction for coalgebraic modal logic. J. Log. Algebr. Prog. **73**, 97–110 (2007), https://doi.org/10.1016/j.jlap.2006.11.004
43. Schröder, L.: Expressivity of coalgebraic modal logic: The limits and beyond. Theor. Comput. Sci. **390**(2-3), 230–247 (2008), https://doi.org/10.1016/j.tcs.2007.09.023
44. Schröder, L., Pattinson, D.: Shallow models for non-iterative modal logics. In: Advances in Artificial Intelligence, KI 2008. LNCS, vol. 5243, pp. 324–331. Springer (2008), https://doi.org/10.1007/978-3-540-85845-4_40
45. Schröder, L., Pattinson, D.: PSPACE bounds for rank-1 modal logics. ACM Trans. Comput. Log. **10**(2), 13:1–13:33 (2009), https://doi.org/10.1145/1462179.1462185
46. Schröder, L., Pattinson, D.: Strong completeness of coalgebraic modal logics. In: Theoretical Aspects of Computer Science, STACS 09. pp. 673–684. Schloss Dagstuhl – Leibniz-Zentrum für Informatik; Dagstuhl, Germany (2009), https://doi.org/10.4230/LIPIcs.STACS.2009.1855
47. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. J. Web Semant. **5**(2), 51–53 (2007), https://doi.org/10.1016/j.websem.2007.03.004
48. Snell, W., Pattinson, D., Widmann, F.: Solving graded/probabilistic modal logic via linear inequalities (system description). In: Logic for Programming, Artificial Intelligence, and Reasoning, LPAR 2012. LNCS, vol. 7180, pp. 383–390. Springer (2012), https://doi.org/10.1007/978-3-642-28717-6_30
49. Tsarkov, D., Horrocks, I.: FaCT++ description logic reasoner: System description. In: Furbach, U., Shankar, N. (eds.) Automated Reasoning, IJCAR 2006. LNAI, vol. 4130, pp. 292–297. Springer (2006), https://doi.org/10.1007/11814771_26

A Appendix: Additional Details

A.1 Details for Remark 2.3

The following set of modal tableau rules (one-step rules) has been claimed to be one-step cutfree complete [46, 48]:

$$\frac{\langle n_1\rangle p_1 \wedge \cdots \wedge \langle n_u\rangle p_u \wedge \neg\langle m_1\rangle q_1 \wedge \cdots \wedge \neg\langle m_v\rangle q_v}{\sum_{i=1}^{u} r_i p_i - \sum_{i=1}^{v} s_i q_i > 0}$$

for $r_i, s_i \in \mathbb{N} \setminus \{0\}$, subject to the side condition

$$\sum_{i=1}^{u} r_i(n_i + 1) - \sum_{i=1}^{v} s_i m_i \geq 1.$$

The conclusion of the rule denotes a propositional formula (in conjunctive normal form) representing the Boolean function obtained by reading the numerical inequality as a constraint on Boolean variables p_i, q_i, with true interpreted as 1 and false as 0. In the relevant instance of the one-step logic, models consist of a set X, a valuation τ of the variables p_i, q_i as subsets of X, and a finite multiset μ over X. Given these data, the conclusion ϕ of (an instance of) the rule, a purely propositional formulas, is interpreted as a subset $[\![\phi]\!]\tau$ of X, using the Boolean algebra structure of the powerset. The premiss is evaluated w.r.t. satisfaction by μ, with the expected clauses for the propositional operators, and with $\langle n\rangle a$, for a variable a, being satisfied if $\mu(\tau(a)) > n$, where $\mu(A) = \sum_{x\in A} \mu(x)$ for $A \subseteq X$. The premiss is *satisfiable over* τ if $[\![\phi]\!]\tau \neq \emptyset$, and the conclusion is satisfiable over τ if it is satisfied by some multiset μ over X in the sense just defined.

In the one-step logic, the rules are then applied to conjunctions of *modal literals*, i.e. formulas of the form either $\langle n\rangle a$ or $\neg\langle n\rangle a$, for a a variable, requiring an exact match of the rule premiss with a subset of the conjuncts (thus incorporating weakening into the rule application). Such formulas are called *one-step clauses*. The rules are read as tableau rules, i.e. to establish that the premiss is satisfiable, one needs to establish that the conclusions of all matching rule applications are satisfiable.

The rules are easily seen to be *one-step sound*, i.e. if for any rule instance matching a given one-step clause, the conclusion is propositionally unsatisfiable over a valuation τ, then the premiss is also unsatisfiable over τ. The rule set is *one-step complete* if, given a valuation τ in the powerset of X and a one-step clause χ, whenever all rule matches to χ have satisfiable conclusions, then χ is satisfiable.

We show that the latter property fails. To this end, consider the set $X = \{a, b, c, d\}$, propositional variables p_A for all $A \subseteq X$, the valuation τ given by $\tau(p_A) = A$, and the one-step clause χ consisting of the positive literals $\langle 2\rangle p_A$ for all $A \subseteq X$ such that $|A| = 2$, and the negative literal $\neg\langle 6\rangle p_X$. First, note that χ is clearly unsatisfiable over τ: A multiset μ over X would have to satisfy $\mu(A) \geq 3$ for all $A \subseteq X$ such that $|A| = 2$, so at least three of the four elements of X need to

have multiplicity at least 2 under μ; moreover, if any element has multiplicity 0, then all others need to have multiplicity at least 3. Consequently, the total weight $\mu(X)$ is at least 7, so μ does not satisfy the negative literal $\neg\langle 6\rangle p_X$. On the other hand, none of the rule instances matching χ have unsatisfiable conclusions. The easiest way to see this is to note that the rules are, with fairly evident adaptations, still sound for a real-valued relaxation of the logic where μ may assume non-negative real values (of course, $\langle n\rangle a$ then means that $\mu(\tau(a)) \geq n+1$); under this semantics, however, χ is satisfiable over τ by taking $\mu(x) = \frac{3}{2}$ for all $x \in X$.

A.2 Additional Details for Section 2

We give additional details on syntactic notions for the coalgebraic μ-calculus, and on predicate liftings. Furthermore, we sketch the satisfiability checking algorithm, first introduced in [21]; here we give a presentation of the algorithm in terms of automata and games, tailored towards our implementation in COOL 2.

Fixpoint operators *bind* their variable, yielding notions of bound and free fixpoint variables; a formula then is *closed* if it does not contain any free variables. A formula is *clean* if every fixpoint variable is bound at most once in it. We restrict attention to closed and clean formulas (being aware of issues with formula size measures [27]). Variables X that are bound by μX then are μ-variables, and variables bound by νX are ν-variables. A variable X is *active* in a formula ψ if X has a free occurrence in the formula that is obtained from ψ by exhaustively replacing free occurrences of fixpoint variables by their binding fixpoint formulas.

E.g. for the formula $\chi = \mu X.\,(p \vee \Diamond X)$ we have $\theta(X) = \chi$, and the closure of χ is a graph with nodes $\chi, p \vee \Diamond\chi, p, \Diamond\chi$ and edges $\chi \to p \vee \Diamond\chi$, $p \vee \Diamond\chi \to p$, $p \vee \Diamond\chi \to \Diamond\chi$, and $\Diamond\chi \to \chi$. When we refer to the closure, we typically mean just the set of nodes of this closure graph.

We give more details on Example 2.1:

A coalgebraic modelling of the graded μ-calculus [5] is obtained by using the functor $F = \mathcal{G}$ that maps a set C to the set $\mathcal{G}C = \{\theta : C \to \mathbb{N} \mid \theta \text{ has finite support}\}$ of finite multisets over C, the similarity type $\Lambda = \{\langle n\rangle, [n] \mid n \in \mathbb{N}\}$, and predicate liftings

$$[\![\langle n\rangle]\!]_C(B) = \{\theta \in \mathcal{G}C \mid \Sigma_{v\in B}\theta(v) > n\} \quad [\![\,[n]\,]\!]_C(B) = \{\theta \in \mathcal{G}C \mid \Sigma_{v\notin B}\theta(v) \leq n\}$$

The *concurrent game functor* $\mathcal{F}$ maps a set C to the set $\mathcal{F}C = \{(S_1, \ldots, S_n, f) \mid \emptyset \neq S_i, f : \prod_{i\leq n} S_i \to C\}$, where the S_i are viewed as sets of available moves for agent i, and f as an outcome function that evaluates joint moves of all agents. $\mathcal{F}$-Coalgebras are concurrent game frames [37]. Coalitions D induce sets $S_D = \prod_{i\in D} S_i$ and $S_{\overline{D}} = \prod_{i\in N\setminus D} S_i$, and given $s_D \in S_D$ and $s_{\overline{D}} \in S_{\overline{D}}$, the pair $(s_D, s_{\overline{D}})$ represents an element of $\prod_{i\leq n} S_i$. We use the modal similarity type $\Lambda = \{\langle D\rangle, [D] \mid D \subseteq N\}$ and the predicate liftings

$$[\![\langle D\rangle]\!]_C(B) = \{(S_1, \ldots, S_n) \in \mathcal{F}C \mid \exists s_D \in S_D.\, \forall s_{\overline{D}} \in S_{\overline{D}}.\, f(s_D, s_{\overline{D}}) \in B\}$$
$$[\![[D]]\!]_C(B) = \{(S_1, \ldots, S_n) \in \mathcal{F}C \mid \forall s_D \in S_D.\, \exists s_{\overline{D}} \in S_{\overline{D}}.\, f(s_D, s_{\overline{D}}) \in B\}$$

for $B \subseteq C$ and $D \subseteq N$.

Additional details regarding the satisfiability checking algorithm sketched in the main paper are as follows.

Depending on the syntactic structure of the input formula χ, it may be possible to employ simpler determinisation procedures for the construction of the automaton B_χ, resulting in asymptotically smaller games. Currently known bounds for particular fragments of the coalgebraic μ-calculus are summarised in the following table, where n denotes the closure size of the target formula, k denotes its alternation-depth, and where NCBA and LDBA stand for *nondeterministic co-Büchi automaton* and *limit-deterministic Büchi automaton*, respectively.

fragment	type of A_χ	determinisation	size	rank
alternation-free	NCBA	Miyano-Hayashi [31]	$\mathcal{O}(3^n)$	2
aconjunctive	LDBA	permutation method [9, 22]	$\mathcal{O}((nk)!)$	$2nk$
unrestricted	NBA	Safra-Piterman [39]	$2^{\mathcal{O}((nk)\log n)}$	$2nk$

We sketch the construction of the automata A_χ and B_χ to describe the coalgebraic satisfiability game. Recall that the accepted language $L(\mathsf{A})$ of a parity automaton $\mathsf{A} = (Q, \Sigma, \delta, v_0, \Omega)$ with priority function $\Omega : Q \to \mathbb{N}$ consists of those infinite words over Σ on which there is an accepting run of A, where a run is accepting if and only if the maximal priority (according to Ω) that is visited infinitely often is even. The logical connectives are captured by the alphabet of the automaton. Propositional connectives are treated by letters from a set Σ_p, also encoding choices of disjuncts using letters of the shape $(\psi_1 \vee \psi_2, b)$ where $\psi_1 \vee \psi_2 \in \mathsf{cl}$ and $b \in \{1, 2\}$. For modalities, the automaton needs to work on the conjunctive satisfiability of a set of operators, so we put

$$\Sigma_s := \{\kappa \in \mathcal{P}(\mathsf{cl}) \mid \forall \psi \in \kappa.\, \exists \heartsuit \in \Lambda.\, \psi = \heartsuit\phi\}.$$

The alphabet of the automata is $\Sigma = \Sigma_p \cup \Sigma_s$. Furthermore, we let $\mathsf{choices} = \{w \in \Sigma_p^* \mid |w| \leq n^2\}$ denote the set of propositional words of length at most n^2.

The nondeterministic parity automaton $\mathsf{A}_\chi = (\mathsf{cl}, \Sigma, \Delta, \chi, \Omega')$ reads sequences $w \in \Sigma^\omega$ of formula manipulations and traces formulas through the closure. For the transition function we have, for example $\Delta(\psi_1 \vee \psi_2, (\psi_1 \vee \psi_2, b)) = \{\psi_b\}$ for $b \in \{1, 2\}$, $\Delta(\psi_1 \wedge \psi_1, (\psi_1 \wedge \psi_1)) = \{\psi_1, \psi_2\}$, $\Delta(\heartsuit\psi, \kappa) = \{\psi\}$ if $\heartsuit\psi \in \kappa \in \Sigma_s$ and $\Delta(\heartsuit\psi, \kappa) = \emptyset$ otherwise. The priority function Ω' is defined by the alternation of least and greatest fixpoints in the formula, and A_χ accepts words encoding infinite traversals through formulas where the outermost unfolded fixpoint is a least fixpoint. We now consider the deterministic parity automaton $\mathsf{B}_\chi = (D_\chi, \Sigma, \delta, v_0, \Omega)$ that is obtained from A_χ by co-determinisation ($L(\mathsf{A}_\chi) = \overline{L(\mathsf{B}_\chi)}$), noting that B_χ is a parity automaton with at most $2nk$ priorities by the results from [39]. Then B_χ accepts infinite traversals through formulas for which there is no formulas trace (run of A_χ) on which the outermost fixpoint that is unfolded infinitely often is a least fixpoint.

The set D_χ consists of macro-states, that is, data structures organising elements of cl in way that depends on the concrete determinisation construction

that is used (see the table above), e.g., as Safra-trees [39] or permutations [9]. The *labelling* function $l\colon D_\chi \to \mathcal{P}(\mathsf{cl})$ assigns to each macro-state $v \in D_\chi$ its *label* $l(v)$, i.e. the set of formulas that occur in v. We denote by cores and states the sets of all states in D_χ that have an unsaturated or a saturated label, respectively; here, a set of formulas is *saturated* if all its elements are either modalised formulas $\heartsuit\psi$ or $\top$, and unsaturated otherwise. We extend δ to words over Σ in the usual way and to subsets of Σ_s by putting $\delta(v, A) = \{\delta(v, a) \mid a \in A\}$ for $v \in D_\chi$, $A \subseteq \Sigma_s$.

Definition A.1 (One-step satisfiability problem [32, 42, 44]). Let V be a finite set, $\Lambda(V) = \{\heartsuit a \mid a \in V, \heartsuit \in \Lambda\}$, and let $\Theta \subseteq \mathcal{P}(V)$. We interpret $a \in V$ and $\gamma \subseteq \Lambda(V)$ over Θ by

$$[\![a]\!]_0^\Theta = \{u \in \Theta \mid a \in u\} \qquad [\![\gamma]\!]_1^\Theta = \textstyle\bigcap_{\heartsuit a \in \gamma} [\![\heartsuit]\!]_\Theta [\![a]\!]_0^\Theta.$$

We refer to the data (γ, Θ) as a *one-step pair* (over V) and say that (γ, Θ) is *satisfiable* (over F) if $[\![\gamma]\!]_1^\Theta \neq \emptyset$.

Example A.2. (1) For the *relational modal μ-calculus* (Example 2.1.1.), where $\Lambda = \{\Diamond, \Box\}$, the one-step satisfiability problem is to decide, for a given one-step pair (γ, Θ) over V, whether there is $A \in [\![\gamma]\!]_1^\Theta$, that is, a subset $A \in \mathcal{P}\Theta$ such that for each $\Diamond a \in \gamma$, there is $u \in A$ such that $a \in u$, and for each $\Box b \in \gamma$ and each $u \in A$, $b \in u$. Equivalently, one needs to check that for each $\Diamond a \in \gamma$ there is $u \in \Theta$ such that $a \in u$ and moreover $b \in u$ for all $\Box b \in \gamma$.

(2) For the *graded μ-calculus* (Example 2.1.2.), the one-step satisfiability problem is to decide, for a one-step pair (γ, Θ), whether there is a multiset $\beta \in \mathcal{B}\Theta$ such that $\sum_{u \in \Theta \mid a \in u} \beta(u) > m$ for each $\langle m \rangle a \in \gamma$ and $\sum_{u \in \Theta \mid a \notin u} \beta(u) \leq m$ for each $[m]a \in \gamma$. This problem can be solved via a nondeterministic algorithm that goes through all $u \in \Theta$, guessing multiplicities $\beta(u) \in \{0, \dots, m+1\}$ where m is the greatest index of any diamond modality $\langle m \rangle$ that occurs in γ. This multiplicity is used to update $|V|$ counters that keep track of the total measure $\beta([\![a]\!]_0^\Theta)$ for $a \in V$ and then forgotten. Once all multiplicities have been guessed, the algorithm verifies that $\beta \in [\![\gamma]\!]_1^\Theta$, using only the final counter values [26, Lemma 1].

Recall (e.g. [17]) that *parity games* are history-free determined infinite-duration two-player games given by data $G = (V_\exists, V_\forall, E, v_0, \Omega)$, consisting of disjoint sets $V_\exists$ and $V_\forall$ of nodes belonging to the existential and universal player, respectively, a set $E \subseteq V \times V$ of edges (where $V = V_\exists \cup V_\forall$), an initial node v_0, and a priority function $\Omega : E \to \mathbb{N}$ that assigns priorities $\Omega(e)$ to *edges* $e \in E$. A *play* is a finite or infinite sequence $\pi = v_0, v_1, \dots$ of nodes such that $(v_i, v_{i+1}) \in E$ whenever applicable. Finite plays are required to end in nodes without outgoing moves, and then are won by the player that does not own the last node in the play; infinite plays $\pi \in V^\omega$ are won by the existential player if and only if the maximal priority visited in π is even (and by the universal player otherwise). The existential player wins the game G if she has a strategy to move at her nodes

such that she wins every play that is compatible with this strategy; otherwise, the universal player wins G.

Now we are ready to characterise satisfiability in the coalgebraic μ-calculus by parity games. Recall that $\mathsf{B}_\chi = (D_\chi, \Sigma, \delta, v_0, \Omega)$ is a deterministic parity automaton with $2nk$ priorities.

Definition A.3 (Satisfiability games). The *satisfiability game* $G_\chi = (V_\exists, V_\forall, E, v_0, \Omega)$ for χ is a parity game with sets of nodes $V_\exists = D_\chi$ and $V_\forall = D_\chi \times \mathcal{P}(\Sigma_s)$. The other components of the game are defined by the following table, where $\Omega(v,\tau)$ denotes the maximal Ω-priority that is visited by the partial run of B_χ that leads from v to $\delta(v,\tau)$ by reading τ letter by letter.

node	owner	moves to	priority
$v \in \mathsf{cores}$	$\exists$	$\{\delta(v,\tau) \in \mathsf{states} \mid \tau \in \mathsf{choices}\}$	$\Omega(v,\tau)$
$v \in \mathsf{states}$	$\exists$	$\{(v,\Xi) \in V_\forall \mid [\![l(v)]\!]_1^{l[\delta(v,\Xi)]} \neq \emptyset\}$	$\Omega(v)$
$(v,\Xi) \in V_\forall$	$\forall$	$\{\delta(v,\kappa) \mid \kappa \in \Xi\}$	0

Thus the existential player attempts to show the existence of a specific subautomaton of B_χ, intuitively by selecting, at each core node v of B_χ, some nonmodal word τ that saturates v, leading to a state node. For modal steps at state nodes v, the existential player has to provide a set Ξ of letters (selecting a set of modal out-edges of v) such that the label of v is one-step satisfiable in the labels of the selected successors of v in B_χ. The universal player then can challenge any letter $\kappa \in \Xi$ and perform the corresponding modal step by moving to $\delta(v,\kappa)$. The existential player has to make her choices in such a way that for every compound word that results from the choices, the corresponding run of B_χ is accepting (that is, does not contain infinite deferrals of least fixpoints), and never visits nodes with $\bot$ in the label.

Given a set $G \subseteq D_\chi$, we let $\mathsf{win}_G^\exists$ and $\mathsf{win}_G^\forall$ denote the winning regions for the existential and the universal player, respectively, in the partial game $G_\chi|_G$ that is obtained from G_χ by removing all nodes that are not contained in G. Nodes in this partial game $G_\chi|_G$ may be undetermined so that we in general do not have $\mathsf{win}_G^\exists \cup \mathsf{win}_G^\forall = G$. However, we do have $\mathsf{win}_{D_\chi}^\exists \cup \mathsf{win}_{D_\chi}^\forall = D_\chi$. While the number of nodes in $G_\chi|_G$ is doubly exponential in n (Σ_s is singly exponential, so $\mathcal{P}(\Sigma_s)$ is doubly exponential), we use a fixpoint computation over (subsets of) D_χ [21] to compute the sets $\mathsf{win}_G^\exists$ and $\mathsf{win}_G^\forall$ in singly exponential time.

Definition A.4 (Game solving). Given a set $G \subseteq D_\chi$, we define the *one-step solving function* $f_G : \mathcal{P}(G)^{2nk} \to \mathcal{P}(G)$ by

$$\begin{aligned} f_G(\mathbf{X}) = &\{v \in \mathsf{cores} \mid \exists \tau \in \mathsf{choices}.\, \delta(v,\tau) \in X_{\Omega(v,\tau)} \cap \mathsf{states}\} \cup \\ &\{v \in \mathsf{states} \mid [\![l(v)]\!]_1^{l[\delta(v,\Sigma_s)\cap X_{\Omega(v)}]} \neq \emptyset\} \end{aligned}$$

where $\mathbf{X} = X_1, \ldots, X_{2nk} \subseteq G$. This function intuitively encodes one step in the partial game $G_\chi|_G$ that is obtained from G_χ by removing all nodes that are not

contained in G. The winning regions in $G_\chi|_G$ then can be characterised by the nested fixpoints $\mathbf{E}_G$ and $\mathbf{A}_G$, defined by

$$\mathbf{E}_G = \eta_{2nk} X_{2nk}. \ldots \eta_1 X_1. f_G(\mathbf{X}) \qquad \mathbf{A}_G = \overline{\eta_{2nk}} X_{2nk} \ldots \overline{\eta_1} X_1. \overline{f_G}(\overline{\mathbf{X}}),$$

where $\eta_i = \mu$ for odd i, $\eta_i = \nu$ for even i, where $\overline{\nu} = \mu$ and $\overline{\mu} = \nu$, and where $\overline{f_G}(\overline{\mathbf{X}}) = D_\chi \setminus f_G(\mathbf{X})$.

We note that nodes in the partial game $G_\chi|_G$ may be undetermined so that we in general do not have $\mathbf{E}_G \cup \mathbf{A}_G = G$. However, we do have $\mathbf{E}_{D_\chi} \cup \mathbf{A}_{D_\chi} = D_\chi$.

Algorithm 1 Satisfiability checking by global caching; input: formula χ;

Initialise set of *unexpanded* nodes $U = \{v_0\}$ and *expanded* nodes $G = \emptyset$.

(1) Expansion: Choose some unexpanded node $u \in U$, remove u from U, and add u to G. Add to U all nodes in the sets $\{\delta(u,\tau) \in \mathsf{states} \mid \tau \in \mathsf{choices}\} \setminus G$ (if $u \in \mathsf{cores}$) or $\{\delta(u,\kappa) \mid \kappa \in \Sigma_s, \kappa \subseteq l(u)\} \setminus G$ (if $u \in \mathsf{states}$).

(2) Optional game solving: Compute $\mathsf{win}^\exists_G$ and/or $\mathsf{win}^\forall_G$. If $v_0 \in \mathsf{win}^\exists_G$, then return 'satisfiable', if $v_0 \in \mathsf{win}^\forall_G$, then return 'unsatisfiable'.

(3) If $U \neq \emptyset$, then continue with Step 1. Otherwise, $G = D_\chi$; continue with Step 4.

(4) Final game solving: Compute $\mathsf{win}^\exists_G$. If $v_0 \in \mathsf{win}^\exists_G$, then return 'satisfiable', otherwise return 'unsatisfiable'.

Theorem A.5 ([21]). *Existential player wins G_χ if and only if χ is satisfiable, and Algorithm 1 computes the winning regions in G_χ on-the-fly.*

(The proof given in [21] elides the game formulation and instead shows directly that the fixpoint captures satisfiability; however, equivalence of the game and the fixpoint computation is straightforward.)

A.3 Additional Details for Section 4

The omitted formulas for the parity conditions in Section 4 are defined as follows.

$$\mathsf{Parity}(n,k) = \mu X_k. \nu X_{k-1}. \ldots . \nu X_2. \mu X_1. \bigvee_{1 \leq i \leq k} p_i \wedge \langle n \rangle X_i$$

These formulas express that *'there is an $n+1$-branching tree starting here in which each path satisfies the parity condition encoded by the priorities p_i'*. This property implies that *'there is an $n+1$-branching tree such that for every path there is an even priority p_i that occurs infinitely often on the path and priorities larger than p_i occur only finitely often on the path'*:

$$\mathsf{Buechi}(n,k) = \mu X. \langle n \rangle X \vee \bigvee_{1 \leq i \leq k, i \text{ even}} \nu Y. \mu Z. \bigwedge_{j > i} \neg p_j \wedge ((p_i \wedge \langle n \rangle Y) \vee \langle n \rangle Z))$$

The formulas $\mathsf{Buechi}(f,\psi)$, $\mathsf{RabinPair}(i,f,\psi)$ and $\mathsf{Rabin}(k,\psi)$ for the rabin properties $\langle i_j, f_j\rangle_{j\leq k}$ in Section 4 are defined as follows.

$$\begin{aligned}
\mathsf{Buechi}(f,\psi) &= \nu X.\,\mu Y.\,((f\wedge\psi(X))\vee(\neg f\wedge\psi(Y)))\\
\mathsf{RabinPair}(i,f,\psi) &= \mu X.\,\nu Y.\,\mu Z.\,(f\wedge\psi(X))\vee(\neg f\wedge i\wedge\psi(Y))\vee(\neg f\wedge\neg i\wedge\psi(Z))\\
\mathsf{Rabin}(k,\psi) &= \mu X_{2k+1}.\,\mathsf{Disj}(2k,[],\psi)\\
\mathsf{Disj}(c,p,\psi) &= \bigvee_{j\notin p}\nu X_c^{p:j}.\,\mu X_{c-1}^{p:j}.\,\mathsf{Disj}(c-2,p:j,\psi)\\
\mathsf{Disj}(0,p,\psi) &= (f_{p(1)}\wedge\psi(X_{2k+1}))\vee\\
&\qquad\bigvee_{j\leq k}((\bigwedge_{j'\leq j}\neg f_{p(j')}\wedge i_{p(j)}\wedge\psi(X^{p|j}_{2(k-j)+2}))\vee\\
&\qquad\qquad(\bigwedge_{j'\leq j}\neg f_{p(j')}\wedge\neg i_{p(j)}\wedge\psi(X^{p|j}_{2(k-j)+1})))
\end{aligned}$$

Here, p is a partial permutation over $[k]=\{1,2,\ldots,k\}$, that is, p is a list of elements of $[k]$ without duplicates; the empty permutation is denoted by $[]$. We write $j\notin p$ to denote the fact that $j\in[k]$ does not occur in the partial permutation p. By $p:j$ we denote the partial permutation that is obtained by concatenating p and j (that is, by appending j to the end of the list p). Finally, $p|i$ denotes the permutation that is obtained from p by keeping just the first i entries, and $p(j)$ denotes the j-th entry in p.

The full formulas for Büchi and Rabin games mentioned in Section 4 are as follows.

The formula $\theta=\mathsf{AG}((v_\exists\wedge\neg v_\forall)\vee(\neg v_\exists\wedge v_\forall))$ expresses that every node belongs to exactly one of the two players $v_\exists$ or $v_\forall$. We also introduce graded formulas $\mathsf{Cpre}^n_i(X)$ stating that player $i\in\{\exists,\forall\}$ can ensure that X is reached in the next step with at least (all but at most) n moves:

$$\mathsf{Cpre}^n_\exists(X)=(v_\exists\wedge\langle n\rangle X)\vee(v_\forall\wedge[n]X)\qquad\mathsf{Cpre}^n_\forall(X)=(v_\forall\wedge\langle n\rangle X)\vee(v_\exists\wedge[n]X)$$

The winning regions in graded Büchi and Rabin games then are defined by

$$\mathsf{BuechiG}(f,n)=\theta\wedge\mathsf{Buechi}(f,\mathsf{Cpre}^n_\exists)\qquad\mathsf{RabinG}(k,n)=\theta\wedge\mathsf{Rabin}(k,\mathsf{Cpre}^n_\exists)$$

The comparison between TATL and COOL 2 is based on the following formulas taken from [6]:

(1) p
(2) $p\wedge q$
(3) $p\vee q$
(4) $p\rightarrow q$
(5) $\langle\!\langle 1\rangle\!\rangle Xp$
(6) $\langle\!\langle 1\rangle\!\rangle Fp$
(7) $\langle\!\langle 1\rangle\!\rangle Gp$
(8) $\langle\!\langle 1\rangle\!\rangle pUq$
(9) $\neg\langle\!\langle 1\rangle\!\rangle pUq$

(10) $\neg\langle\langle 1\rangle\rangle Fp$
(11) $\langle\langle 1,2\rangle\rangle pUq \wedge \langle\langle 1,2\rangle\rangle Xr$
(12) $\langle\langle 1,2\rangle\rangle pUq \wedge \langle\langle 3,4\rangle\rangle Xr$
(13) $\langle\langle 1,2\rangle\rangle pUq \wedge \langle\langle 2,3\rangle\rangle Xr$
(14) $\langle\langle 2,1\rangle\rangle pUq \wedge \langle\langle 3,2\rangle\rangle Xr$
(15) $\langle\langle\rangle\rangle pUq \wedge \langle\langle 1,2\rangle\rangle Xr$
(16) $\neg\langle\langle 1,2\rangle\rangle Xp \wedge \langle\langle 1\rangle\rangle Gp$
(17) $\neg\langle\langle 1,2\rangle\rangle Xp \wedge \langle\langle 1,2,3\rangle\rangle Gp$
(18) $\neg p \vee \langle\langle 1\rangle\rangle Fp$
(19) $p \wedge \neg p$
(20) $(p \wedge q) \wedge \langle\langle 1\rangle\rangle G\neg(p \wedge q)$
(21) $\langle\langle 1\rangle\rangle Gp \wedge \neg\langle\langle 2\rangle\rangle F\langle\langle 1\rangle\rangle Gp$
(22) $\langle\langle 1\rangle\rangle Xp \wedge \neg\langle\langle 1\rangle\rangle Xp$
(23) $\langle\langle 1\rangle\rangle pUq \vee \neg\langle\langle 1\rangle\rangle Gq$
(24) $\langle\langle 1,2\rangle\rangle pU(\neg\langle\langle 1\rangle\rangle Gp)$
(25) $\langle\langle 1\rangle\rangle(\neg\langle\langle 1,2\rangle\rangle Gp)Uq$
(26) $\langle\langle\rangle\rangle G\langle\langle\rangle\rangle pUq$
(27) $\neg\langle\langle 1\rangle\rangle Gp \wedge \langle\langle 1,2\rangle\rangle Xp \wedge \neg\langle\langle 2\rangle\rangle X\neg p$
(28) $\langle\langle 1\rangle\rangle Xp \wedge \langle\langle 2\rangle\rangle Xq \wedge \langle\langle 1,2\rangle\rangle Xr \wedge \neg\langle\langle 1\rangle\rangle Xr \wedge \neg\langle\langle 3\rangle\rangle Xq$
(29) $\neg\langle\langle 1\rangle\rangle Xr \wedge \neg\langle\langle 3\rangle\rangle Xq \wedge \langle\langle 1\rangle\rangle Xp \wedge \langle\langle 2\rangle\rangle Xq \wedge \langle\langle 1,2\rangle\rangle Xr$
(30) $\neg\langle\langle 1\rangle\rangle Xr \wedge \langle\langle 1\rangle\rangle Xp \wedge \langle\langle 2\rangle\rangle Xq \wedge \neg\langle\langle 3\rangle\rangle Xq \wedge \langle\langle 1,2\rangle\rangle Xr$
(31) $\langle\langle 1,2,3\rangle\rangle G\langle\langle 2,3,4\rangle\rangle G(p \wedge q)$
(32) $\langle\langle 1,2,3\rangle\rangle G\langle\langle 2,3\rangle\rangle G(p \wedge q) \wedge \langle\langle 4\rangle\rangle X\neg p$
(33) $\neg\neg\langle\langle 1\rangle\rangle pUq$
(34) $\neg(\langle\langle 1\rangle\rangle Gp \vee \langle\langle 1\rangle\rangle G\neg p)$
(35) $\neg(\langle\langle 1\rangle\rangle Gp \wedge \langle\langle 1\rangle\rangle G\neg p)$
(36) $\neg\langle\langle 1\rangle\rangle pU\neg\langle\langle 2\rangle\rangle qUr$
(37) $\langle\langle 1\rangle\rangle G\neg q \wedge \langle\langle 2\rangle\rangle pUq$
(38) $\langle\langle 1\rangle\rangle Gp \wedge \neg\langle\langle 1,2\rangle\rangle Gp$
(39) $\neg\langle\langle 1\rangle\rangle Xp \wedge \langle\langle 2\rangle\rangle X\neg p$
(40) $\langle\langle 1\rangle\rangle Xp \wedge \langle\langle 2\rangle\rangle X\neg p$
(41) $\langle\langle 1\rangle\rangle pUq \wedge \langle\langle 2\rangle\rangle qUr \wedge \langle\langle 2\rangle\rangle G\neg r$
(42) $\langle\langle 1\rangle\rangle pUq \wedge \langle\langle 2\rangle\rangle qUr \wedge \langle\langle 1\rangle\rangle G\neg r$

The results of the comparison between COOL 2 and TATL on these formulas are shown in Figure 6.

Furthermore, we turn formula 36, that is, the formula $\neg\langle\langle 1\rangle\rangle p \,\mathsf{U}\, \neg\langle\langle 2\rangle\rangle q \,\mathsf{U}\, r$ into a formula series with increasing number of nested temporal operators, defined

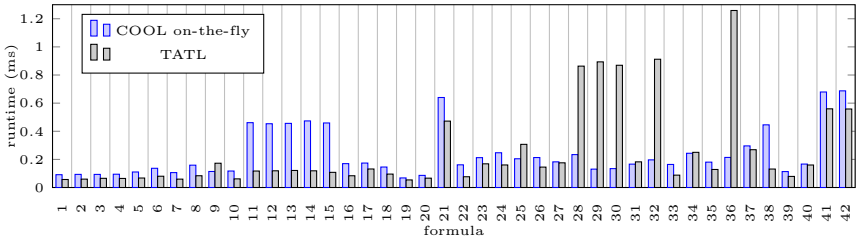

Fig. 6. Runtimes on selected ATL formulas [6]

inductively by

$$\psi(0) = \neg\langle\langle 2\rangle\rangle q \,\mathsf{U}\, r$$
$$\psi(i+1) = \neg\langle\langle (i \bmod 2) + 1\rangle\rangle p_{i \bmod 2} \,\mathsf{U}\, \psi(i) \qquad (i > 0)$$

The according experiment results depicted in Figure 7 show that COOL 2 outperforms TATL on this formula series by a large margin, with runtime for COOL 2 remaining almost constant as n increases while TATL quickly runs into timeouts of 60 seconds. This presumably is due to the nesting of temporal operators.

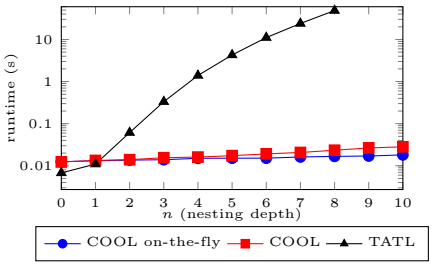

Fig. 7. Runtimes for $\psi(n)$