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

Oliver Görlitz[1], Daniel Hausmann[2], Merlin Humml[1(✉)], Dirk Pattinson[3], Simon Prucker[1], and Lutz Schröder[1]

[1] Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, Germany
merlin.humml@fau.de
[2] Gothenburg University, Gothenburg, 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 [1,36], monotone neighbourhoods structures as in game logic [34] and concurrent dynamic logic [37], 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* [38]. It provides reasoning algorithms of optimal complexity at various levels of expressiveness, up to the coalgebraic

D. Hausmann—Supported by the ERC Consolidator grant D-SynMA (No. 772459)
M. Humml—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).
L. Schröder—Supported by Deutsche Forschungsgemeinschaft (DFG) under project no. 419850228.

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 [41] 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 [41]; 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 full version [17].

Related Work: We have already mentioned work in coalgebraic logic on which COOL is based [3,13,21–23,41]. 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++ [44], Pellet [42], 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 [43], 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 [32]. 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* [41]; 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. [21,29,33]) 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 [35,40]. 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) [39] extends coalition logic [36] 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 $N = \{1, \dots, n\}$ 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 [17,21].

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* [41]; 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 [41,43] but has since turned out to be incomplete; we give a counterexample in the full version [17]. (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 Sect. 2, treating modal steps by solving instances of the one-step satisfiability problem[1]. 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

[1] 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).[2] Complete definitions of all formula series used in the evaluation as well as additional experimental results can be found in the full version [17].

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. [43].

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

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

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:

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

- 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:

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

  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 unsatisfiable.

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

– 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 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:

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

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

– 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:

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

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[3]. Below, 'COOL' and 'COOL on-the-fly' refer to invoking COOL 2 with solving rate once and adaptive, respectively.

Results for the Cardinality and CardinalityU series are shown in Fig. 1 and Fig. 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.

Results for the unsatisfiable tree property are shown in Fig. 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.

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

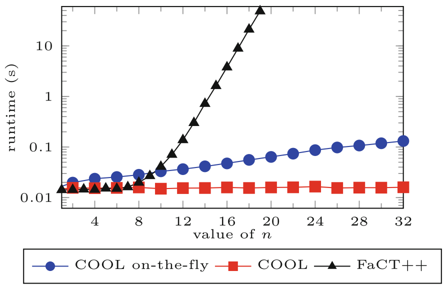

Fig. 1. Runtimes for Cardinality(n)

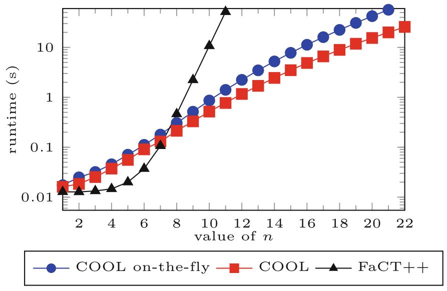

Fig. 2. Runtimes for CardinalityU(n)

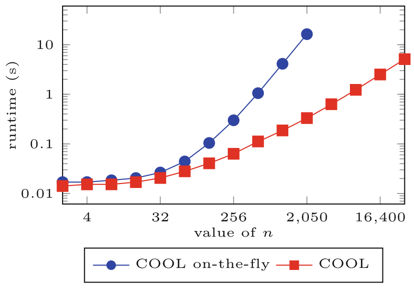

Fig. 3. Runtimes for TreeU(n)

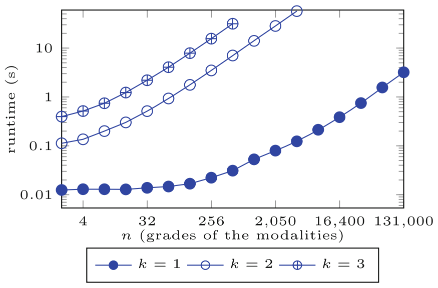

Fig. 4. Runtimes for ¬ParityToBuechi(n, k)

Runtimes for COOL 2 (using on-the-fly solving) on the unsatisfiable series of parity formulas ¬ParityToBuechi(n, k) are shown in Fig. 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 s. 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 Fig. 5 which shows the stepped execution times for the series Nest that alternates between being satisfiable and unsatisfiable

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

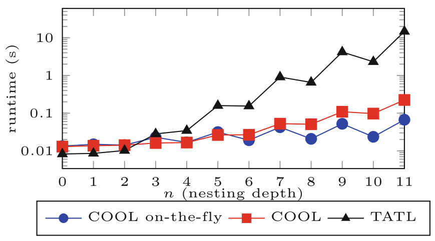

Fig. 5. Runtimes for the ATL series $\mathsf{Nest}(n)$

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 $\neg\mathsf{ParityToBuechi}$ 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.

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: Galmiche, D., Larchey-Wendling, D. (eds.) TABLEAUX 2013. LNCS (LNAI), vol. 8123, pp. 97–103. Springer, Heidelberg (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.) CADE 2015. LNCS (LNAI), vol. 9195, pp. 214–228. Springer, 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.) SAT 2003. LNCS, vol. 2919, pp. 502–518. Springer, Heidelberg (2004). 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. Technical report, 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: Giesl, J., Hähnle, R. (eds.) IJCAR 2010. LNCS (LNAI), vol. 6173, pp. 46–60. Springer, Heidelberg (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: Demri, S., Kapur, D., Weidenbach, C. (eds.) IJCAR 2014. LNCS (LNAI), vol. 8562, pp. 396–402. Springer, Cham (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. Görlitz, O., Hausmann, D., Humml, M., Pattinson, D., Prucker, S., Schröder, L.: Cool 2 - a generic reasoner for modal fixpoint logics (2023). https://arxiv.org/abs/2305.11015
18. Haarslev, V., Möller, R.: RACER system description. In: Goré, R., Leitsch, A., Nipkow, T. (eds.) IJCAR 2001. LNCS, vol. 2083, pp. 701–705. Springer, Heidelberg (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: Madeira, A., Benevides, M. (eds.) DALI 2017. LNCS, vol. 10669, pp. 115–132. Springer, Cham (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: Bojańczyk, M., Simpson, A. (eds.) FoSSaCS 2019. LNCS, vol. 11425, pp. 277–294. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17127-8_16
22. Hausmann, D., Schröder, L., Deifel, H.-P.: Permutation games for the weakly aconjunctive μ-calculus. In: Beyer, D., Huisman, M. (eds.) TACAS 2018. LNCS, vol. 10806, pp. 361–378. Springer, Cham (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.Y.: Fair equivalence relations. In: Dershowitz, N. (ed.) Verification: Theory and Practice. LNCS, vol. 2772, pp. 702–732. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-39910-0_30
26. Kupferman, O., Sattler, U., Vardi, M.Y.: The complexity of the graded μ-calculus. In: Voronkov, A. (ed.) CADE 2002. LNCS (LNAI), vol. 2392, pp. 423–437. Springer, Heidelberg (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. 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
33. Niwiński, D.: On fixed-point clones. In: Kott, L. (ed.) ICALP 1986. LNCS, vol. 226, pp. 464–473. Springer, Heidelberg (1986). https://doi.org/10.1007/3-540-16761-7_96
34. Parikh, R.: Propositional game logic. In: Foundations of Computer Science, FOCS 1983. IEEE Computer Society (1983). https://doi.org/10.1109/SFCS.1983.47
35. 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
36. 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
37. Peleg, D.: Concurrent dynamic logic. J. ACM **34**, 450–479 (1987). https://doi.org/10.1145/23005.23008
38. 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
39. Schewe, S.: Synthesis of distributed systems. Ph.D. thesis, Universität des Saarlands (2008)

40. 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
41. 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
42. 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
43. Snell, W., Pattinson, D., Widmann, F.: Solving graded/probabilistic modal logic via linear inequalities (system description). In: Bjørner, N., Voronkov, A. (eds.) LPAR 2012. LNCS, vol. 7180, pp. 383–390. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28717-6_30
44. Tsarkov, D., Horrocks, I.: FaCT++ description logic reasoner: system description. In: Furbach, U., Shankar, N. (eds.) IJCAR 2006. LNCS (LNAI), vol. 4130, pp. 292–297. Springer, Heidelberg (2006). https://doi.org/10.1007/11814771_26