COOL – A Generic Reasoner For Coalgebraic Hybrid Logics (System Description)

Daniel Gorín[1], Dirk Pattinson[2], Lutz Schröder[1⋆], Florian Widmann[3], and Thorsten Wißmann[1]

[1] Friedrich-Alexander-Universität Erlangen-Nürnberg
[2] The Australian National University
[3] Imperial College London

Abstract. We describe the Coalgebraic Ontology Logic solver COOL, a generic reasoner that decides the satisfiability of modal (and, more generally, hybrid) formulas with respect to a set of global assumptions – in Description Logic parlance, we support a general TBox and internalize a Boolean ABox. The level of generality is that of coalgebraic logic, a logical framework covering a wide range of modal logics, beyond relational semantics. The core of COOL is an efficient unlabelled tableaux search procedure using global caching. Concrete logics are added by implemening the corresponding (one-step) tableaux rules. The logics covered at the moment include standard relational examples as well as graded modal logic and Pauly's Coalition Logic (the next-step fragment of Alternating-time Temporal Logic), plus every logic that arises as a fusion of the above. We compare the performance of COOL with state-of-the-art reasoners.

1 Introduction

Many modal logics can be interpreted using a Kripke-style relational semantics, but there is a vast array of modal logics that cannot be captured using relational models. Examples include classical and monotone modal logics [5], coalition logic / alternating-time logic [18, 1], and probabilistic modal logic [10]. Graded modal logic [11] was originally formulated as a relational logic but is more naturally seen as talking about weighted graphs [7]. Semantically, these logics are captured using *coalgebraic logic* [17], a unifying framework that systematizes semantics, meta-theory and algorithms. Reasoning algorithms harness the syntactical presentation of these logics in terms of *one-step rules*. In tableaux presentation, these rules have the form

$$\frac{\Gamma_0}{\Gamma_1 \mid \cdots \mid \Gamma_n}$$

where $\Gamma_1, \ldots, \Gamma_n$ are sets of literals (read conjunctively) over a set V of propositional variables, and Γ_0 is set of literals over $\Lambda(V) = \{\heartsuit a \mid \heartsuit \in \Lambda, a \in V\}$ where Λ is an abstract set of modal operators. The reading of these rules is standard: to show satisfiability of the premiss, one needs to establish satisfiability of one conclusion, for every applicable rule.

⋆ Work of the first, third, and fifth author supported by DFG grant GenMod2 (SCHR 1118/5-2)

Example 1. 1. *Modal Logic K.* A simple one-conclusion example is the standard (unlabeled) tableau rule $\Box a_1, \ldots, \Box a_n, \Diamond b / a_1, \ldots, a_n, b$ for the modal logic K, over modal operators $\Lambda = \{\Box, \Diamond\}$.

2. *Coalition logic:* Pauly's coalition logic [18], or the next-step fragment of alternating-time temporal logic ATL [1], is parametrized by a set $N = \{1, \ldots, n\}$ of *agents*; subsets of N are called *coalitions*. Operators are of the form $[C]$, 'coalition C can force', with duals $\langle C\rangle$ 'coalition C cannot avoid'. In the terminology of [1], the semantics is based on *concurrent game frames*. A complete set of rules is given by

$$\frac{[C_1]a_1, \ldots, [C_n]a_n}{a_1, \ldots, a_n} \qquad \frac{[C_1]a_1, \ldots, [C_n]a_n, \langle D\rangle b, \langle N\rangle c_1, \ldots, \langle N\rangle c_m}{a_1, \ldots, a_n, b, c_1, \ldots, c_n}$$

where $n, m \geq 0$ and the C_i are disjoint and contained in D [18, 19, 12].

3. *Graded logic:* Fine's graded modal logic [11] counts successor states in relational models; it has found its way into modern description logics [3] in the shape of *qualified number restrictions*. Its operators are $\Diamond_k$, read 'in more than k successors', with duals $\Box_k$ 'in all but at most k successors'. A complete set of rules [19, 6] is given by

$$\frac{\Diamond_{k_1} a_1, \ldots, \Diamond_{k_n} a_n, \Box_{l_1} b_1, \ldots, \Box_{l_m} b_m}{\sum_{1\leq i\leq n} r_i a_i - \sum_{1\leq j\leq m} s_j(\neg q_j) > 0} \quad (\textstyle\sum_{1\leq i\leq n} r_i(k_i+1) - \sum_{1\leq j\leq m} s_j l_j \geq 1)$$

where $n, m \geq 0$ and $r_i, s_j > 0$, subject to the side condition indicated, and with the sums in the conclusion of the rule referring to arithmetic of characteristic functions, i.e. counting 1 for 'true' and 0 for 'false'. Sufficient tractability of this rule is shown using results from integer linear programming [19].

The one-step rules are combined with propositional rules such as $\Gamma, a \vee b / \Gamma, a \mid \Gamma, b$ and rules that deal with nominal and satisfaction operators. One of the crucial feature of these logics is *compositionality*: the restriction on the rule format allows us to synthesize logics in a modular fashion. This is best understood by thinking of the one-step tableau rules as building blocks for logics that de-construct modal operators of a given type.

Sequencing of Logics. To describe, e.g. simple Segala systems [20] that describe systems that perform non-deterministic actions followed by a (probabilistic) action of the environment, we use a two-sorted syntax

$$L_0 \ni \phi ::= p_0 \mid \neg\phi \mid \phi \wedge \phi \mid \Diamond_a \psi \qquad L_1 \ni \psi ::= p_1 \mid \neg\psi \mid \psi \wedge \psi \mid \langle p\rangle\phi$$

where p_i is a typed propositional variable of the language L_i and $\langle p\rangle$ is an exemplaric operator of probabilistic modal logic 'with probability at least p'. To show satisfiability of $\phi \in L_0$ we deconstruct propositional connectives and apply tableaux rules for Hennessy-Milner logic. This leaves us with formulae in L_1 that are deconstructed in the same way, but using the rules of propositional modal logic, recursively yielding a formula in L_0 to which the same procedure is applied.

Fusion of Logics. To ensure the same typing discipline we describe the fusion of two logics over modal operators Λ_1 and Λ_2 in the same typed framework using three sorts and two new operators $[\pi_1]$ and $[\pi_2]$:

$$L_0 \ni \phi ::= p_0 \mid \neg\phi \mid \phi \wedge \phi \mid [\pi_1]\psi \mid [\pi_2]\sigma$$
$$L_1 \ni \psi ::= p_1 \mid \neg\psi \mid \psi \wedge \psi \mid \heartsuit_1\phi \qquad L_2 \ni \sigma ::= p_2 \mid \neg\sigma \mid \sigma \wedge \sigma \mid \heartsuit_2\phi$$

where $\heartsuit_i \in \Lambda_i$ is an operator of type i, and p_i is a propositional variable of type i. It is straightforward to embed the standard (language of the) fusion into the language L_0. We have two modal operators together with the tableau rules

$$\frac{\neg[\pi_i]a_1, \ldots, \neg[\pi_i]a_i, [\pi_i]b_1, \ldots, [\pi_i]b_k}{\neg a_1, \ldots, \neg a_n, b_1, \ldots, b_k}$$

for $i = 1, 2$. This allows us to reason about fusion using the same, typed, reasoning algorithm as described above for sequencing.

Choice. Choice allows us to axiomatise, e.g., the alternating systems of Hansson and Jonsson [16] where a successor state either originates from a labelled transition, or from a probabilistic action of the environment. Like fusion, we describe choice by means of a multi-sorted language that introduces one new modal operator, $+$, described by a one-step tableau rule. For alternating systems, we have

$$L_0 \ni \phi ::= p_0 \mid \neg\phi \mid \phi \wedge \phi \mid \psi + \sigma$$
$$L_1 \ni \psi ::= p_1 \mid \neg\psi \mid \psi \wedge \psi \mid \heartsuit_1\phi \qquad L_2 \ni \sigma ::= p_2 \mid \neg\sigma \mid \sigma \wedge \sigma \mid \heartsuit_2\phi$$

where we read the binary operator $\psi + \sigma$ as 'ψ for labelled successors and σ for probabilistic ones'. Reasoning over logics defined using choice is governed by the rules

$$\frac{\neg(a_1 + c_1), \ldots, \neg(a_n + c_n), (b_1 + d_1), \ldots, (b_k + d_k)}{\neg a_1, \ldots, \neg a_n, b_1, \ldots, b_n \mid \neg c_1, \ldots, \neg c_n, d_1, \ldots, d_k}$$

that induce type-correct formulae of sort L_1 (left) and L_2 (right).

A range of generic reasoning procedures of optimal complexity has been developed for coalgebraic logics with various additional features, including global assumptions, nominals, and fixpoints that all support modular combinations as described. The most basic of these, the generic PSPACE algorithm for satisfiability in next-step logics [19], has been implemented (already supporting modularity) in the CoLoSS tool [4]. Here, we present the *Coalgebraic Ontology Logic* Reasoner (COOL), available at https://www8.cs.fau.de/research/cool, which supports modular combinations of logics, global assumptions, and nominals, and uses global caching.

2 The COOL Solver: Supported Features

COOL implements a global caching algorithm for coalgebraic hybrid logic with global assumptions [13]. In description logic parlance, we support terminological reasoning (TBoxes) as well as nominals and satisfaction operators (thus internalizing Boolean ABoxes in concepts). These features are orthogonal to the underlying base logic which is constructed in a modular way from a number of basic building blocks, and the effort of adding a new logic is typically quite limited. Global caching combines theoretical optimality (i.e. an exponential time upper bound) with amenability to heuristic optimization [15]. In more detail, COOL supports the following.

– *Global assumptions*, or, in description logic parlance, a *general TBox*: one can restrict the class of models to ones in which all states/worlds/individuals satisfy a given finite set of formulas, the global assumptions. In knowledge representation, such global assumptions serve to express background knowledge about the terminological domain.

– *Nominals:* we incorporate two key features of hybrid logic [2], *nominals* and *satisfaction operators*. Here, a nominal is a name i for an individual state in the model; as a formula, i is satisfied precisely in the unique state named by i. The satisfaction operator $@_i$ lets the evaluation point of a formula jump to i. Reasoning with these features encompasses DL-style ABox reasoning: recall that an *ABox* (*assertional box*) contains statements of the forms $\phi(i)$ or $R(i, j)$, respectively read 'individual i satisfies formula ϕ' and 'individuals i and j are in relation R'. In hybrid logic, these statements can be expressed as $@_i\phi$ and $@_i\Diamond_R j$, respectively.

For reasoning with these features, we use the global caching algorithm introduced in [13]. Global caching for relational modal logics (phrased in DL terminology) goes back to [14]; the principle has been generalized to coalgebraic logic in [12]. The basic idea of global caching is to regard a tableau as a directed (possibly cyclic) graph rather than a tree, thus enabling sharing of nodes. This allows one to visit each label (i.e. finite set of subformulas) at most once, ensuring at most exponential (hence in most cases asymptotically optimal) run time. The algorithm partitions the set of currently created tableau nodes into *unexpanded* (X), *undecided* (U), *satisfiable* (E), and *unsatisfiable* (A) nodes. It consists in applying the following two types of steps in near-arbitrary sequence, until either the root node is marked A or E or no further steps are applicable:

– *Expand:* Apply all matching rules to an unexpanded node (then moved from X to U), creating either new successor nodes (initially marked X) or links to existing nodes.

– *Propagate:* Mark expanded nodes as unsatisfiable if there is a matching tableau rule with only unsatisfiable conclusions, and as satisfiable if all matching rules have some satisfiable conclusion. Here, the recursion is understood as a *least* fixed point for unsatisfiability, and as a *greatest* fixed point for satisfiability.

After the final propagation step, all nodes marked U are reported as satisfiable. Note that the algorithm may leave nodes marked *X*, thus allowing for quick answers in many cases; the apparent non-determinism works in favour of the implementer, as *any* terminating sequence of steps will yield a correct result, thus leaving room for heuristics.

The novel global caching algorithm for coalgebraic hybrid logic [13] deals with the global demands arising from satisfaction operators ($@_i\phi$ holds everywhere or nowhere) by means of a dedicated second type of nodes called @-*constraints*. An @-constraint records @-formulas to be satisfied for a given standard node to be satisfiable. It is linked to standard nodes in a new type of step called @-*expansion* (essentially, having $@_i\phi$ in an @-constraint requires a standard node satisfying i and ϕ). In @-*propagation* steps, the @-constraints are updated throughout the model using greatest fixed points, essentially following a winning strategy of the player advocating satisfiability.

3 The COOL Solver: Implementation Details

The implementation of COOL focuses on modal rules, and uses the minisat sat-solver [9] for reasoning in classical propositional logic, more precisely for expanding the propositional part of nodes in the tableau graph. The SAT solver is used as a black box, and no optimisations that concern propositional reasoning are implemented on top of those performed by minisat. Rules for graded and probabilistic modal logics are generated using

the GNU Linear Programming Toolkit [4]. We refer to [21] for the details of generating rules for propositional and graded logics on the basis of linear inequalities.

Compositionality of Logics. The underlying modal logic (semantically: the branching type of systems) is described using an algebraic term with one free variable S (for state) where each base logic is represented by a unary function symbol and two binary ones for choice and fusion. These terms are best read as equalities, and e.g. the alternating systems of Hansson and Jonsson mentioned in the introduction can be specified by

$$S = \mathsf{Ch}(\mathsf{HM}(S), \mathsf{P}(S))$$

where Ch represents choice, HM represents Hennessy-Milner logic (multi-modal K) and P is probabilistic modal logic. Semantically, this expression defines the system type: given a state, we either see labelled successors, or a probability distribution over states, i.e. we observe an external choice between both. Similarly, simple Segala systems are modelled by $S = \mathsf{HM}(\mathsf{P}(S))$ and the fusion of probabilistic modal logic and Hennessy-Milner logic would be $S = \mathsf{Fus}(\mathsf{HM}(S)), \mathsf{P}(S))$ where Fus is a binary function representing fusion.

Generic Reasoning and Tableau Rules. Our reasoner is *generic* in that the reasoning algorithm is conceptually independent of the underlying modal logic. This is achieved by isolating the modal rules into OCAML functions that – given a premiss – compute the set of all (instances of) applicable tableau rules The underlying reasoning algorithm then invokes the respective rules in sequence, following the construction of any given particular logic. The treatment of nominals, satisfaction operators and global assumptions is identical for all logics, and is hard-coded into the reasoning algorithm.

Optimisations. As mentioned above, we do not implement any optimisations on the propositional level, but we use global caching for dealing both with nominals and modal tableaux. The only conceptual implementation supported by COOL (and implemented for K) is backjumping [3]: each logical feature provides a hook by which a subset of literals that cause a clash can be passed back to the reasoner.

4 Experimental Evaluation

The COOL reasoner is still on its early stages of development and, having finished a robust implementation of the generic core, our main focus at the moment is on adding support for more logics. Still, our measurements suggest that it already offers a fair performance: even for the logic K (that is, the baseline logic $\mathcal{ALC}$ of the DL community), the response times of COOL are often within those of a long-established DL reasoner, implementing many advanced optimization strategies, such as FACT++.[22]

The experiments we report here are based on random formula generation in clausal form. This methodology allows one to compare different reasoners without risking an *optimization bias* (testing w.r.t. a set of problems for which one of the reasoners was specifically tuned) and give reasonable expectations regarding the capacity of a reasoner to handle large problems. We acknowledge that on real data-sets the difference in

[4] http://www.gnu.org/s/glpk/

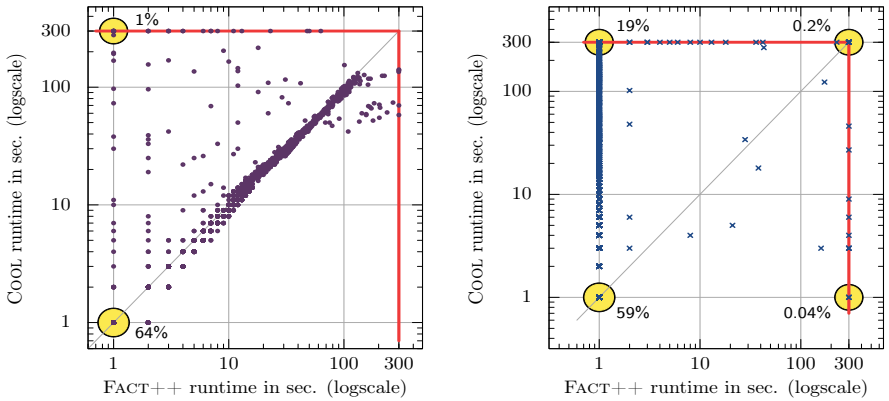

Fig. 1. Comparative evaluation of COOL vs. FACT++ on random formulas in $\mathcal{ALC}$ and $\mathcal{ALCQ}$, with 5 atoms and 0.25 chance of occurring; up to 6 dis-/conjuncts per dis-/conjunction; TBox formulas with modal depth of 2. Percentages shown refer to percentage of samples represented by indicated points. Times correspond to the USER TIME field as reported by the GNU TIME command. Test conducted on a heterogeneous cluster of computers with similar load.

performance between reasoners can become more noticeable and that random testing may oversample trivial formulas, but we defer alternative measurements (benchmarking with dedicated formula series) until more substantial sets of benchmark formulas are available also for non-relational logics.

We report on three comparisons: i) COOL vs. FACT++ on $\mathcal{ALC}$ (with a TBox), ii) COOL vs. FACT++ on $\mathcal{ALCQ}$ (with and without a TBox), and iii) COOL vs. TATL (a tableau reasoner for full ATL [8]) on coalition logic (with and without a TBox, encoded using ATL temporal operators for TATL). We kept fixed a number of parameters such as number of atoms, average number of conjuncts/disjuncts etc., and gradually increased the modal depth. In coalition logic, COOL answered consistently and substantially faster and with fewer timeouts than TATL, especially in the presence of a TBox (a scatter plot of the comparison reveals no additional information). Scatter plots for COOL vs. FACT++ are shown in Fig. 1. On $\mathcal{ALC}$, COOL shows a behaviour comparable to that of FACT++. Contrastingly, FACT++ is still substantially faster on $\mathcal{ALCQ}$, possibly due to the fact that COOL does not yet implement backjumping for $\mathcal{ALCQ}$.

5 Conclusions

Based on generic results from coalgebraic logic, the COOL reasoner supports a simple implementation and automatic combination of a wide spectrum of logics; very few other reasoners support any logic outside the standard relational setup. A preliminary empirical evaluation suggests that, while there is still plenty of room for optimizations, the implementation of the core global-caching algorithm is robust and efficient.

Acknowledgments The authors wish to thank Erwin R. Catesbeiana for helpful comments on consistency checking.

References

1. R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *J. ACM*, 49:672–713, 2002.
2. C. Areces and B. ten Cate. Hybrid logics. In P. Blackburn, J. van Benthem, and F. Wolter, eds., *Handbook of Modal Logic*, pp. 821–868. Elsevier, 2007.
3. F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, eds. *The Description Logic Handbook*. Cambridge University Press, 2003.
4. G. Calin, R. Myers, D. Pattinson, and L. Schröder. Coloss: The coalgebraic logic satisfiability solver. In *Methods for Modalities, M4M-5*, vol. 231 of *ENTCS*, pp. 41–54. Elsevier, 2009.
5. B. Chellas. *Modal Logic*. Cambridge University Press, 1980.
6. C. Cîrstea, C. Kupke, and D. Pattinson. EXPTIME tableaux for the coalgebraic μ-calculus. In *Computer Science Logic, CSL 09*, vol. 5771 of *LNCS*, pp. 179–193. Springer, 2009.
7. G. D'Agostino and A. Visser. Finality regained: A coalgebraic study of Scott-sets and multisets. *Arch. Math. Logic*, 41:267–298, 2002.
8. A. David. TATL: Implementation of ATL tableau-based decision procedure. In *Automated Reasoning with Analytic Tableaux and Related Methods, TABLEAUX 2013*, vol. 8123 of *LNCS*, pp. 97–103. Springer, 2013.
9. N. Eén and N. Sörensson. An extensible SAT-solver. In *Theory and Applications of Satisfiability Testing, SAT 2003*, vol. 2919 of *LNCS*, pp. 502–518. Springer, 2004.
10. R. Fagin and J. Y. Halpern. Reasoning about knowledge and probability. *J. ACM*, 41:340–367, 1994.
11. K. Fine. In so many possible worlds. *Notre Dame J. Formal Logic*, 13:516–520, 1972.
12. R. Goré, C. Kupke, and D. Pattinson. Optimal tableau algorithms for coalgebraic logics. In *Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2010*, vol. 6015 of *LNCS*, pp. 114–128. Springer, 2010.
13. R. Goré, C. Kupke, D. Pattinson, and L. Schröder. Global caching for coalgebraic description logics. In *Automated Reasoning, IJCAR 2010*, vol. 6173 of *LNCS*, pp. 46–60. Springer, 2010.
14. R. Goré and L. Nguyen. EXPTIME tableaux for $\mathcal{ALC}$ using sound global caching. In *Description Logics, DL 2007*, vol. 250 of *CEUR Workshop Proceedings*, 2007.
15. R. Goré and L. Postniece. An experimental evaluation of global caching for $\mathcal{ALC}$ (system description). In *Automated Reasoning, IJCAR 2008*, vol. 5195 of *LNCS*, pp. 299–305. Springer, 2008.
16. H. Hansson and B. Jonsson. A calculus for communicating systems with time and probabilities. In *Real-Time Systems, RTSS 1990*, pp. 278–287. IEEE Computer Society, 1990.
17. D. Pattinson. Coalgebraic modal logic: Soundness, completeness and decidability of local consequence. *Theoret. Comput. Sci.*, 309:177–193, 2003.
18. M. Pauly. A modal logic for coalitional power in games. *J. Log. Comput.*, 12:149–166, 2002.
19. L. Schröder and D. Pattinson. PSPACE bounds for rank-1 modal logics. *ACM Trans. Comput. Log.*, 10:13:1–13:33, 2009.
20. R. Segala. *Modelling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, Massachusetts Institute of Technology, 1995.
21. W. Snell, D. Pattinson, and F. Widmann. Solving graded/probabilistic modal logic via linear inequalities (system description). In *Logic Programming and Automated Reasoning, LPAR 2012*, vol. 7180 of *LNCS*, pp. 383–390. Springer, 2012.
22. D. Tsarkov and I. Horrocks. FaCT++ description logic reasoner: System description. In U. Furbach and N. Shankar, eds., *Int. Joint Conf. on Automated Reasoning, IJCAR 2006*, vol. 4130 of *LNAI*, pp. 292–297. Springer, 2006.