Global Caching for the Alternation-free μ-Calculus

Daniel Hausmann[1], Lutz Schröder[2], and Christoph Egger[3]

1 **Friedrich-Alexander Universität Erlangen-Nürnberg, Germany**
2 **Friedrich-Alexander Universität Erlangen-Nürnberg, Germany**
3 **Friedrich-Alexander Universität Erlangen-Nürnberg, Germany**

Abstract

We present a sound, complete, and optimal single-pass tableau algorithm for the alternation-free μ-calculus. The algorithm supports global caching with intermediate propagation and runs in time $2^{\mathcal{O}(n)}$. In game-theoretic terms, our algorithm integrates the steps for constructing and solving the Büchi game arising from the input tableau into a single procedure; this is done on-the-fly, i.e. may terminate before the game has been fully constructed. This suggests a slogan to the effect that *global caching = game solving on-the-fly*. A prototypical implementation shows promising initial results.

1998 ACM Subject Classification F.4.1 Mathematical Logic - Temporal Logic

Keywords and phrases modal logic, fixpoint logic, satisfiability, global caching, coalgebraic logic

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2016.34

1 Introduction

The modal μ-calculus [25, 3] serves as an expressive temporal logic for the specification of sequential and concurrent systems containing many standard formalisms such as linear time temporal logic LTL [28, 33], CTL [7], and PDL [34]. Satisfiability checking in the modal μ-calculus is ExpTime-complete [31, 10]. There appears to be, to date, no readily implementable reasoning algorithm for the μ-calculus, and in fact (prior to [23]) even for its fragment CTL, that is simultaneously *optimal*, i.e. runs in ExpTime, and *single-pass*, i.e. avoids building an exponential-sized data structure in a first pass. Typical data structures used in worst-case-optimal algorithms are automata [10], games [13], and, for sublogics such as CTL, first-pass tableaux [9].

The term *global caching* describes a family of single-pass tableau algorithms [18, 21] that build graph-shaped tableaux bottom-up in so-called *expansion* steps, with no label ever generated twice, and attempt to terminate before the tableau is completely expanded by means of judicious intermediate *propagation* of satisfiability and/or unsatisfiability through partially expanded tableaux. Global caching offers wide room for heuristic optimization, regarding standard tableau optimizations as well as the order in which expansion and propagation steps are triggered, and has been shown to perform competitively in practice; see [21] for an evaluation of heuristics in global caching for the description logic $\mathcal{ALCI}$. One major challenge with global caching algorithms is typically to prove soundness and completeness, which becomes harder in the presence of fixpoint operators. A global caching algorithm for PDL has been described by Goré and Widmann [20]; finding an optimal global caching algorithm even for CTL has been named as an open problem as late as 2014 [15] (a non-optimal, doubly exponential algorithm is known [15]).

The contribution of the present work is an optimal global-caching algorithm for satisfiability in the alternation-free μ-calculus, extending our earlier work on the single-variable (*flat*) fragment of the μ-calculus [23]. The algorithm actually works at the level of generality of the

alternation-free fragment of the coalgebraic μ-calculus [6], and thus covers also logics beyond the realm of standard Kripke semantics such as alternating-time temporal logic ATL [1], neighbourhood-based logics such as the monotone μ-calculus that underlies Parikh's game logic [32], or probabilistic fixpoint logic. To aid readability, we phrase our results in terms of the relational μ-calculus, and discuss the coalgebraic generalization only at the end of Section 4. The model construction in the completeness proof yields models of size $2^{\mathcal{O}(n)}$.

We have implemented of our algorithm as an extension of the Coalgebraic Ontology Logic Reasoner COOL, a generic reasoner for coalgebraic modal logics [22]; given the current state of the implementation of instance logics in COOL, this means that we effectively support alternation-free fragments of relational, monotone, and alternating-time [1] μ-calculi, thus in particular covering CTL and ATL. We have evaluated the tool in comparison with existing reasoners on benchmark formulas for CTL [19] (which appears to be the only candidate logic for which well-developed benchmarks are currently available) and on random formulas for ATL and the alternation-free relational μ-calculus, with promising results; details are discussed in Section 5.

Related Work. The theoretical upper bound ExpTime has been established for the full coalgebraic μ-calculus [6] (and earlier for instances such as the alternating-time μ-calculus AMC [36]), using a multi-pass algorithm that combines games and automata in a similar way as for the standard relational case, in particular involving the Safra construction. *Global caching* has been employed successfully for a variety of description logics [18, 21], and lifted to the level of generality of coalgebraic logics with global assumptions [16] and nominals [17].

A tableaux-based non-optimal (NExpTime) decision procedure for the full μ-calculus has been proposed in [24]. Friedmann and Lange [13] describe an optimal tableau method for the full μ-calculus that, unlike most other methods including the one we present here, makes do without requiring guardedness. Like earlier algorithms for the full μ-calculus, the algorithm constructs and solves a parity game, and in principle allows for an on-the-fly implementation. The models constructed in the completeness proof are asymptotically larger than ours, but presumably the proof can be adapted for the alternation-free case by using determinization of co-Büchi automata [29] instead of Safra's determinization of Büchi automata [35] to yield models of size $2^{\mathcal{O}(n)}$, like ours. For non-relational instances of the coalgebraic μ-calculus, including the alternation-free fragment of the alternating-time μ-calculus AMC, the $2^{\mathcal{O}(n)}$ bound on model size appears to be new, with the best known bound for the alternation-free AMC being $2^{\mathcal{O}(n \log n)}$ [36].

In comparison to our own recent work [23], we move from the flat to the alternation-free fragment, which means essentially that fixpoints may now be defined by mutual recursion, and thus can express properties such as 'all paths reach states satisfying p and q, respectively, in strict alternation until they eventually reach a state satisfying r'. Technically, the main additional challenge is the more involved structure of eventualities and deferrals, which now need to be represented using cascaded sequences of unfoldings in the focusing approach; this affects mainly the soundness proof, which now needs to organize termination counters in a tree structure. While the alternation-free algorithm instantiates to the algorithm from [23] for flat input formulas, its completeness proof includes a new model construction which yields a bound of $3^n \in 2^{\mathcal{O}(n)}$ on model size, slightly improving upon the bound $n \cdot 4^n$ from [23]. We present the new algorithm in terms that are amenable to a game-theoretic perspective, emphasizing the correspondence between global caching and game-solving. In fact, it turns out that global caching algorithms effectively consist in an integration of the separate steps of typical game-based methods for the μ-calculus [13, 14, 31] into a single

on-the-fly procedure that talks only about partially expanded tableau graphs, implicitly combining on-the-fly determinization of co-Büchi automata with on-the-fly solving of the resulting Büchi games [11]. This motivates the mentioned slogan that

global caching is on-the-fly determinization and game solving.

In particular, the propagation steps in the global caching pattern can be seen as solving an incomplete Büchi game that is built directly by the expansion steps, avoiding explicit determinization of co-Büchi automata analogously to [29]. One benefit of an explicit global caching algorithm integrating the pipeline from tableaux to game solving is the implementation freedom afforded by the global caching pattern, in which suitable heuristics can be used to trigger expansion and propagation steps in any order that looks promising.

2 Preliminaries: The μ-Calculus

We briefly recall the definition of the (relational) μ-calculus. We fix a set P of *propositions*, a set A of *actions*, and a set $\mathfrak{V}$ of fixpoint variables. Formulas ϕ, ψ of the μ-calculus are then defined by the grammar

$$\psi, \phi ::= \bot \mid \top \mid p \mid \neg p \mid X \mid \psi \wedge \phi \mid \psi \vee \phi \mid \langle a \rangle \psi \mid [a]\psi \mid \mu X.\, \psi \mid \nu X.\, \psi$$

where $p \in P$, $a \in A$, and $X \in \mathfrak{V}$; we write $|\psi|$ for the size of a formula ψ. Throughout the paper, we use η to denote one of the fixpoint operators μ or ν. We refer to formulas of the form $\eta X.\, \psi$ as *fixpoint literals*, to formulas of the form $\langle a \rangle \psi$ or $[a]\psi$ as *modal literals*, and to p, $\neg p$ as *propositional literals*. The operators μ and ν *bind* their variables, inducing a standard notion of *free variables* in formulas. We denote the set of free variables of a formula ψ by $FV(\psi)$. A formula ψ is *closed* if $FV(\psi) = \emptyset$, and *open* otherwise. We write $\psi \leq \phi$ ($\psi < \phi$) to indicate that ψ is a (proper) subformula of ϕ. We say that ϕ *occurs free* in ψ if ϕ occurs as a subformula in ψ that is not in the scope of any fixpoint. Throughout, we *restrict to formulas that are guarded*, i.e. have at least one modal operator between any occurrence of a variable X and an enclosing binder ηX. (This is standard although possibly not without loss of generality [13].) Moreover we assume w.l.o.g. that input formulas are *clean*, i.e. all fixpoint variables are distinct, and *irredundant*, i.e. $X \in FV(\psi)$ for all subformulas $\eta X.\, \psi$.

Formulas are evaluated over *Kripke structures* $\mathcal{K} = (W, (R_a)_{a \in A}, \pi)$, consisting of a set W of *states*, a family $(R_a)_{a \in A}$ of relations $R_a \subseteq W \times W$, and a valuation $\pi : P \to \mathcal{P}(W)$ of the propositions. Given an *interpretation* $i : \mathfrak{V} \to \mathcal{P}(W)$ of the fixpoint variables, define $[\![\psi]\!]_i \subseteq W$ by the obvious clauses for Boolean operators and propositions, $[\![X]\!]_i = i(X)$, $[\![\langle a \rangle \psi]\!]_i = \{v \in W \mid \exists w \in R_a(v).w \in [\![\psi]\!]_i\}$, $[\![[a]\psi]\!]_i = \{v \in W \mid \forall w \in R_a(v).w \in [\![\psi]\!]_i\}$, $[\![\mu X.\, \psi]\!]_i = \mu [\![\psi]\!]_i^X$ and $[\![\nu X.\, \psi]\!]_i = \nu [\![\psi]\!]_i^X$, where $R_a(v) = \{w \in W \mid (v, w) \in R_a\}$, $[\![\psi]\!]_i^X(G) = [\![\psi]\!]_{i[X \mapsto G]}$, and μ, ν take least and greatest fixpoints of monotone functions, respectively. If ψ is closed, then $[\![\psi]\!]_i$ does not depend on i, so we just write $[\![\psi]\!]$. We write $x \models \psi$ for $x \in [\![\psi]\!]$. The *alternation-free fragment* of the μ-calculus is obtained by prohibiting formulas in which some subformula contains both a free ν-variable and a free μ-variable. E.g. $\mu X.\, \mu Y.\, (\Box X \wedge \Diamond Y \wedge \nu Z.\, \Diamond Z)$ is alternation-free but $\nu Z.\, \mu X.\, (\Box X \wedge \nu Y.\, (\Diamond Y \wedge \Diamond Z))$ is not. CTL is contained in the alternation-free fragment.

We have the standard *tableau rules* (each consisting of one *premise* and a possibly empty set of *conclusions*) which will be interpreted AND-OR style, i.e. to show satisfiability of a set of formulas Δ, it will be necessary to show that *every* rule application that matches Δ has

some conclusion that is satisfiable. Our algorithm will use these rules in the expansion step.

$$(\bot)\ \frac{\Gamma, \bot}{} \qquad (\lightning)\ \frac{\Gamma, p, \neg p}{}$$

$$(\wedge)\ \frac{\Gamma, \psi \wedge \phi}{\Gamma, \psi, \phi} \qquad (\vee)\ \frac{\Gamma, \psi \vee \phi}{\Gamma, \psi \quad \Gamma, \phi}$$

$$(\langle a \rangle)\ \frac{\Gamma, [a]\psi_1, \ldots, [a]\psi_n, \langle a \rangle \phi}{\psi_1, \ldots, \psi_n, \phi} \qquad (\eta)\ \frac{\Gamma, \eta X.\, \psi}{\Gamma, \psi[X \mapsto \eta X.\, \psi]}$$

(for $a \in A$, $n \in \mathbb{N}$, $p \in P$); we refer to the set of modal rules $(\langle a \rangle)$ by $\mathcal{R}_m$ and to the set of the remaining rules by $\mathcal{R}_p$ and usually write rules with premise Γ and conclusion $\Sigma = \Gamma_1, \ldots, \Gamma_n$ in sequential form, i.e. as (Γ / Σ).

▶ **Example 1.** As our running example, we pick a non-flat formula, i.e. one that requires two recursion variables. Consider the alternation-free formulas

$$\psi_1 = \mu X.\, ((p \wedge (r \vee \Box \psi_2)) \vee (\neg q \wedge \Box X)) \qquad \psi_2 = \mu Y.\, ((q \wedge (r \vee \Box X)) \vee (\neg p \wedge \Box Y))$$

(where $A = \{*\}$ and we write $\Box = [*]$, $\Diamond = \langle * \rangle$). The formulas ψ_1 and $\psi_2[X \mapsto \psi_1]$ state that all paths will visit p and q in strict alternation until r is eventually reached, starting with p and with q, respectively. Using the measure of *entanglement* [2], one can show that these properties cannot be expressed using only one variable.

3 The Global Caching Algorithm

We proceed to describe our global caching algorithm for the alternation-free μ-calculus. First off, we need some syntactic notions regarding decomposition of fixpoint literals.

▶ **Definition 2** (Deferrals)**.** Given fixpoint literals $\chi_i = \eta X_i.\, \psi_i$, $i = 1, \ldots, n$, we say that a substitution $\sigma = [X_1 \mapsto \chi_1]; \ldots; [X_n \mapsto \chi_n]$ *sequentially unfolds* χ_n if $\chi_i <_f \chi_{i+1}$ for all $1 \leq i < n$, where we write $\psi <_f \eta X.\, \phi$ if $\psi \leq \phi$ and ψ is open and occurs free in ϕ (i.e. σ unfolds a nested sequence of fixpoints in χ_n innermost-first). We say that a formula χ is *irreducible* if for every substitution $[X_1 \mapsto \chi_1]; \ldots; [X_n \mapsto \chi_n]$ that sequentially unfolds χ_n, we have that $\chi = \chi_1([X_2 \mapsto \chi_2]; \ldots; [X_n \mapsto \chi_n])$ implies $n = 1$ (i.e. $\chi = \chi_1$). An *eventuality* is an irreducible closed least fixpoint literal. A formula ψ *belongs* to an eventuality θ_n, or is a θ_n*-deferral*, if $\psi = \alpha\sigma$ for some substitution $\sigma = [X_1 \mapsto \theta_1]; \ldots; [X_n \mapsto \theta_n]$ that sequentially unfolds θ_n and some $\alpha <_f \theta_1$. We denote the set of θ_n-deferrals by $dfr(\theta_n)$.

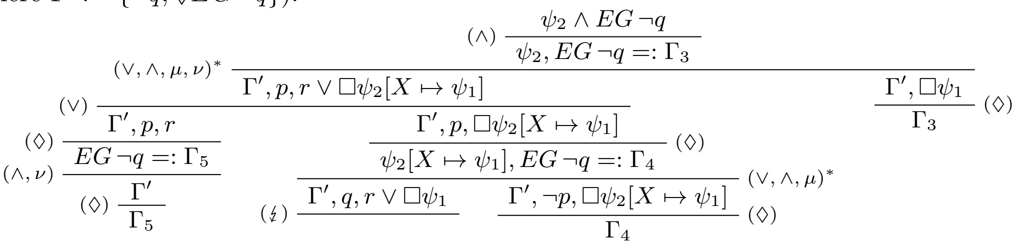
E.g. the substitution $\sigma = [Y \mapsto \mu Y.\, (\Box X \wedge \Diamond\Diamond Y)]; [X \mapsto \theta]$ sequentially unfolds the eventuality $\theta = \mu X.\, \mu Y.\, (\Box X \wedge \Diamond\Diamond Y)$, and $(\Diamond Y)\sigma = \Diamond \mu Y.\, (\Box\theta \wedge \Diamond\Diamond Y)$ is a θ-deferral. A fixpoint literal is irreducible if it is not an unfolding $\psi[X \mapsto \eta X.\, \psi]$ of a fixpoint literal $\eta X.\, \psi$; in particular, every clean irredundant fixpoint literal is irreducible.

▶ **Lemma 3.** *Each formula ψ belongs to at most one eventuality θ, and then $\theta \leq \psi$.*

▶ **Example 4.** Applying the tableau rules $\mathcal{R}_m$ and $\mathcal{R}_p$ to the formula $\psi_1 \wedge EG\, \neg r$, where ψ_1 is defined as in Example 1 and $EG\, \phi$ abbreviates $\nu X.\, (\phi \wedge \Diamond X)$, results in a cyclic graph, with relevant parts depicted as follows:

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\psi_1 \wedge EG\, \neg r}{\psi_1, EG\, \neg r =: \Gamma_1}(\wedge)}{\Gamma, p, \Box\psi_2[X \mapsto \psi_1]}(\vee,\wedge,\nu,\mu)^*}{\psi_2[X \mapsto \psi_1], EG\, \neg r =: \Gamma_2}(\Diamond) \qquad \cfrac{\Gamma, \neg q, \Box\psi_1}{\Gamma_1}(\Diamond)}{\Gamma, q, \Box\psi_1 \quad \Gamma, \neg p, \Box\psi_2[X \mapsto \psi_1]}(\vee,\wedge,\nu,\mu)^* \qquad \cfrac{\Gamma, \neg q, \Box\psi_1}{\Gamma_1}(\Diamond)}{\Gamma_1 \qquad \Gamma_2}(\Diamond)$$

where $\Gamma = \{\neg r, \Diamond EG\,\neg r\}$. The graph contains three cycles, all of which contain but never *finish* a formula that belongs to ψ_1 (where a formula belonging to an eventuality ψ_1 is said to be *finished* if it evolves to a formula that does not belong to ψ_1): In the rightmost cycle, the deferral $\delta_1 := \psi_1$ evolves to the deferral $\delta_2 := \Box\psi_1$ which then evolves back to δ_1. For the cycle in the middle, δ_1 evolves to $\delta_3 := \Box\psi_2[X \mapsto \psi_1]$ which in turn evolves to $\delta_4 := \psi_2[X \mapsto \psi_1]$ before looping back to δ_3. In the leftmost cycle, δ_1 evolves via δ_3 and δ_4 to δ_2 before cycling back to δ_1. The satisfaction of ψ_1 is thus being postponed indefinitely, since $EG\,\neg r$ enforces the existence of a path on which r never holds. As a successful example, consider the graph that is obtained when attempting to show the satisfiability of $\psi_1 \wedge EG\,\neg q$, (where $\Gamma' := \{\neg q, \Diamond EG\,\neg q\}$):



The two loops through Γ_3 and Γ_4 are unsuccessful as they indefinitely postpone the satisfaction of the deferrals δ_2 and δ_3, respectively; also there is the unsuccessful clashing node $\Gamma', q, r \vee \Box\psi_1$, containing both q and $\neg q$. However, the loop through Γ_5 is successful since it contains no deferral that is never finished; as all branching in this example is disjunctive, the single successful loop suffices to show that the initial node is successful. Our algorithm implements this check for 'good' and 'bad' loops by *simultaneously* tracking all deferrals that occur through the proof graph, checking whether each deferral is eventually finished.

We fix an input formula ψ_0 and denote the Fischer-Ladner closure [26] of ψ_0 by $\mathbf{F}$; notice that $|\mathbf{F}| \leq |\psi_0|$. Let $\mathbf{N} = \mathcal{P}(\mathbf{F})$ be the set of all *nodes* and $\mathbf{S} \subseteq \mathbf{N}$ the set of all *state nodes*, i.e. nodes that contain only $\top$, non-clashing propositional literals (where p *clashes* with $\neg p$) and modal literals; so $|\mathbf{S}| \leq |\mathbf{N}| \leq 2^{|\psi_0|}$. Put

$$\mathbf{C} = \{(\Gamma, d) \in \mathbf{N} \times \mathcal{P}(\mathbf{F}) \mid d \subseteq \Gamma\}, \quad \text{and} \quad \mathbf{C}_G = \{(\Gamma, d) \in \mathbf{C} \mid \Gamma \in G\} \text{ for } G \subseteq \mathbf{N},$$

recalling that nodes are just sets of formulas; note $|\mathbf{C}| \leq 3^{|\psi_0|}$. Elements $v = (\Gamma, d) \in \mathbf{C}$ are called *focused nodes*, with *label* $l(v) = \Gamma$ and *focus* d. The idea of focusing single eventualities comes from work on LTL and CTL [27, 4]. In the alternation-free μ-calculus, eventualities may give rise to multiple deferrals so that one needs to focus *sets of deferrals* instead of single eventualities. Our algorithm incrementally builds a set of nodes but performs fixpoint computations on $\mathcal{P}(\mathbf{C})$, essentially computing winning regions of the corresponding Büchi game (with the target set of player 0 being the nodes with empty focus) on-the-fly.

▶ **Definition 5** (Conclusions). For a node $\Gamma \in \mathbf{N}$ and a set $\mathcal{S}$ of tableau rules, the set of *conclusions* of Γ under $\mathcal{S}$ is

$$Cn(\mathcal{S}, \Gamma) = \{\{\Gamma_1, \dots, \Gamma_n\} \in \mathcal{P}(\mathbf{N}) \mid (\Gamma/\Gamma_1 \dots \Gamma_n) \in \mathcal{S}\}.$$

We define $Cn(\Gamma)$ as $Cn(\mathcal{R}_m, \Gamma)$ if Γ is a state node and as $Cn(\mathcal{R}_p, \Gamma)$ otherwise. A set $N \subseteq \mathbf{N}$ of nodes is *fully expanded* if for each $\Gamma \in N$, $\bigcup Cn(\Gamma) \subseteq N$.

▶ **Definition 6** (Deferral tracking). Given a node $\Gamma = \psi_1, \dots, \psi_n, \phi$ and a state node $\Delta \in \mathbf{S}$ that contains $[a]\psi_1, \dots, [a]\psi_n, \langle a\rangle\phi$ as a subset, we say that Γ *inherits* ϕ *from* $(\langle a\rangle\phi, \Delta)$ and ψ_i from $([a]\psi_i, \Delta)$. For a non-state node $\Delta \in \mathbf{N}$, a node $\Gamma \in \mathbf{N}$ with $\phi \in \Gamma$, and $\psi \in \Delta$, Γ *inherits* ϕ *from* (ψ, Δ) if $\Gamma = \Gamma_i$ is conclusion of a non-modal rule $(\Gamma_0/\Gamma_1 \dots \Gamma_n)$ with

$\Gamma_0 = \Delta$ and either ψ has one of the forms ϕ, $\phi \vee \chi$, $\chi \vee \phi$, $\phi \wedge \chi$, $\chi \wedge \phi$, or $\psi = \eta X.\, \chi$ and $\phi = \chi[X \mapsto \psi]$. We put

$$\begin{aligned} \mathit{Inh}_m(\phi, \langle a\rangle\phi, \Delta) &= \{\Gamma \in \mathbf{N} \mid \Gamma \text{ inherits } \phi \text{ from } (\langle a\rangle\phi, \Delta)\} \\ \mathit{Inh}_m(\phi, [a]\phi, \Delta) &= \{\Gamma \in \mathbf{N} \mid \Gamma \text{ inherits } \phi \text{ from } ([a]\phi, \Delta)\} \\ \mathit{Inh}_p(\phi, \psi, \Delta) &= \{\Gamma \in \mathbf{N} \mid \Gamma \text{ inherits } \phi \text{ from } (\psi, \Delta)\}, \end{aligned}$$

where Δ is a state node in the first two clauses and a non-state node in the third clause. We write *evs* for the set of eventualities in $\mathbf{F}$. For a node $\Gamma \in \mathbf{N}$, the set of deferrals of Γ is

$$d(\Gamma) = \{\delta \in \Gamma \mid \exists \theta \in \mathit{evs}.\, \delta \in \mathit{dfr}(\theta)\}.$$

For a set $d \neq \emptyset$ of deferrals and nodes $\Gamma, \Delta \in \mathbf{N}$, we put

$$\begin{aligned} d_{\Delta \rightsquigarrow \Gamma} = \{\delta \in d(\Gamma) \mid\ & \exists \theta \in \mathit{evs}.\, \exists \langle a\rangle\delta \in d.\, \Gamma \in \mathit{Inh}_m(\delta, \langle a\rangle\delta, \Delta) \text{ and } \delta, \langle a\rangle\delta \in \mathit{dfr}(\theta) \text{ or} \\ & \exists [a]\delta \in d.\, \Gamma \in \mathit{Inh}_m(\delta, [a]\delta, \Delta) \text{ and } \delta, \langle a\rangle\delta \in \mathit{dfr}(\theta)\} \end{aligned}$$

if Δ is a state node, and

$$d_{\Delta \rightsquigarrow \Gamma} = \{\delta_1 \in d(\Gamma) \mid \exists \theta \in \mathit{evs}. \exists \delta_2 \in d.\, \Gamma \in \mathit{Inh}_p(\delta_1, \delta_2, \Delta) \text{ and } \delta_1, \delta_2 \in \mathit{dfr}(\theta)\}$$

if Δ is a non-state node. I.e. $d_{\Delta \rightsquigarrow \Gamma}$ is the set of deferrals that is obtained by *tracking* d from Δ to Γ, where Γ is the conclusion of a rule application to Δ. We put $\emptyset_{\Delta \rightsquigarrow \Gamma} = d(\Gamma)$, with the intuition that if the focus d is empty at (Δ, d), then we *refocus*, i.e. choose as new focus for the conclusion Γ the set $d(\Gamma)$ of *all* deferrals in Γ.

▶ **Example 7.** Revisiting the proof graphs from Example 4, we fix additional abbreviations $\Gamma_6 := \Gamma, \neg p, \Box\psi_2[X \mapsto \psi_1]$, $\Gamma_7 := \Gamma', p, r \vee \Box\psi_2[X \mapsto \psi_1]$ and $\Gamma_8 := \Gamma', p, r$. In the first graph, e.g. $d(\Gamma_6) = \{\delta_3\}$ and $d(\Gamma_2) = \{\delta_4\}$; in the second graph, e.g. $d(\Gamma_7) = \{r \vee \Box\psi_2[X \mapsto \psi_1]\}$ and $d(\Gamma_8) = \emptyset$. In the first graph, the node Γ_6 inherits the deferral δ_3 from δ_4 at Γ_2, i.e. $d(\Gamma_2)_{\Gamma_2 \rightsquigarrow \Gamma_6} = \{\delta_4\}_{\Gamma_2 \rightsquigarrow \Gamma_6} = \{\delta_3\}$ since $\Gamma_6 \in \mathit{Inh}_m(\psi_2[X \mapsto \psi_1], \Box\psi_2[X \mapsto \psi_1], \Gamma_2)$. Regarding the second graph, Γ_8 does not inherit any deferral from Γ_7, i.e. $d(\Gamma_7)_{\Gamma_8 \rightsquigarrow \Gamma_7} = \{r \vee \Box\psi_2[X \mapsto \psi_1]\}_{\Gamma_8 \rightsquigarrow \Gamma_7} = \emptyset$ since $\Gamma_8 \in \mathit{Inh}_p(r, r \vee \Box\psi_2[X \mapsto \psi_1], \Gamma_7)$ but $r \vee \Box\psi_2[X \mapsto \psi_1] \in \mathit{dfr}(\psi_1)$ while $r \notin \mathit{dfr}(\psi_1)$, i.e. $r \vee \Box\psi_2[X \mapsto \psi_1]$ belongs to ψ_1 but r does not. This corresponds to the intuition that Γ_8 represents a branch originating from Γ_7 that actually finishes the deferral $r \vee \Box\psi_2[X \mapsto \psi_1]$.

We next introduce the functionals underlying the fixpoint computations for propagation of satisfiability and unsatisfiability.

▶ **Definition 8.** Let $C \subseteq \mathbf{C}$ be a set of focused nodes. We define the functions $f : \mathcal{P}(C) \to \mathcal{P}(C)$ and $g : \mathcal{P}(C) \to \mathcal{P}(C)$ by

$$\begin{aligned} f(Y) &= \{(\Delta, d) \in C \mid \forall \Sigma \in \mathit{Cn}(\Delta).\, \exists \Gamma \in \Sigma.\, (\Gamma, d_{\Delta \rightsquigarrow \Gamma}) \in Y\} \\ g(Y) &= \{(\Delta, d) \in C \mid \exists \Sigma \in \mathit{Cn}(\Delta).\, \forall \Gamma \in \Sigma.\, (\Gamma, d_{\Delta \rightsquigarrow \Gamma}) \in Y\} \end{aligned}$$

for $Y \subseteq C$. We refer to C as the *base set* of f and g.

That is, a focused node (Δ, d) is in $f(Y)$ if each rule matching Δ has a conclusion Γ such that $(\Gamma, d') \in Y$, where the focus d' is the set of deferrals obtained by tracking d from Δ to Γ.

▶ **Definition 9** (Proof transitionals). For $X \subseteq C \subseteq \mathbf{C}$, we define the *proof transitionals* $\hat{f}_X : \mathcal{P}(C) \to \mathcal{P}(C)$, $\hat{g}_X : \mathcal{P}(C) \to \mathcal{P}(C)$ by

$$\hat{f}_X(Y) := (f(Y) \cap \overline{F}) \cup (f(X) \cap F) = f(Y) \cup (f(X) \cap F)$$
$$\hat{g}_X(Y) := (g(Y) \cup F) \cap (g(X) \cup \overline{F}) = g(X) \cup (g(Y) \cap \overline{F}),$$

for $Y \subseteq C$, where $F = \{(\Gamma, d) \in C \mid d = \emptyset\}$ and $\overline{F} = \{(\Gamma, d) \in C \mid d \neq \emptyset\}$ are the sets of focused nodes with empty and non-empty focus, respectively, and where C is the base set of f and g.

That is, $\hat{f}_X(Y)$ contains nodes with non-empty focus that have for each matching rule a successor node in Y as well as nodes with empty focus that have for each matching rule a successor node in X. The least fixpoint of $\hat{f}_X$ thus consists of those nodes that finish their focus – by eventually reaching nodes from F with empty focus – and loop to X afterwards.

▶ **Lemma 10.** *The proof transitionals are monotone w.r.t. set inclusion, i.e. if $X' \subseteq X$, $Y' \subseteq Y$, then $\hat{f}_{X'}(Y') \subseteq \hat{f}_X(Y)$ and $\hat{g}_{X'}(Y') \subseteq \hat{g}_X(Y)$.*

▶ **Definition 11** (Propagation). For $G \subseteq \mathbf{N}$, we define $E_G, A_G \subseteq \mathbf{C}_G$ as

$$E_G = \nu X.\mu Y.\ \hat{f}_X(Y) \quad \text{and} \quad A_G = \mu X.\nu Y.\ \hat{g}_X(Y),$$

where $\mathbf{C}_G$ is the base set of f and g.

Notice that in terms of games, the computation of E_G and A_G corresponds to solving an incomplete Büchi game. The set E_G contains nodes (Γ, d) for which player 0 has a strategy to enforce – for each infinite play starting at (Γ, d) – the Büchi condition that nodes in F, i.e. with empty focus, are visited infinitely often; similarly A_G is the winning region of player 1 in the corresponding game, i.e. contains the nodes for which player 1 has a strategy to enforce an infinite play that passes F only finitely often or a finite play that gets stuck in a winning position for player 1.

▶ **Example 12.** Returning to Example 4, we have $(\Gamma_1, d(\Gamma_1)) = (\Gamma_1, \{\psi_1\}) \in A_{G_1}$ and $(\Gamma_3, d(\Gamma_3)) = (\Gamma_3, \{\psi_1\}) \in E_{G_2}$ where G_1 and G_2 denote the set of all nodes of the first and the second proof graph, respectively; the global caching algorithm described later will therefore answer 'unsatisfiable' to Γ_1, and 'satisfiable' to Γ_3. To see $(\Gamma_1, \{\psi_1\}) \in A_{G_1}$ note that $A_{G_1} = \nu Y.\, \hat{g}_{A_{G_1}}(Y)$ by definition, so $A_{G_1} = (\hat{g}_{A_{G_1}})^n(\mathbf{C}_{G_1})$ for some n. For each focused node $(\Delta, d) \in \mathbf{C}_{G_1}$ there is a rule matching Δ all whose conclusions Γ satisfy $(\Gamma, d_{\Delta \rightsquigarrow \Gamma}) \in \mathbf{C}_{G_1}$, i.e. $g(\mathbf{C}_{G_1}) = \mathbf{C}_{G_1}$. Moreover, since all loops in G_1 indefinitely postpone some eventuality, no node with non-empty focus ever reaches one with empty focus, so $\hat{g}_\emptyset(\mathbf{C}_{G_1}) = \overline{F}$. Since $\hat{g}$ is monotone and $(\Gamma_1, \{\psi_1\}) \in \overline{F}$, we obtain by induction over n that $(\Gamma_1, \{\psi_1\}) \in (\hat{g}_{A_{G_1}})^n(\mathbf{C}_{G_1})$. To see $(\Gamma_3, d(\Gamma_3)) = (\Gamma_3, \{\psi_1\}) \in E_{G_2}$, note that that starting from Γ_3, the single deferral ψ_1 can be finished in finite time while staying in E_{G_2}. This holds because we can reach $(\Gamma_8, \emptyset)$ by branching to the left twice and $(\Gamma_8, \emptyset) \in E_{G_2}$, since the loop through Γ_5 does not contain any deferrals whose satisfaction is postponed indefinitely and hence is contained in E_{G_2}.

▶ **Lemma 13.** *If $G' \subseteq G$, then $E_{G'} \subseteq E_G$ and $A_{G'} \subseteq A_G$.*

▶ **Lemma 14.** *Let $G \subseteq \mathbf{N}$ be fully expanded. Then $E_G = \overline{A_G}$.*

Our algorithm constructs a partial tableau, maintaining sets $G, U \subseteq \mathbf{N}$ of *expanded* and *unexpanded* nodes, respectively. It computes $E_G, A_G \subseteq \mathbf{C}_G$ in the propagation steps; as these sets grow monotonically, they can be computed incrementally.

Algorithm (Global caching). Decide satisfiability of a closed formula ϕ_0.

1. (Initialization) Let $G := \emptyset$, $\Gamma_0 := \{\phi_0\}$, $U := \{\Gamma_0\}$.
2. (Expansion) Pick $t \in U$ and let $G := G \cup \{t\}$, $U := (U - \{t\}) \cup (\bigcup Cn(t) - G)$.
3. (Intermediate propagation) Optional: Compute E_G and/or A_G. If $(\Gamma_0, d(\Gamma_0)) \in E_G$, return 'Yes'. If $(\Gamma_0, d(\Gamma_0)) \in A_G$, return 'No'.
4. If $U \neq \emptyset$, continue with Step 2.
5. (Final propagation) Compute E_G. If $(\Gamma_0, d(\Gamma_0)) \in E_G$, return 'Yes', else 'No'.

Note that in Step 5, G is fully expanded. For purposes of the soundness proof, we note an immediate consequence of Lemmas 13 and 14:

▶ **Lemma 15.** *If some run of the algorithm without intermediate propagation steps is successful on input ϕ_0, then all runs on input ϕ_0 are successful.*

▶ Remark. For alternation-free fixpoint logics, the game-based approach (e.g. [14]) is to (1.) define a nondeterministic co-Büchi automaton of size $\mathcal{O}(n)$ that recognizes unsuccessful branches of the tableau. This automaton is then (2.) determinized to a deterministic co-Büchi automaton of size $2^{\mathcal{O}(n)}$ (avoiding the Safra construction using instead the method of [29]; here, alternation-freeness is crucial) and (3.) complemented to a deterministic Büchi automaton of the same size that recognizes successful branches of the tableau. A Büchi game is (4.) constructed as the product game of the carrier of the tableau and the carrier of the Büchi automaton. This game is of size $2^{\mathcal{O}(n)}$ and can be (5.) solved in time $2^{\mathcal{O}(n)}$. Our global caching algorithm integrates analogues of items (1.) to (5.) in one go: We directly construct the Büchi game (thus replacing (1.) through (4.) by a single definition) step-by-step during the computation of the sets E and A of (un)successful nodes as nested fixpoints of the proof transitionals; the propagation step corresponds to (5.). Our algorithm allows for intermediate propagation, corresponding to solving the Büchi game on-the-fly, i.e. before it has been fully constructed.

4 Soundness, Completeness and Complexity

Soundness. Let ϕ_0 be a satisfiable formula. By Lemma 15, it suffices to show that a run without intermediate propagation is successful.

▶ **Definition 16.** For a formula ψ, we define $\psi_X(\phi) = \psi[X \mapsto \phi]$, $\psi_X^0 = \bot$ and $\psi_X^{n+1} = \psi_X(\psi_X^n)$. We say that a Kripke structure $\mathcal{K}$ is *stabilizing* if for each state x in $\mathcal{K}$, each $\mu X.\, \psi$, and each fixpoint-free context $c(-)$ such that $x \models c(\mu X.\, \psi)$, there is $n \geq 0$ such that $x \models c(\psi_X^n)$.

We note that finite Kripke structures are stabilizing and import the finite model property (without requiring a bound on model size) for the μ-calculus from [26]; for the rest of the section, we thus fix w.l.o.g. a stabilizing Kripke structure $\mathcal{K} = (W, (R_a)_{a \in A}, \pi)$ satisfying the target formula ϕ_0 in some state.

▶ **Definition 17** (Unfolding tree)**.** Given a formula ψ, an *unfolding tree* t for ψ consists of the syntax tree of ψ together with a natural number as additional label for each node that represents a least fixpoint operator. We denote this number by $t(\kappa, \mu X.\, \phi)$ for an occurrence of a fixpoint literal $\mu X.\, \phi$ at position $\kappa \in \{0,1\}^*$ in ψ. We define the *unfolding* $\psi(t)$ of ψ according to an unfolding tree t for ψ by

$$X(t) = X \qquad (\phi_1 \wedge \phi_2)(t) = \phi_1(t_1) \wedge \phi_2(t_2) \quad (\mu X.\, \phi_1)(t) = (\phi_1(t_1))_X^{t(\epsilon, \mu X.\, \phi_1)},$$

where t_i is the i-th child of the root of t, and similar clauses for $\langle a \rangle$, $[a]$, $\vee$, and ν as for $\wedge$.

Given a formula ψ, we define the order $<_\psi$ on unfolding trees for ψ by lexically ordering the lists of labels obtained by pre-order traversal of the syntax tree of ψ.

▶ **Definition 18** (Unfolding). The *unfolding* of a formula ψ at a state x with $x \models \psi$ is defined as $unf(\psi, x) = \psi(t)$, where t is the least unfolding tree for ψ (w.r.t. $<_\psi$) such that $x \models \psi(t)$ (such a t exists by stabilization).

Note that in unfoldings, all least fixpoint literals $\mu X.\, \phi$ are replaced with finite iterates of ϕ.

▶ **Theorem 19** (Soundness). *The algorithm returns 'Yes' on input ϕ_0 if ϕ_0 is satisfiable.*

Proof. (Sketch) We show that any node (Γ, d) that is constructed by the algorithm and whose label is satisfied at some state x in $\mathcal{K}$ is successful, i.e. $(\Gamma, d) \in E_G$; the proof is by induction over the maximal modal depth of $unf(\delta, x)$ for $\delta \in d$. ◀

Completeness. Assume that the algorithm answers 'Yes' on input ϕ_0, having constructed the set $E := E_G$ of successful nodes. Put $D = \{(\Gamma, d) \in E \mid \Gamma \in \mathbf{S}\}$; note $|D| \leq |E| \leq 3^{|\phi_0|}$.

▶ **Definition 20** (Propositional entailment). For a finite set Ψ of formulas, we write $\bigwedge \Psi$ for the conjunction of the elements of Ψ. We say that Ψ *propositionally entails* a formula ϕ (written $\Psi \vdash_{PL} \phi$) if $\bigwedge \Psi \to \phi$ is a propositional tautology, where modal literals are treated as propositional atoms and fixpoint literals $\eta X.\phi$ are unfolded to $\phi(\eta X.\phi)$ (recall that fixpoint operators are guarded).

▶ **Definition 21.** We denote the set of formulas in a node Γ that do *not* belong to an eventuality θ by

$$N(\Gamma, \theta) = \{\phi \in \Gamma \mid \phi \notin dfr(\theta)\}.$$

A set d of deferrals is *sufficient* for $\delta \in dfr(\theta)$ at a node Γ, in symbols $d \vdash_\Gamma \delta$, if $d \cup N(\Gamma, \theta) \vdash_{PL} \delta$. We write $\vdash_\Gamma \delta$ to abbreviate $\emptyset \vdash_\Gamma \delta$.

▶ **Definition 22** (Timed-out tableau). Let $U \subseteq \mathbf{S} \times \mathbf{S}$ and let $L \subseteq U \times U$. We denote the set of L-successors of $v \in U$ by $L(v) = \{w \mid (v, w) \in L\}$. Let d be a set of deferrals. We put $to(\emptyset, n) = U$ for all n (to for *timeout*). For $d \neq \emptyset$, we put $to(d, 0) = \emptyset$ and define $to(d, m+1)$ to be the set of of focused nodes (Δ, d') such that writing $Cn(\Delta) = \{\Sigma_1, \dots, \Sigma_n\}$, we have $L(\Delta, d') = \{(\Gamma_1, d_1), \dots, (\Gamma_n, d_n)\}$ where for each i there exists $\Gamma \in \Sigma_i$ such that

- $\Gamma_i \vdash_{PL} \bigwedge \Gamma$ and $d_i \vdash_{\Gamma_i} d'_{\Delta \rightsquigarrow \Gamma}$, and
- $(\Gamma_i, d_i) \in to(d'', m)$ for some $d'' \subseteq d(\Gamma_i)$ with $d'' \vdash_{\Gamma_i} d_{\Delta \rightsquigarrow \Gamma}$.

If for each focused node $(\Gamma, d) \in U$ there is a number m such that $(\Gamma, d) \in to(d(\Gamma), m)$, then L is a *timed-out tableau* over U.

Roughly, $to(d, m)$ can be understood as the set of all focused nodes in U that finish all deferrals in d within m modal steps, i.e. with *time-out* m; this is similar to Kozen's *μ-counters* [25].

▶ **Lemma 23** (Tableau existence). *There exists a timed-out tableau over D.*

Proof sketch. Since $D \subseteq E_G$, we can define $L \subseteq D \times D$ in such a way that all paths in L visit F (the set of nodes with empty focus) infinitely often, so every deferral contained in some node in D will be focused by the unavoidable eventual refocusing; this new focus will in turn eventually be finished so that L is a timed-out tableau. ◀

For the rest of the section, we fix a timed-out tableau L over D and define a Kripke structure $\mathcal{K} = (D, (R_a)_{a \in A}, \pi)$ by taking $R_a(v)$ to be the set of focused nodes in $L(v)$ whose label is the conclusion of an $(\langle a \rangle)$-rule that matches $l(v)$ and by putting $\pi(p) = \{v \in D \mid p \in l(v)\}$.

▶ **Definition 24** (Pseudo-extension). The *pseudo-extension* $\widehat{[\![\phi]\!]}$ of ϕ in D is

$$\widehat{[\![\phi]\!]} = \{v \in D \mid l(v) \vdash_{PL} \phi\}.$$

▶ **Lemma 25** (Truth). *In the Kripke structure $\mathcal{K}$, $\widehat{[\![\psi]\!]} \subseteq [\![\psi]\!]$ for all $\psi \in \mathbf{F}$.*

Proof sketch. Induction on ψ, with an additional induction on time-outs in the case for least fixpoint literals, exploiting alternation-freeness. ◀

▶ **Corollary 26** (Completeness). *If a run of the algorithm with input ϕ_0 returns 'Yes', then ϕ_0 is satisfiable.*

Proof sketch. Combine the existence lemma and the truth lemma to obtain a model over D. Since $(\{\phi_0\}, d(\{\phi_0\})) \in E$ and $\widehat{[\![\phi_0]\!]} \subseteq [\![\phi_0]\!]$, there is a focused node in D that satisfies ϕ_0. ◀

As a by-product, our model construction yields

▶ **Corollary 27.** *Every satisfiable alternation-free fixpoint formula ϕ_0 has a model of size at most $3^{|\phi_0|}$.*

Thus we recover the bound of $2^{\mathcal{O}(n)}$ for the alternation-free relational μ-calculus, which can be obtained, e.g., by carefully adapting results from [13] to the alternation-free case; for the alternation-free fragment of the alternating-time μ-calculus, covered by the coalgebraic generalization discussed next, the best previous bound appears to be $n^{\mathcal{O}(n)} = 2^{\mathcal{O}(n \log n)}$ [36].

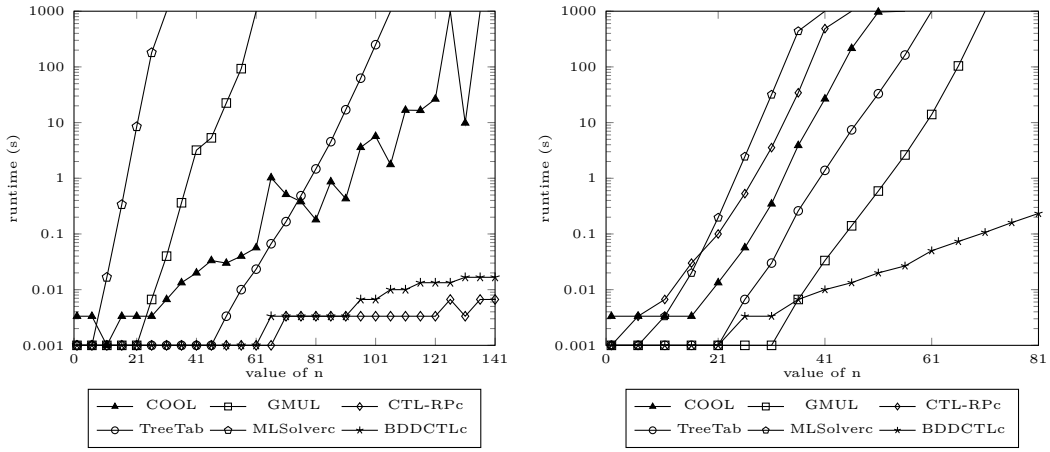
Complexity. Our algorithm has optimal complexity (given that the problem is known to be ExpTime-hard):

▶ **Theorem 28.** *The global caching algorithm decides the satisfiability problem of the alternation-free μ-calculus in* ExpTime*, more precisely in time $2^{\mathcal{O}(n)}$.*

The Alternation-Free Coalgebraic μ-Calculus. Coalgebraic logic [6] serves as a unifying framework for modal logics beyond standard relational semantics, subsuming systems with, e.g., probabilistic, weighted, game-oriented, or preference-based behaviour under the concept of coalgebras for a set functor F. All our results lift to the level of generality of the (alternation-free) coalgebraic μ-calculus [5]; details are in a technical report at `https://www8.cs.fau.de/hausmann/afgc.pdf`. In consequence, our results apply also to the alternation-free fragments of the alternating-time μ-calculus [1], probabilistic fixpoint logics, and the monotone μ-calculus (the ambient fixpoint logic of Parikh's game logic [32]), as all these can be cast as instances of the coalgebraic μ-calculus.

5 Implementation and Benchmarking

The global caching algorithm has been implemented as an extension of the *Coalgebraic Ontology Logic Reasoner* (COOL) [22], a generic reasoner for coalgebraic modal logics, available at `https://www8.cs.fau.de/research:software:cool`. COOL achieves its genericity by instantiating an abstract core reasoner that works for all coalgebraic logics to concrete instances of logics; our global caching algorithm extends this core. Instance logics implemented in COOL currently include relational, monotone, and alternating-time logics, as well as any logics that arise as fusions thereof. In particular, this makes COOL, to our knowledge, the only implemented reasoner for the alternation-free fragment of the alternating-time



(a) Montali, $n = 1$ (satisfiable). **(b)** Montali, $n = 1$ (unsatisfiable).

Figure 1 Runtimes for the Montali-formulas.

Table 1 Runtimes (in s) for the Alternating Bit Protocol formulas

Type of formula	COOL	TreeTab	GMUL	MLSolverc	BDDCTLc	CTL-RPc
(i)	<0.01	<0.01	<0.01	0.02	<0.01	0.02
(ii)	0.12	–	0.02	0.95	<0.01	0.15
(iii)	0.12	–	0.02	0.87	<0.01	0.16

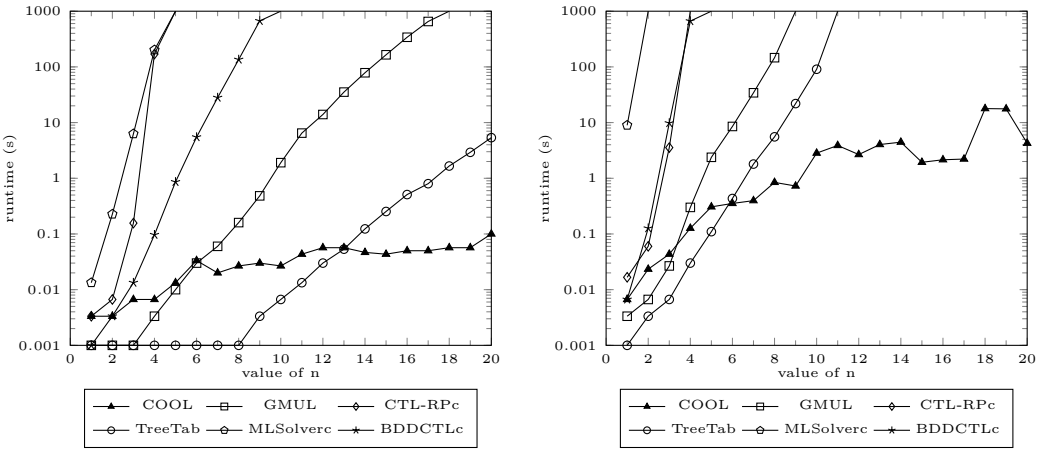
μ-calculus (a tableau calculus for the sublogic ATL is prototypically implemented in the TATL reasoner [8]) and the star-nesting free fragment of Parikh's game logic.

Although our tool supports the full alternation-free μ-calculus, we concentrate on CTL for experiments, as this appears to be the only candidate logic for which substantial sets of benchmark formulas are available [19]. CTL reasoners can be broadly classified as being either *top-down*, i.e. building graphs or tableaux by recursion over the formula, or *bottom-up*; the two groups perform very differently [19]. We compare our implementation with the top-down solvers TreeTab [15], GMUL [19], MLSolver [12] and the bottom-up solvers CTL-RP [37] and BDDCTL [19]. Out of the top-down solvers, only TreeTab is single-pass like COOL; however, TreeTab has suboptimal (doubly exponential) worst-case runtime. MLSolver supports the full μ-calculus. For MLSolver, CTL-RP and BDDCTL, formulas have first been *compacted* [19]. All tests have been executed on a system with Intel Core i7 3.60GHz CPU with 16GB RAM, and a stack limit of 512MB.

On the benchmark formulas of [19], COOL essentially performs similarly as the other top-down tools, and closer to the better tools when substantial differences show up. As an example, the runtimes of COOL, TreeTab, GMUL, MLSolver, CTL-RP, and BDDCTL on the Montali-formulas [30, 19] are shown in Figure 1. To single out one more example, Table 1 shows the runtimes for the alternating bit protocol benchmark from [19]; COOL performs closer to GMUL than to MLSolverc on these formulas.

This part of the evaluation may be summed up as saying that COOL performs well despite being, at the moment, essentially unoptimized: the only heuristics currently implemented is a simple-minded dependency of the frequency of intermediate propagation on the number of unexpanded nodes.

In addition, we design two series of unsatisfiable benchmark formulas that have an exponen-



(a) $early(n, 4, 2)$ (unsatisfiable). **(b)** $early_{gc}(n, 4, 2)$ (unsatisfiable).

Figure 2 Formulas with exponential search space and sub-exponential refutations.

tially large search space but allow for detection of unsatisfiability at an early stage. Recall that in CTL we can express the statement 'in the next step, the n-bit counter x represented by the variables $x_1, \ldots, x_n$ will be incremented' (with wraparound) as a formula $c(x, n)$ of polynomial size in n. We define unsatisfiable formulas $early(n, j, k)$ that specify an n-bit counter p with n bits and additionally branch after 2^j steps (i.e. when p_j holds) to start a counter r with k bits which in turn forever postpones the eventuality $EF\ p$:

$$\begin{aligned} early(n, j, k) = {} & start_p \wedge init(p, n) \wedge init(r, k) \wedge AG\ ((r \to c(r, k)) \wedge (p \to c(p, n))) \wedge \\ & AG\ ((\textstyle\bigwedge_{0 \le i \le j} p_i \to EX(start_r \wedge EF\ p)) \wedge \neg(p \wedge r) \wedge (r \to AX\ r)) \\ init(x, m) = {} & AG\ ((start_x \to (x \wedge \textstyle\bigwedge_{0 \le i < m} \neg x_i)) \wedge (x \to EX\ x)). \end{aligned}$$

Note here that *init* uses x as a string argument; $start_x$ is an atom indicating the start of counter x, and the atom x itself indicates that the counter x is running. The second series of unsatisfiable formulas $early_{gc}(n, j, k)$ is obtained by extending the formulas $early(n, j, k)$ with the additional requirement that a further counter q with n bits is started infinitely often, but at most at every second step:

$$\begin{aligned} early_{gc}(n, j, k) = {} & early(n, j, k) \wedge b \wedge init(q, n) \wedge AG\ (\neg(p \wedge q) \wedge \neg(q \wedge r) \wedge (q \to c(q, n))) \\ & \wedge\ AG\ (AF\ b \wedge (b \to (EX\ p \wedge EX\ start_q \wedge AX\ \neg b))) \end{aligned}$$

Figure 2 shows the respective runtimes for these formulas. In all cases, COOL finishes before the tableau is fully expanded, while GMUL and MLSolver will necessarily complete their first pass before being able to decide the formulas, and hence exhibit exponential behaviour; TreeTab seems not to benefit substantially from its capability to close tableaux early. For the $early_{gc}$ formulas, the ability to cache previously seen nodes appears to provide COOL with additional advantages. The $early_{gc}$ series can be converted into satisfiable formulas by replacing AX with EX, with similar results.

Due to the apparent lack of benchmarking formulas for the alternation-free μ-calculus and ATL, we compare runtimes on random formulas for these logics. For the alternation-free μ-calculus, formulas were built from 250 random operators (where disjunction and conjunction are twice as likely as the other operators). The experiment was conducted with formulas over three and over ten propositional atoms, respectively. MLSolver ran out of memory on

21% on the formulas over three atoms and on 16% of the formulas over ten atoms. COOL answered all queries without exceeding memory restrictions, and in under one second for all queries but one. Altogether, COOL was faster than MLSolver for more than 98% of the random alternation-free formulas, with the median of the ratios of the runtimes being 0.0431 in favour of COOL for formulas over three atoms and 0.0833 for formulas over ten atoms (recall however that MLSolver supports the full μ-calculus). For alternating-time temporal logic ATL, we compared the runtimes of TATL and COOL on random formulas consisting of 50 random operators; COOL answered faster than TATL on all of the formulas, with the median of the ratios of runtimes being 0.000668 in favour of COOL.

6 Conclusion

We have presented a tableau-based global caching algorithm of optimal (ExpTime) complexity for satisfiability in the alternation-free coalgebraic μ-calculus; the algorithm instantiates to the alternation-free fragments of e.g. the relational μ-calculus, the alternating-time μ-calculus (AMC) and the serial monotone μ-calculus. Essentially, it simultaneously generates and solves a deterministic Büchi game on-the-fly in a direct construction, in particular skipping the determinization of co-Büchi automata; the correctness proof, however, is stand-alone. We have generalized the $2^{\mathcal{O}(n)}$ bound on model size for alternation-free fixpoint formulas from the relational case to the coalgebraic level of generality, in particular to the AMC.

We have implemented the algorithm as part of the generic solver COOL; the implementation shows promising performance for CTL, ATL and the alternation-free relational μ-calculus. An extension of our global caching algorithm to the full μ-calculus would have to integrate Safra-style determinization of Büchi automata [35] and solving of the resulting parity game, both on-the-fly.

References

1. Rajeev Alur, Thomas Henzinger, and Orna Kupferman. Alternating-time temporal logic. *J. ACM*, 49:672–713, 2002.
2. Dietmar Berwanger, Erich Grädel, and Giacomo Lenzi. The variable hierarchy of the μ-calculus is strict. *Theory Comput. Sys.*, 40:437–466, 2007.
3. Julian Bradfield and Colin Stirling. Modal μ-calculi. In *Handbook of Modal Logic*, pages 721–756. Elsevier, 2006.
4. Kai Brünnler and Martin Lange. Cut-free sequent systems for temporal logic. *J. Log. Algebr. Prog.*, 76:216–225, 2008.
5. Corina Cîrstea, Clemens Kupke, and Dirk Pattinson. EXPTIME tableaux for the coalgebraic μ-calculus. *Log. Meth. Comput. Sci.*, 7, 2011.
6. Corina Cîrstea, Alexander Kurz, Dirk Pattinson, Lutz Schröder, and Yde Venema. Modal logics are coalgebraic. *Comput. J.*, 54:31–41, 2011.
7. Edmund Clarke and E. Allen Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Logics of Programs*, volume 131 of *LNCS*, pages 52–71. Springer, 1982.
8. Amélie David. TATL: Implementation of ATL tableau-based decision procedure. In *Automated Reasoning with Analytic Tableaux and Related Methods, TABLEAUX 2013*, volume 8123 of *LNCS*, pages 97–103. Springer, 2013.
9. E. Allen Emerson and Joseph Halpern. Decision procedures and expressiveness in the temporal logic of branching time. *J. Comput. Sys. Sci.*, 30:1–24, 1985.
10. E. Allen Emerson and Charanjit Jutla. The complexity of tree automata and logics of programs. *SIAM J. Comput.*, 29(1):132–158, September 1999.

11 Oliver Friedmann and Martin Lange. Local strategy improvement for parity game solving. In *Games, Automata, Logic, and Formal Verification, GANDALF 2010*, volume 25 of *EPTCS*, pages 118–131. Open Publishing Association, 2010.

12 Oliver Friedmann and Martin Lange. A solver for modal fixpoint logics. In *Methods for Modalities, M4M-6 2009*, volume 262 of *ENTCS*, pages 99–111, 2010.

13 Oliver Friedmann and Martin Lange. Deciding the unguarded modal μ-calculus. *J. Appl. Non-Classical Log.*, 23:353–371, 2013.

14 Oliver Friedmann, Markus Latte, and Martin Lange. Satisfiability games for branching-time logics. *Log. Methods Comput. Sci.*, 9, 2013.

15 Rajeev Goré. And-Or tableaux for fixpoint logics with converse: LTL, CTL, PDL and CPDL. In *Automated Reasoning, IJCAR 2014*, volume 8562 of *LNCS*, pages 26–45. Springer, 2014.

16 Rajeev Goré, Clemens Kupke, and Dirk Pattinson. Optimal tableau algorithms for coalgebraic logics. In *Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2010*, volume 6015 of *LNCS*, pages 114–128. Springer, 2010.

17 Rajeev Goré, Clemens Kupke, Dirk Pattinson, and Lutz Schröder. Global caching for coalgebraic description logics. In *Automated Reasoning, IJCAR 2010*, volume 6173 of *LNCS*, pages 46–60. Springer, 2010.

18 Rajeev Goré and Linh Anh Nguyen. Exptime tableaux for ALC using sound global caching. *J. Autom. Reasoning*, 50:355–381, 2013.

19 Rajeev Goré, Jimmy Thomson, and Florian Widmann. An experimental comparison of theorem provers for CTL. In *Temporal Representation and Reasoning, TIME 2011*, pages 49–56. IEEE, 2011.

20 Rajeev Goré and Florian Widmann. An optimal on-the-fly tableau-based decision procedure for PDL-satisfiability. In *Automated Deduction, CADE 2009*, volume 5663 of *LNCS*, pages 437–452. Springer, 2009.

21 Rajeev Goré and Florian Widmann. Sound global state caching for $\mathcal{ALC}$ with inverse roles. In *Automated Reasoning with Analytic Tableaux and Related Methods, TABLEAUX 2009*, volume 5607 of *LNCS*, pages 205–219. Springer, 2009.

22 Daniel Gorín, Dirk Pattinson, Lutz Schröder, Florian Widmann, and Thorsten Wißmann. COOL – a generic reasoner for coalgebraic hybrid logics (system description). In *Automated Reasoning, IJCAR 2014*, volume 8562 of *LNCS*, pages 396–402. Springer, 2014.

23 Daniel Hausmann and Lutz Schröder. Global caching for the flat coalgebraic μ-calculus. In *Temporal Representation and Reasoning, TIME 2015*, pages 121–143. IEEE, 2015.

24 Natthapong Jungteerapanich. A tableau system for the modal μ-calculus. In *Automated Reasoning with Analytic Tableaux and Related Methods, TABLEAUX 2009*, volume 5607 of *LNCS*, pages 220–234. Springer, 2009.

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

26 Dexter Kozen. A finite model theorem for the propositional μ-calculus. *Stud. Log.*, 47:233–241, 1988.

27 Martin Lange and Colin Stirling. Focus games for satisfiability and completeness of temporal logic. In *Logic in Computer Science, LICS 2001*, pages 357–365. IEEE Computer Society, 2001.

28 Zohar Manna and Amir Pnueli. The modal logic of programs. In *Automata, Languages and Programming, ICALP 1979*, volume 71 of *LNCS*, pages 385–409. Springer, 1979.

29 Satoru Miyano and Takeshi Hayashi. Alternating finite automata on ω-words. *Theoret. Comput. Sci.*, 32:321–330, 1984.

30 Marco Montali, Paolo Torroni, Marco Alberti, Federico Chesani, Marco Gavanelli, Evelina Lamma, and Paola Mello. Verification from declarative specifications using logic programming. In *Logic Programming, ICLP 2008*, volume 5366 of *LNCS*, pages 440–454. Springer, 2008.

31 Damian Niwinski and Igor Walukiewicz. Games for the μ-calculus. *Theor. Comput. Sci.*, 163:99–116, 1996.

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

33 Amir Pnueli. The temporal logic of programs. In *Foundations of Computer Science, FOCS 1977*, pages 46–57. IEEE Computer Society, 1977.

34 Vaughan Pratt. Semantical considerations on Floyd-Hoare logic. In *Foundations of Computer Science, FOCS 1976*, pages 109–121. IEEE Computer Society, 1976.

35 Shmuel Safra. On the complexity of omega-automata. In *Foundations of Computer Science, FOCS 1988*, pages 319–327. IEEE Computer Society, 1988.

36 Sven Schewe. *Synthesis of distributed systems*. PhD thesis, Universität des Saarlands, 2008.

37 Lan Zhang, Ullrich Hustadt, and Clare Dixon. A resolution calculus for the branching-time temporal logic CTL. *ACM Trans. Comput. Log.*, 15, 2014.