Global Caching for the Flat Coalgebraic μ-Calculus

Daniel Hausmann and Lutz Schröder
Friedrich-Alexander Universität Erlangen-Nürnberg

***Abstract*—Branching-time temporal logics generalizing relational temporal logics such as CTL have been proposed for various system types beyond the purely relational world. This includes, e.g., alternating-time logics, which talk about winning strategies over concurrent game structures, and Parikh's game logic, which is interpreted over monotone neighbourhood frames, as well as probabilistic fixpoint logics. Coalgebraic logic has emerged as a unifying semantic and algorithmic framework for logics featuring generalized modalities of this type. Here, we present a generic global caching algorithm for satisfiability checking in the flat coalgebraic mu-calculus, which realizes known tight exponential-time upper complexity bounds but offers potential for heuristic optimization. It is based on a tableau system that makes do without additional labelling of nodes beyond formulas from the standard Fischer-Ladner closure, such as foci or termination counters for eventualities. Moreover, the tableau system is single-pass, i.e. avoids building an exponential-sized structure in a first pass; to our best knowledge, optimal single-pass systems without numeric time-outs were not previously available even for CTL.**

***Keywords*-fixpoint logics; satisfiability; coalgebraic logic; global caching;**

Temporal logics of various descriptions traditionally play a central role in the verification of concurrent systems; typical reasoning tasks are model checking and satisfiability checking. Many of these logics, in particular Computation Tree Logic (CTL), can be seen as fixpoint logics; fixpoints appear also, e.g., in epistemic logics of common knowledge and in logics for knowledge representation [1]. Traditionally, logics of this type are equipped with a relational semantics, i.e. are interpreted over Kripke models; they can thus be seen as sublogics of the relational μ-calculus [2]. However, there is also increasing interest in non-relational fixpoint logics such as Alternating-Time Temporal Logic (ATL) [3], which is interpreted over so-called concurrent game structures, and Parikh's game logic, interpreted over monotone neighbourhood frames [4]. Another, not quite as standard but sensible, example is the extension of probabilistic modal logic [5], [6], [7] with fixpoints, interpreted over Markov chains.

For all of the mentioned logics, satisfiability is known to be EXPTIME-complete. However, all known tableau-based reasoning algorithms, even for the basic case of CTL, share the problem that they either run in suboptimal time or build an exponential-sized data structure in a first pass [8] (or use exponential time-outs, see the discussion of [9] below).

Coalgebraic logic serves as a unifying framework for logics such as these, i.e. for modalities beyond the relational world: by parametrizing the system type as a set functor, one can give a uniform treatment of all logics mentioned above. In the current work, we introduce a generic tableau system for satisfiability checking in the flat (i.e. single variable) fragment of the coalgebraic μ-calculus [10], [9]. It is well-known that the flat relational μ-calculus [11] contains CTL as a sublogic, e.g. $AF\phi = \mu X.(\phi \vee AX\, X)$. Similarly, ATL is a sublogic of the flat alternating-time μ-calculus [3]. Like earlier generic algorithms [10], [9], our algorithm has – typically optimal – complexity EXPTIME under mild assumptions on the parameters of the logic. While the theoretical complexity bound is thus not new, our algorithm has some distinguishing features in practical terms:

- It is single-pass, i.e. does not rely on building exponential-sized intermediate data structures such as automata, games, or first-pass tableaux; in other words, there is hope that good heuristics will often allow closing the tableau before it reaches exponential size.
- It supports sound *global caching* [12], [13] in the sense that one never needs to generate different tableau nodes with identical labels, even when these come from morally different branches of the tableau; it thus potentially offers comparative practical efficiency and opportunities for heuristic optimization [14].

Moreover, our tableau system avoids annotating formulas or tableau labels with additional data such as foci [15], [16] or time-outs [9], and hence has shorter branches. As a side benefit, our tableau method implies, via its completeness proof, a bound $2^{O(n)}$ on the size of models, the same as for CTL [17]; for ATL , the best previous bound appears to be $n^{O(n)} = 2^{O(n \log n)}$ [18].

Related Work: The EXPTIME upper bound for CTL was first established by Emerson and Halpern [19], using a two-pass method where an exponential-size tableau is built in the first pass and then pruned in the second pass. Later work includes the focussing method introduced in game-theoretic terms by Lange and Stirling [15] and translated into a complete and cutfree sequent system by Brünnler and Lange [16]. The focus game algorithm realizes the EXPTIME upper bound via alternating depth-first search in an exponentially large (but polynomially deep) and-or tree of plays. Tableau labels are annotated with a *focus* singling out one of the eventualities. The focussing sequent system for CTL [16] is single-pass but no claims are made about its computational complexity. Our method owes much to focussing in that we track how eventualities propagate through the tableau, and use methods similar to focussing in the model construction; however, we avoid any additional annotations in the tableau labels themselves. An alternative to tableau methods for CTL are resolution methods, e.g. [20].

Our algorithm works for the full flat μ-calculus, which is strictly more expressive than CTL. Friedmann and Lange [21] have recently described a tableau method for the full μ-calculus, whose decision procedure constructs an exponential-sized parity

game. Dedicated tableau algorithms exploiting flatness in the relational μ-calculus do not currently seem to exist.

For ATL, the upper bound EXPTIME has been established for a fixed number of agents by Goranko and van Drimmelen [22], and for a variable number of agents by Walther et al. [23]. It has been extended to the full alternating-time μ-calculus AMC by Schewe [18]. The methods used in [22], [18] are automata-theoretic, and in particular will always construct an exponential-sized automaton from the target formula. The algorithm of [23] is by type elimination, i.e. starts from an exponential-sized set of types. Goranko and Shkatov [24] present an improved 'type-free' tableau for ATL in the spirit of [19], implemented in the TATL prover [25], which however still begins by constructing an exponential-sized pretableau.

Satisfiability checking in the full coalgebraic μ-calculus has been shown to be, under mild conditions, in EXPTIME by Cirstea et al. [26]. The algorithm combines game-theoretic and automata-theoretic methods, and in particular will always construct an exponential-sized game from the target formula. A single-pass tableau-oriented algorithm that does support global caching has been described for the flat coalgebraic μ-calculus [9]. However, this algorithm annotates eventualities with time-outs that are initialized to exponentially large values; this is clearly detrimental to the practical efficiency of the algorithm.

Global caching has been employed with considerable success for a variety of description logics [12], [13], [14], and lifted to the level of generality of coalgebraic logics with global assumptions [27] and nominals [28]. A doubly exponential global caching method for CTL has recently been described by Goré, with the problem of finding a single-pass tableau method for CTL that supports global caching *and* has optimal, i.e. singly exponential, complexity explicitly left open [8].

I. THE FLAT COALGEBRAIC μ-CALCULUS

We recall the generic framework of coalgebraic logic [29], [30] and its extension with flat fixpoints, i.e. the single-variable fragment of the coalgebraic μ-calculus [10], [9]; we briefly call single-variable μ-calculi *flat*. We then discuss examples, including CTL, ATL, and probabilistic fixpoint logics.

A coalgebraic logic is given in terms of syntactic and semantic parameters. The *syntax* of a *flat coalgebraic μ-calculus* is determined by a *(modal) similarity type* Λ, i.e. a set of modal operators with associated finite arities. For the sake of readability, we will pretend that all operators are unary, although nullary operators, i.e. propositional atoms, will appear in the examples. We work with formulas in negation normal form throughout, and therefore assume that every modal operator $\heartsuit \in \Lambda$ comes with a *dual* operator $\overline{\heartsuit} \in \Lambda$ of the same arity, where $\overline{\overline{\heartsuit}} = \heartsuit$. (For propositional atoms, dual is negation). This determines the set $\mathcal{F}(\Lambda)$ of *flat modal fixpoint formulas* ϕ, ψ by the grammar

$$\phi, \psi := \top \mid \bot \mid X \mid \psi \wedge \phi \mid \psi \vee \phi \mid \heartsuit\psi \mid \nu X.\psi \mid \mu X.\psi$$

where $\heartsuit \in \Lambda$ and X is a fixed single recursion variable. Negation is derived in the standard way, e.g. $\neg\heartsuit\phi = \overline{\heartsuit}\neg\phi$. The operations $\rightarrow, \leftrightarrow$ are defined as usual. We use the standard notion of free variables; a formula without free variables is *closed*. Throughout, we *restrict to formulas that are guarded*, i.e. have at least one modal operator between any occurrence of the recursion variable X and its enclosing binder μ or ν. (This is standard although not without loss of generality [21].)

We parametrize the *semantics* over the underlying class of systems and the interpretation of the modal operators. The former is determined by the choice of a *type functor* $T : \mathsf{Set} \to \mathsf{Set}$, i.e. an operation T that maps sets X to sets TX and functions $f : X \to Y$ to functions $Tf : TX \to TY$, preserving identities and composition, and the latter by the choice of a predicate lifting $[\![\heartsuit]\!]$ for each $\heartsuit \in \Lambda$. Here, a *predicate lifting* (for T) is a family of maps

$$\lambda_X : \mathcal{P}X \to \mathcal{P}TX,$$

for each set X (satisfying *naturality*, i.e. commutation with preimage). To enable fixpoints, we require that *modal operators are monotone*, i.e. $[\![\heartsuit]\!] : \mathcal{P}(X) \to \mathcal{P}(TX)$ is monotone w.r.t. set inclusion. Predicate liftings must respect duality of operators, i.e. $[\![\overline{\heartsuit}]\!]_X(A) = TX - [\![\heartsuit]\!]_X(X - A)$. Models of the logic are *T-coalgebras*, i.e. pairs (C, ξ) where C is a set of *states* and

$$\xi : C \to TC$$

is the transition function; thinking of TC informally as a parametrized datatype over C, we regard ξ as associating with each state x a structured collection $\xi(x)$ of successor states and observations. E.g. for $TX = \mathcal{P}(X) \times \mathcal{P}(U)$, with U a set of *propositional atoms*, a T-coalgebra $\xi : C \to \mathcal{P}(C) \times \mathcal{P}(U)$ is a Kripke model, as ξ maps a state to a set of successor states and a set of valid propositional atoms. Our main interest here is in examples beyond Kripke semantics, see Example 1. *We fix the data T, Λ, etc. throughout.*

Given a T-coalgebra (C, ξ) and a set $A \subseteq C$ interpreting the variable X, the semantics of a formula ϕ is a subset $[\![\phi]\!](A) \subseteq C$, or just $[\![\phi]\!]$ for closed ϕ. We write $x \models \phi$ for $x \in [\![\phi]\!]$. One has obvious clauses for Boolean operators, $[\![X]\!](A) = A$, and

$$[\![\heartsuit\phi]\!](A) = \xi^{-1}[[\![\heartsuit]\!]_C([\![\phi]\!](A))].$$

As expected, the semantics of $\mu X.\,\phi$ (resp. $\nu X.\,\phi$) is the least (resp. greatest) fixpoint of the monotone function $[\![\phi]\!](\_)$.

Example 1. We discuss selected examples covered by the coalgebraic approach, starting with a more detailed exposition of the basic example of Kripke semantics and then moving on to non-relational examples. The range of examples is unlimited in principle [31]; we concentrate on the main examples found in the literature. See [32] for additional examples.

1. *The relational μ-calculus:* As seen above, a Kripke frame can be viewed as a coalgebra for the powerset functor $\mathcal{P}$, and a Kripke model for a set U of propositional atoms is a coalgebra for $\mathcal{P} \times \mathcal{P}(U)$. The usual semantics of the modality $\Box$ (with dual $\bar{\Box} = \Diamond$) is induced coalgebraically by the predicate lifting

$$[\![\Box]\!]_X(A) = \{(B, P) \in \mathcal{P}(X) \times \mathcal{P}(U) \mid B \subseteq A\},$$

(and that of a propositional atom p by a *nullary* predicate lifting $[\![p]\!]_X = \{(B,P) \in \mathcal{P}(X) \times \mathcal{P}(U) \mid p \in P\}$; we elide propositional atoms from now on). Multi-relational versions, with modalities $[a]$ indexed over atomic programs or agents $a \in \Sigma$, are interpreted over coalgebras for $TX = \mathcal{P}(\Sigma \times X) \times \mathcal{P}(U)$. CTL, the $*$-nesting-free fragment of PDL, and the logic of common knowledge embed into the flat relational μ-calculus; e.g., $E[\phi U \psi] = \mu X.(\psi \vee (\phi \wedge \Diamond X))$. The flat μ-calculus is strictly more expressive than CTL; e.g. it expresses 'p holds in all even states on any path' as $\nu X.\, (p \wedge \Box\Box X)$ [17], [33].

2. *Probabilistic fixpoint logic* extends probabilistic modal logic [5], [6], [7] with fixpoints. It has modal operators L_p 'with probability at least p, the next state satisfies' with duals $\bar{L}_p = \neg L_p \neg$ 'with probability more than $1-p$, the next state satisfies', for $p \in [0,1] \cap \mathbb{Q}$. These are interpreted over coalgebras for the functor $\mathcal{D}$ that takes a set X to the set $\mathcal{D}(X)$ of discrete probability distributions on X by predicate liftings

$$[\![L_p]\!]_X(A) = \{\mu \in \mathcal{D}(X) \mid \mu(A) \geq p\}.$$

Coalgebras for $\mathcal{D}$ are Markov chains. We intentionally refrain from referring to this logic as the probabilistic μ-calculus, which is standardly understood as taking truth values in $[0,1]$. As a simple example, the AG-like formula

$$\nu X.\, (\neg \mathsf{fail} \wedge L_p X)$$

expresses that the system will, at any point during its run time, fail with probability less than $1-p$.

3. *The alternating-time μ-calculus (AMC)* [3] has modal operators $\langle\!\langle A \rangle\!\rangle \bigcirc$ read 'coalition A has a joint strategy to enforce ... in one step', where a *coalition* is a subset of a fixed set N of agents; in coalition logic [34], these operators are denoted $[A]$, and we will use the latter notation in the AMC (but not in ATL) for brevity, writing dual operators as $\langle A \rangle$. The semantics is defined over *concurrent game structures* (or the very similar *game frames* [34]), which are coalgebras $\xi : C \to \mathcal{G}(C)$ for the functor

$$\mathcal{G}(X) = \{(f,(k_i)) \mid (k_i) \in \mathbb{N}^N; f : \textstyle\prod_{i \in N}[k_i] \to X\}$$

where $[k] = \{0,\dots,k\}$ for $k \in \mathbb{N}$; i.e. each state $x \in C$ specifies a game $(f,(k_i))$ in which each agent $i \in N$ has $k_i + 1$ moves, with the next state determined by applying the outcome function f to the moves chosen by the agents. The operators $[A]$ are interpreted via the liftings

$$[\![[A]]\!]_X(B) = \{(f,(k_i)) \in \mathcal{G}(X) \mid \exists (s_i) \in \textstyle\prod_{i \in A}[k_i].\ \forall (s_i) \in \prod_{i \in N-A}[k_i].\, f((s_i)_{i \in N}) \in B\},$$

i.e. $[A]\phi$ states that the agents in A have moves $(s_i)_{i \in A}$ such that regardless of the moves $(s_i)_{i \in N-A}$ by the other agents, the next state $f((s_i)_{i \in N})$ satisfies ϕ. The flat AMC contains alternating-time temporal logic (ATL); e.g. the ATL formula $\langle\!\langle A \rangle\!\rangle \phi U \psi$ ('coalition A can eventually force ψ and meanwhile maintain ϕ') is the flat fixpoint formula $\mu X.\, (\psi \vee (\phi \wedge [A]X))$.

4. *The serial monotone μ-calculus* is the ambient fixpoint calculus of Parikh's game logic [35] (and its sublogic, concurrent PDL [36]). It has modal operators $\langle a \rangle$, with duals $[a]$, indexed over *atomic games* $a \in \Sigma$. Its semantics is defined in terms of *serial monotone neighbourhood frames*, which are coalgebras for the functor $\mathcal{M}$ defined by

$$\mathcal{M}(X) = \{\mathfrak{A} : \Sigma \to \breve{\mathcal{P}}(\breve{\mathcal{P}}(X)) \mid \mathfrak{A}(a) \text{ upwards closed}, \emptyset \notin \mathfrak{A}(a) \ni X \text{ for all } a\}$$

with $\breve{\mathcal{P}}$ denoting contravariant powerset (i.e. $\breve{\mathcal{P}}(X)$ is powerset, and $\breve{\mathcal{P}}(f) : \breve{\mathcal{P}}(Y) \to \breve{\mathcal{P}}(X)$, $A \mapsto f^{-1}[A]$ for $f : X \to Y$). That is, an $\mathcal{M}$-coalgebra on X assigns to each state $x \in X$ and each atomic game a a set of subsets of X, the *a-neighbourhoods* of x, which designate properties of states that angel can enforce in the atomic game a. The semantics of the modal operators is given by the predicate liftings

$$[\![\langle a \rangle]\!]_X(A) = \{\mathfrak{A} \in \mathcal{M}(X) \mid A \in \mathfrak{A}(a)\}.$$

The $*$-nesting-free fragment of game logic embeds into the flat serial monotone μ-calculus. Game logic has operators $\langle \gamma \rangle$ 'Angel has a strategy to enforce ... in game γ', where γ can be composite; e.g., $\gamma^\times$ is (in finite frames) the game in which the game γ is played a finite number of times, with Demon deciding when to stop. The formula $\langle \gamma^\times \rangle \phi$ is equivalent to the flat fixpoint formula $\nu X.\, (\phi \wedge \langle \gamma \rangle X)$.

Our tableau algorithm will be parametrized by an axiomatization of the modalities by *one-step* (tableau) *rules* [32]:

Definition 2 (One-step tableau rules)**.** We fix a set P of *(propositional) variables*. We denote by $\Lambda(P)$ the set $\{\heartsuit b \mid \heartsuit \in \Lambda, b \in P\}$ of formulas consisting of an application of a modal operator $\heartsuit$ to an element of P. Given a set U, a *U-(tableau-)sequent* is a subset of U, written $u_1,\dots,u_n$ for $u_i \in U$ and read *conjunctively*. We also use the comma to denote the union of sequents. A *one-step tableau rule* $R = (\Gamma_0/\Gamma_1 \dots \Gamma_n)$ consists of a $\Lambda(P)$-sequent Γ_0, the *premise*, and P-sequents $\Gamma_1,\dots,\Gamma_n$ ($n \geq 0$), the *conclusions*, with the additional provisos that Γ_0 mentions every variable at most once, and $\Gamma_1,\dots,\Gamma_n$ mention only variables occurring in Γ_0. Given a $\mathcal{P}(C)$-valuation $\tau : P \to \mathcal{P}(C)$, we define the extension $[\![\Gamma]\!]_C\tau \subseteq C$ of a P-sequent Γ by $[\![\Gamma]\!]_C\tau = \bigcap_{b \in \Gamma} \tau(b)$, and the extension $[\![\Delta]\!]_{TC} \subseteq TC$ of a $\Lambda(P)$-sequent Δ by $[\![\Delta]\!]_{TC}\tau = \bigcap_{\heartsuit b \in \Delta} [\![\heartsuit]\!]_C(\tau(b))$.

A rule $(\Gamma_0/\Gamma_1 \dots \Gamma_n)$ is *one-step tableau sound* if for all $\mathcal{P}(C)$-valuations $\tau : P \to \mathcal{P}(C)$, $[\![\Gamma_0]\!]_{TC}\tau \neq \emptyset$ implies that $[\![\Gamma_i]\!]_C\tau \neq \emptyset$ for some $i \in \{1,\dots,n\}$.

A set of one-step tableau rules $\mathcal{R}$ is *one-step tableau complete* if whenever Γ is a $\Lambda(P)$-sequent and $\tau : P \to \mathcal{P}(C)$ is a $\mathcal{P}(C)$-valuation such that for each rule $(\Gamma_0/\Gamma_1 \dots \Gamma_n) \in \mathcal{R}$ and for each renaming $\sigma : P \to P$ such that $\Gamma_0\sigma \subseteq \Gamma$ and σ acts injectively on Γ_0, $[\![\Gamma_i\sigma]\!]_C\tau \neq \emptyset$ for some $1 < i \leq n$, then $[\![\Gamma]\!]_{TC}\tau \neq \emptyset$.

One-step tableau soundness captures, at the very simple level of a semantics involving only single elements of the type functor T rather than full-blown T-coalgebras, the intuition that, dually to proof rules, a tableau rule is sound if satisfiability of the premise implies that one of the conclusions is satisfiable. Similarly, one-step tableau completeness embodies the intuition that a tableau

system is complete if a tableau sequent is satisfiable whenever all rule instances matching it have a satisfiable conclusion (for fixpoint logics, the actual satisfiability algorithm is necessarily more complicated than that).

Remark 3. In the terminology of [26], our notion of one-step tableau completeness comprises one-step tableau completeness *and* closure under contraction.

We *fix a one-step tableau sound and complete set $\mathcal{R}$ of one-step tableau rules throughout*. Such rule sets exist for all coalgebraic logics, and concrete sets have been identified for all our example logics [32]; they are determined by the functor and predicate liftings alone and have been reused for coalgebraic logics of varying expressivity, including coalgebraic hybrid logics [37] and coalgebraic description logics [38].

Example 4. We briefly recall one-step tableau sound and complete rule sets for our main examples [32], [26].

1. The *relational μ-calculus* has rules

$$\frac{\langle a\rangle b_0, [a]b_1, \ldots, [a]b_n,}{b_0, b_1, \ldots, b_n} \ (a \in \Sigma, n \geq 0)$$

– i.e. the usual tableau rules for relational modalities. For atomic propositions p with duals (i.e. negations) $\bar{p}$, we have additional rules $p, \bar{p}/\bot$ with $\bot$ denoting the absence of a conclusion; that is, the usual *axiom rule* becomes a one-step tableau rule.

2. *Probabilistic fixpoint logic* has rules with multiple conclusions (slightly rearranged in comparison to [32]): for each choice of positive integers r_i and s_j, we have a rule

$$\frac{L_{p_1}a_1, \ldots, L_{p_n}a_n, \bar{L}_{q_1}b_1, \ldots, \bar{L}_{q_m}b_m}{\sum_{i=1}^{n} r_i(a_i - p_i) \sqsupset \sum_{j=1}^{m} s_j(\neg b_j - q_j)}$$

where $\sqsupset$ is $\geq$ in case $m = 0$, and $>$ otherwise. Here, we read the linear inequality as Boolean function f in the variables $a_1, \ldots, a_n, b_1 \ldots, b_m$ by interpreting the truth value $\top$ as 1 and $\bot$ as 0. The conclusions of the rule are then the prime implicants of f, that is, the minimal conjunctive clauses (i.e. conjunctions of literals) χ over P such that χ propositionally implies $f(a_1, \ldots, a_n, b_1, \ldots, b_m) = 1$. As f is clearly monotone, its prime implicants are sequents in the sense of Definition 2, i.e. do not contain negative literals. E.g. for the sequent $L_{1/2}a, \bar{L}_{1/2}b$, we have a rule match (for $r_1 = 1 = s_1$) with conclusion $a - 1/2 > \neg b - 1/2$, a Boolean function that has $a \wedge b$ as its only prime implicant.

3. The *alternating-time μ-calculus* has rules

$$\frac{[A_1]a_1, \ldots, [A_n]a_n, \langle D\rangle b, \langle N\rangle c_1, \ldots, \langle N\rangle c_m}{a_1, \ldots, a_n, b, c_1, \ldots, c_m}$$

$$\frac{[A_1]a_1, \ldots, [A_n]a_n}{a_1, \ldots, a_n}$$

for coalitions A_i that are pairwise disjoint and, in the first rule, contained in D. The second rule says that disjoint coalitions can combine their abilities; to understand the first rule note additionally that $\langle D\rangle b$ implies that the A_i together cannot prevent b, and $\langle N\rangle c_i$ says that c_i is unavoidable altogether.

4. The *serial monotone μ-calculus* has rules

$$\frac{[a]b, \langle a\rangle c}{b, c} \quad \frac{[a]b}{b} \quad \frac{\langle a\rangle b}{b}$$

capturing monotonicity and seriality.

II. Technical Preliminaries

The *size* $|\phi|$ of a formula ϕ is its length over the alphabet $\{\top, \bot, \wedge, \vee, X, \mu X, \nu X\} \cup \Lambda$. We write $\psi \leq \phi$ ($\psi < \phi$) if ψ is a (proper) subformula of ϕ. We use the metavariable η to denote either μ or ν. A formula of the shape $\heartsuit\phi$ is a *modal literal* while a formula of the shape $\eta X.\phi$ is a *fixpoint literal*; formulas of the form $\mu X.\, \phi$ are also called *eventualities*. A formula ψ is a *free subformula* of ϕ if ψ occurs as a subformula in ϕ that is not in the scope of any fixpoint operator. An occurrence of a subformula is *unguarded* if it is not in the scope of a modality. The *rank* $\mathsf{rk}(\phi)$ of a formula ϕ is the maximal nesting depth of modal operators in ϕ. We write $\phi(\psi)$ for the result of substituting ψ for X in ϕ. The obvious *substitution lemma* states that, for a T-coalgebra (C, ξ) and a subset $A \subseteq C$, $[\![\phi(\psi)]\!](A) = [\![\phi]\!]([\![\psi]\!](A))$.

Definition 5 (Fischer-Ladner closure)**.** The *Fischer-Ladner closure* $\mathcal{FL}(\phi)$ of a formula ϕ is defined as $\mathcal{FL}(\phi) = \mathcal{FL}(\phi)^{\phi}$, where $\mathcal{FL}(\phi)^{\psi}$ is defined, for all formulas ψ, by recursion over ϕ:

$$\begin{aligned}
\mathcal{FL}(\phi_1 \wedge \phi_2)^{\psi} &= \{(\phi_1 \wedge \phi_2)(\psi)\} \cup \mathcal{FL}(\phi_1)^{\psi} \cup \mathcal{FL}(\phi_2)^{\psi} \\
\mathcal{FL}(\phi_1 \vee \phi_2)^{\psi} &= \{(\phi_1 \vee \phi_2)(\psi)\} \cup \mathcal{FL}(\phi_1)^{\psi} \cup \mathcal{FL}(\phi_2)^{\psi} \\
\mathcal{FL}(\heartsuit\phi_1)^{\psi} &= \{(\heartsuit\phi_1)(\psi)\} \cup \mathcal{FL}(\phi_1)^{\psi} \\
\mathcal{FL}(\eta X.\phi_1)^{\psi} &= \{\eta X.\phi_1\} \cup \mathcal{FL}(\phi_1)^{\eta X.\phi_1} \\
\mathcal{FL}(X)^{\psi} &= \emptyset
\end{aligned}$$

Remark 6. The above definition differs slightly from, e.g., the one used for CTL by Emerson and Halpern [19] in that it retains Boolean combinations when unfolding fixpoints. We still have $|\mathcal{FL}(\phi)^{\psi}| \leq |\phi|$, by induction over ϕ.

Definition 7 (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 as $\Psi \vdash_{PL} \phi$) 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 fixpoints are guarded). More generally, Ψ *propositionally entails* a finite set Φ of formulas (written $\Psi \vdash_{PL} \Phi$) if $\Psi \vdash_{PL} \bigwedge \Phi$ (so $\vdash_{PL}$ is different from the $\vdash$ used in sequent calculi).

In logics like CTL and ATL, where the recursion variable always occurs under exactly one modality in the fixpoint operators (e.g. $A[\phi U \psi] = \mu X.\, (\psi \vee (\phi \wedge \Box X))$), the main problem in the design of algorithms are eventualities, such as AU-formulas. In flat μ-calculi, we also have to take into account (unfolded) subformulas of fixpoint formulas; e.g. when building a tableau for the formula $\mu X.\, (p \vee \Diamond\Diamond X)$, which states that p is reachable in an even number of steps, we will, in odd steps, need to take care of the formula $\Diamond(\mu X.\, (p \vee \Diamond\Diamond X))$. We

call such formulas *deferrals*, and represent them in decomposed form – in the example, as $(\Diamond X, \mu X.\,(p \vee \Diamond\Diamond X))$. Formal details are as follows.

Lemma 8. *Let $\chi = \eta X.\psi$ and $\alpha \leq \psi$. Then $\chi \leq \alpha(\chi)$ iff X is free subformula of α, and in this case $\alpha(\chi) \leq \psi(\chi)$ iff α is free subformula of ψ.*

Definition 9 (Deferral)**.** Given a fixpoint literal $\chi = \eta X.\psi$, a *χ-deferral* is a pair (α, χ) such that

$$\alpha \leq \psi \text{ and } \chi \leq \alpha(\chi) \leq \psi(\chi)$$

(equivalently, X is free in α and α is free subformula of ψ). A *modal χ-deferral* is a deferral of the shape $(\heartsuit\beta, \chi)$. We say that (α, χ) *induces* the formula $\alpha(\chi)$. We denote the set of all (modal) χ-deferrals by $dfr(\chi)$ ($mdfr(\chi)$). If χ is an eventuality (i.e. $\eta = \mu$), then χ-deferrals are also called *μ-deferrals*.

Example 10. For the mentioned formula $e = \mu X.(p \vee \Diamond\Diamond X)$, we have $dfr(e) = \{(p \vee \Diamond\Diamond X, e), (\Diamond\Diamond X, e), (\Diamond X, e), (X, e)\}$ with induced formulas $p \vee \Diamond\Diamond e$, $\Diamond\Diamond e$, $\Diamond e$ and e.

III. The Global Caching Algorithm

We next describe our tableau method for the flat coalgebraic μ-calculus, which as advertised in the introduction is generic, single-pass, and optimal. In the following, we fix a closed formula ϕ_0; we put $\mathcal{F} = \mathcal{FL}(\phi_0)$ and $evs = \{\mu X.\psi \in \mathcal{F}\}$.

Definition 11 (Nodes)**.** A *node* is an $\mathcal{F}$-sequent. A *state node* is a node that consists only of modal literals; we denote the set of nodes by $\mathbf{S}$, and the set of state nodes by $\mathbf{V}$.

The next definitions capture how eventualities are discharged (*finished*) or deferred, respectively.

Definition 12 (Sufficiency)**.** Given an eventuality $e = \mu X.\psi$, we say that a set D of deferrals is *sufficient* for an e-deferral (α, e) at a node s, and write $D \vdash_s \alpha(e)$, if

$$\{\beta(e) \mid (\beta, e) \in D\} \cup N(s, e) \vdash_{PL} \alpha(e)$$

where $N(s, e)$ is the subset of s of formulas *not* induced by an e-deferral. We write $\vdash_s \alpha(e)$ for $\emptyset \vdash_s \alpha(e)$. For a set D' of deferrals, we write $D \vdash_s D'$ if $D \vdash_s \alpha(e)$ for each $(\alpha, e) \in D'$.

Definition 13 (Finishing nodes)**.** Let $e = \mu X.\psi$, and let (α, e) be an e-deferral. The set of *finishing nodes* for (α, e) is defined as

$$F(\alpha, e) = \{s \in \mathbf{S} \mid s \vdash_{PL} \alpha(e) \Rightarrow \vdash_s \alpha(e)\}.$$

For a node s we denote by $D(s)$ the set of μ-deferrals (α, e) for which s is *not* finishing, i.e. $s \vdash_{PL} \alpha(e)$ but $\nvdash_s \alpha(e)$.

Example 14. In CTL, the above notions are simple and as expected. E.g. for $e = AF\phi = \mu X.\,(\phi \vee \Box X)$, a node s such that $s \vdash_{PL} e$ is finishing for the e-deferral (X, e) iff $s \vdash_{PL} \phi$, and otherwise $(\Box X, e) \in D(s)$.

For a more complex example, consider eventualities $e_1 = \mu X.\psi_1$ and $e_2 = \mu X.\psi_2$ in the flat relational μ-calculus, where

$$\psi_1 = p \vee \Box\Diamond X \quad \text{and} \quad \psi_2 = q \vee \Diamond X.$$

Take state nodes $v_1 = \{p, q\}$ and $v_2 = \{\Box\Diamond e_1,\ \Diamond e_2\}$. We have $\vdash_{v_1} \psi_1(e_1)$ and $\vdash_{v_1} \psi_2(e_2)$ so that $v_1 \in F(\psi_1, e_1)$ and $v_1 \in F(\psi_2, e_2)$, and hence $D(v_1) = \emptyset$. On the other hand, $v_2 \notin F(\psi_1, e_1)$ since $N(v_2, e_1) = \{\Diamond e_2\} \nvdash_{PL} \psi_1(e_1)$. Similarly, $v_2 \notin F(\psi_2, e_2)$, i.e. $D(v_2) = \{(\Box\Diamond X, e_1), (\Diamond X, e_2)\}$.

We proceed to describe the rules of the calculus, where we distinguish between rules and their instances.

Definition 15 (Rule instance)**.** A *rule instance* is a tuple $(\Gamma_0/\Gamma_1 \ldots \Gamma_n)$ consisting of a *premise* Γ_0 and *conclusions* $\Gamma_1, \ldots, \Gamma_n$, where the Γ_i are $\mathcal{F}(\Lambda)$-sequents.

The rules of our calculus are the *modal rules* in the given set $\mathcal{R}$ of one-step tableau rules (Section I), and additionally the *non-modal rules* $\mathcal{R}_p = \{(\wedge), (\vee), (\eta)\}$ (the axiom rule is a modal rule; cf. Example 4). The *rule instances* of the non-modal rules are of the form

$$(\wedge)\ \frac{\Gamma, \phi \wedge \psi}{\Gamma, \phi, \psi} \quad (\vee)\ \frac{\Gamma, \phi \vee \psi}{\Gamma, \phi \quad \Gamma, \psi} \quad (\eta)\ \frac{\Gamma, \eta X.\phi}{\Gamma, \phi(\eta X.\phi)}\ (\eta \in \{\nu, \mu\})$$

(the formulation of the non-modal rules themselves is immaterial). The *rule instances* of a modal rule $(\Gamma_0/\Gamma_1 \ldots \Gamma_n)$ are

$$\frac{\Delta, \Gamma_0\sigma}{\Gamma_1\sigma \quad \ldots \quad \Gamma_n\sigma}$$

where σ is a substitution and Δ a sequent; that is, the rule instances have a weakening context Δ built in. We abuse $\mathcal{R}$ and $\mathcal{R}_p$ to denote also the respective sets of rule instances.

The following notions govern the expansion step of the global caching algorithm.

Definition 16 (Conclusions)**.** For a node $s \in S$ and a set $\mathcal{S}$ of rule instances, the set of *conclusions* of s under $\mathcal{S}$ is

$$Cn(\mathcal{S}, s) = \{\{\Gamma_1, \ldots, \Gamma_n\} \in \mathcal{P}(\mathbf{S}) \mid (s/\Gamma_1 \ldots \Gamma_n) \in \mathcal{S}\}.$$

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

The salient point of our algorithm is that it tracks how eventualities and more generally deferrals are propagated by applications of rules (similarly to *traces* in tableaux for the full μ-calculus). This is formally captured as follows.

Definition 17 (Recreation)**.** A node w *recreates* α *from* $(\heartsuit\alpha, v)$ for a state node $v \in \mathbf{V}$ if w is a conclusion of a modal rule instance $(\Delta, \Gamma_0\sigma/\Gamma_1\sigma \ldots \Gamma_n\sigma)$ with $\Delta, \Gamma_0\sigma = v$ such that $\heartsuit b \in \Gamma_0$, $b \in \Gamma_i$ and $\sigma(b) = \alpha$ for some variable $b \in P$. For a non-state node $s \in \mathbf{S}$, $\alpha \in w$, and $\beta \in s$, w *recreates* α *from* (β, s) if w is the conclusion of a non-modal rule instance $(s/\Gamma_1 \ldots \Gamma_n)$ and either β has one of the forms α, $\alpha \vee \gamma$, $\gamma \vee \alpha$, $\alpha \wedge \gamma$, $\gamma \wedge \alpha$, or $\beta = \eta X.\psi$ and $\alpha = \psi[X \mapsto \beta]$. We put

$$Rec_m(\alpha, \heartsuit\alpha, v) = \{t \in \mathbf{S} \mid t \text{ recreates } \alpha \text{ from } (\heartsuit\alpha, v)\}$$
$$Rec_p(\alpha, \beta, s) = \{t \in \mathbf{S} \mid t \text{ recreates } \alpha \text{ from } (\beta, s)\}.$$

For the propagation step of our algorithm, we consider two types of monotone functionals that we call *proof transitionals*, one that captures the standard intuition that a node is satisfiable

if every rule that applies to it has a satisfiable conclusion, and, building on the first, a more sophisticated one that additionally tracks recreation of formulas by rule applications.

Definition 18 (Proof transitionals)**.** Let $S \subseteq \mathbf{S}$ be a set of nodes. The proof transitionals $f : \mathcal{P}(S) \to \mathcal{P}(S)$ and $g : \mathcal{P}(S) \to \mathcal{P}(S)$ are defined by

$$f(S') = \{s \in S \mid \forall \Sigma \in Cn(s).\, \exists \Gamma \in \Sigma.\, \Gamma \in S'\}$$
$$g(S') = \{s \in S \mid \exists \Sigma \in Cn(s).\, \forall \Gamma \in \Sigma.\, \Gamma \in S'\}$$

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

Definition 19 (Tracking proof transitionals)**.** Given a set D of μ-deferrals and nodes $s, t \in \mathbf{S}$, if s is a state, we put

$$K(D, s, t) = \{(\alpha, e) \in D(t) \mid \exists (\heartsuit\alpha, e) \in D.\ t \in Rec_m(\alpha(e), \heartsuit\alpha(e), s)\}$$

and if s is not a state, we put

$$K(D, s, t) = \{(\alpha, e) \in D(t) \mid \exists (\beta, e) \in D.\ t \in Rec_p(\alpha(e), \beta(e), s)\}.$$

In words, $K(D, s, t)$ is the set of μ-deferrals that originate from D and that t recreates from s but does not finish.

For $S \subseteq \mathbf{S}$, $n \geq 0$ and a nonempty set D of μ-deferrals, we then define the *tracking transitionals* $\hat{f}^n_D : \mathcal{P}(S) \to \mathcal{P}(S)$ and $\hat{g}^n_D : \mathcal{P}(S) \to \mathcal{P}(S)$ (with base set S) by

$$\hat{f}^0_D(S') = \emptyset$$
$$\hat{f}^{n+1}_D(S') = \{s \in S \mid \forall \Sigma \in Cn(s).\, \exists \Gamma \in \Sigma.\ \Gamma \in S' \cap \hat{f}^n_{K(D,s,\Gamma)}(S')\}$$

for $S' \subseteq S$, and

$$\hat{g}^0_D(S') = S$$
$$\hat{g}^{n+1}_D(S') = \{s \in S \mid \exists \Sigma \in Cn(s).\, \forall \Gamma \in \Sigma.\ \Gamma \in S' \cup \hat{g}^n_{K(D,s,\Gamma)}(S')\}$$

For $D = \emptyset$, we put $\hat{f}^n_\emptyset(S') = f(S')$ and $\hat{g}^n_\emptyset(S') = g(S')$.

In game-theoretic terms, the set $\hat{f}^n_D(S')$ can be understood as the set of nodes at which *Eloise* (who tries to establish satisfiability) has a strategy to finish all deferrals from D within at most n steps while staying in S' throughout.

Fact 20. *For sets $D' \subseteq D$ of μ-deferrals, $S' \subseteq S$, and $n \geq 0$,*

1) *$\hat{f}^n_D(S') \subseteq \hat{f}^n_{D'}(S')$ and $\hat{g}^n_D(S') \supseteq \hat{g}^n_{D'}(S')$.*
2) *$\hat{f}^n_D(S') \subseteq \hat{f}^{n+1}_D(S')$ and $\hat{g}^n_D(S') \supseteq \hat{g}^{n+1}_D(S')$.*

Example 21. Take eventualities $e_1 = \mu X.(p \vee \Box\Box X)$ and $e_2 = \mu X.(q \vee \Diamond X)$ in the flat relational μ-calculus, and a state node $v = \{q, \Box\Box e_1, \Diamond e_2\}$; note that $D(v) = \{(\Box\Box X, e_1), (\Diamond X, e_2)\}$. Recall that $\mathcal{R}$ has rules $\{\Diamond b_0, \Box b_1, \ldots, \Box b_n / b_0, b_1, \ldots, b_n\}$ (Example 4). One instance of the rule for $n = 1$ matches v: it takes b_1 to be $\Box e_1$ and b_0 to be e_2. We obtain as conclusion the node $s = \{\Box e_1, e_2\}$; observe that $s \in Rec_m(\Box e_1, \Box\Box e_1, v) \cap Rec_m(e_2, \Diamond e_2, v)$ and $D(s) = \{(\Box X, e_1), (X, e_2)\}$. Then $K(D(v), v, s) = \{(\Box X, e_1), (X, e_2)\}$ (which in this case equals $D(s)$).

Notation 22. Let D be a set of μ-deferrals, and let $S' \subseteq S$. By Fact 20.2 there exist (minimal) n, m such that $\hat{f}^{n+1}_D(S') = \hat{f}^n_D(S')$ and $\hat{g}^{m+1}_D(S') = \hat{g}^m_D(S')$. We put

$$\hat{f}_D(S') = \hat{f}^n_D(S') \quad \text{and} \quad \hat{g}_D(S') = \hat{g}^m_D(S').$$

Definition 23 (Propagation)**.** Given a base set $S \subseteq \mathbf{S}$, we define $\mathfrak{E}, \mathfrak{A} : \mathcal{P}(S) \to \mathcal{P}(S)$ by

$$\mathfrak{E}(S') = \{s \in S \mid s \in \hat{f}_{D(s)}(S')\}$$
$$\mathfrak{A}(S') = \{s \in S \mid s \in \hat{g}_{D(s)}(S')\}$$

for $S' \subseteq S$. We define E_S and A_S (or just E, A) as

$$E = \nu\mathfrak{E} \quad \text{and} \quad A = \mu\mathfrak{A}.$$

Fact 24. *If $S_1 \subseteq S_2$, then $E_{S_1} \subseteq E_{S_2}$ and $A_{S_1} \subseteq A_{S_2}$.*

Fact 25. *Let $S \subseteq \mathbf{S}$ be fully expanded. Then $E_S = \overline{A_S}$.*

Following the standard setup of global caching, our algorithm maintains as global variables a set G of nodes and a subset U of *unexpanded* nodes. Along the way, it computes sets E_G of *satisfiable* nodes, and A_G of *unsatisfiable* nodes as per Definition 23. These grow monotonically and hence can be maintained in an incremental fashion in implementations.

Algorithm 26 (Global caching)**.** Decide satisfiability of a closed formula ϕ_0.

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

As with every global caching algorithm, the potential for closing the tableau early is realized by combining good choices of nodes for expansion in Step 2 with judicious intermediate propagation in Step 3. Here, the non-determinism works in our favour, i.e. affords maneuvering space for heuristic optimization: the algorithm delivers correct results regardless of the order in which nodes are expanded and of how often propagation is triggered. If no intermediate propagation steps are performed, then the algorithm will build a fully expanded set of nodes that is independent of the order in which nodes are expanded, and then calculate the subset of successful nodes in the final propagation step. For purposes of the soundness proof, we note as an immediate consequence of Facts 24 and 25

Fact 27. *If a run of Algorithm 26 without intermediate propagation steps is successful for some input formula ϕ_0, then any run of the algorithm with input ϕ_0 is successful.*

Example 28. To illustrate the potential avoidance of full expansion enabled by single-pass tableau systems, consider the

CTL formula (shortly writing $\Box$ for AX)

$$\phi = \bigwedge_{i \leq n} \Box^i q \wedge AG(q \to \psi) \wedge E[q \, U \, \chi],$$

interpreted, as usual, over serial frames, where $\psi \wedge \chi$ is unsatisfiable for reasons that a typical two-pass algorithm will detect only in the second pass, e.g. because ψ forces infinite deferral of an eventuality in χ. A simple example of this kind is $\psi = EG\neg p$, $\chi = AFp$ but of course there are much more complicated (and harder to detect) examples. The left part of ϕ dooms any attempt to finish the eventuality $E[q \, U \, \chi]$ before the $n+1$-st step. In a two-pass tableau one would nevertheless need to create, in every step, a disjunctive branch attempting just that, possibly at high computational cost. Contrastingly, given the right heuristics, the global caching algorithm can instead opt to defer $E[q \, U \, \chi]$ for n steps (e.g. on the basis of noticing that things will become easier after the boxes run out) and only then satisfy χ.

IV. Soundness

We now show that the global caching algorithm is sound, i.e. answers 'yes' if the input formula is satisfiable. To this end, we consider the set of nodes constructed by the algorithm when run on a satisfiable formula. We show that every node that is contained in the theory of a state in the given model is successful, by induction over the number of modal steps that are required in the model to satisfy all eventualities in a state node. We require satisfaction of the target formula in a coalgebra that needs only finite unfoldings of deferrals:

Definition 29. For a formula ψ, we define $\psi^0 = X$ and $\psi^{n+1} = \psi(\psi^n)$.

Definition 30. We say that a coalgebra $\mathcal{C}$ is *stabilizing* if for each state x in $\mathcal{C}$ and each μ-deferral (α, e) such that $x \models \alpha(e)$, with $e = \mu X.\psi$, there is $n \geq 0$ such that $x \models \alpha(\psi^n(\bot))$.

Remark 31. If the coalgebraic type functor T is *finitary*, i.e. imposes finite branching, then every T-coalgebra is stabilizing. For the general case, we import the finite model property of the coalgebraic μ-calculus [26]: finite coalgebras are clearly stabilizing, so every satisfiable formula is satisfiable in a stabilizing coalgebra. Note we import only finiteness, not the bound on the size of models.

For the rest of the section, fix (by the previous remark, w.l.o.g.) a stabilizing coalgebra $\mathcal{C} = (C, \xi)$ satisfying the target formula ϕ_0 in some state. To show that every run of the algorithm with input ϕ_0 is successful, it suffices by Fact 27 to show that a run without intermediate propagation is successful. Let $S \subseteq \mathbf{S}$ be the fully expanded set of nodes constructed by all such runs (i.e. the smallest fully expanded set containing the node $\{\phi_0\}$).

Definition 32 (Unfolding)**.** Given a state $x \in C$, an eventuality $e = \mu X.\psi$, and an e-deferral (α, e) with $x \models \alpha(e)$, we define the *unfolding* $unf(\alpha(e), x)$ of (α, e) at x as

$$unf(\alpha(e), x) = \alpha(\psi^n)$$

for the least n such that $x \models \alpha(\psi^n(\bot))$ (which exists by stabilization).

Definition 33 (Realization)**.** The set of $\mathcal{C}$*-realized* nodes is

$$M = \{s \in S \mid \exists x \in C. \forall \phi \in s. \, x \models_C \phi\}.$$

Definition 34 (Rank)**.** Given a set D of μ-deferrals and a state $x \in C$ such that $x \models \alpha(e)$ for each $(\alpha, e) \in D$, we put

$$\mathsf{rk}(D, x) = \max\{\mathsf{rk}(unf(\alpha(e), x)) \mid (\alpha, e) \in D\}.$$

For $s \in M$ and a set $D \subseteq D(s)$ of deferrals, we put

$$\mathsf{rk}(D, s) = \min\{\mathsf{rk}(D, x) \mid \forall \phi \in s. \, x \models \phi\}.$$

Theorem 35 (Soundness)**.** *If ϕ_0 is satisfiable, then Algorithm 26 returns 'Yes' on input ϕ_0.*

Proof: It suffices to show that $\mathcal{C}$-realized nodes are successful, i.e. $M \subseteq E_S = \nu\mathfrak{E}$. We use coinduction, i.e. show that M is a postfixpoint of $\mathfrak{E}$, i.e. $s \in \hat{f}_{D(s)}(M)$ for all $s \in M$. This is by induction over the triple $(\mathsf{rk}(D(s), s), u_f(s), u_p(s))$ in lexicographic order where $u_f(s)$ and $u_p(s)$ denote the number of unguarded occurrences of fixpoint and propositional operators in s, respectively. ∎

V. Completeness

We next show *completeness*, i.e. that a formula is satisfiable if the algorithm answers 'yes'. We build our model by first extracting an enriched structure, a *tableau*, whose states are copies of the successful state nodes constructed in a run of the algorithm. We then show, roughly, that one can build a coalgebraic transition structure on the tableau nodes in such a way that the transitions at each node involve only (copies of) states that contributed to the underlying state being successful under the tracking proof transitionals, i.e. bring all deferrals closer to being finished. In the model construction, we combine focussing [15], [16], time-outs [11], [9], and an array-of-dags construction similar in spirit to the original model construction for CTL [19], [17]. Note that all this happens only in the model construction; the algorithm itself uses neither foci nor time-outs.

We fix the set S of nodes constructed in a successful run of the algorithm, and put $V = E_S \cap \mathbf{V}$, i.e. V is the set of successful state nodes. The carrier of our model will consist of state nodes from V annotated with foci (sets of deferrals). We next fix the properties required of a graph structure L on a carrier labelled with formulas, in the notion of a *tableau*. We show that we can extract a tableau from a successful run of the algorithm and then establish that every tableau supports a model, which together will imply completeness.

Definition 36 (Tableau)**.** Let U be a set of *nodes* with a labelling function $l : U \to V$, 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 modal deferrals. We define $tp(D, n) = U$ for all n if $D = \emptyset$ and $tp(D, 0) = \emptyset$ if $D \neq \emptyset$. If $D \neq \emptyset$, then $tp(D, m+1)$ is the set of those nodes $v \in U$ such that, writing $Cn(v) = \{\Sigma_1, \ldots, \Sigma_n\}$, we have $L(v) = \{w_1, \ldots, w_n\}$ where for each i there exists $\Gamma \in \Sigma_i$ such that

- $l(w_i) \vdash_{PL} \Gamma$ and

- $w_i \in tp(D', m)$ for some $D' \subseteq D(w_i)$ with $D' \vdash_{w_i} K(D, v, \Gamma)$.

If for each node $v \in U$ there is a number m such that $v \in tp(D(v), m)$, then L is a *tableau*.

Roughly, $tp(D, m)$ can be understood as the set of all nodes in U that finish all deferrals in D within m modal steps.

Lemma 37 (Tableau existence)**.** *There exists a tableau L over a carrier W of size at most $|\phi_0| \cdot 4^{|\phi_0|}$.*

Proof (sketch): In a first step, we define a notion of *focussed timed-out DAGs* $G^e_{\mathbb{N}} = (W^e_{\mathbb{N}}, L^e_{\mathbb{N}})$ for eventualities $e \in evs$. The set $W^e_{\mathbb{N}}$ consists of nodes of the form (v, D, m) where $D \subseteq D(v)$ is a set of e-deferrals, the *focus*, and $v \in V \cap \hat{f}^m_D(E_S)$; the number m is called the *time-out*. The successor relation $L^e_{\mathbb{N}}$ relates (v, D, m) to nodes $(w_1, D'_1, m-1), \ldots, (w_n, D'_n, m-1)$ essentially ensuring that $(v, D, m) \in tp(D, m)$ in notation as in Definition 36. Nodes with empty focus are frontier nodes. We show that for each $e \in evs$, a focussed timed-out DAG $G^e_{\mathbb{N}}$ can be extracted from V.

We next eliminate the time-outs by picking, for each (v, D), the node (v, D, m) with the smallest time-out, i.e. finishing its focus as quickly as possible. We thus obtain for each $e \in evs$ a *focussed DAG* $G^e = (W^e, L^e)$, with nodes of the form (v, D).

We combine the focussed DAGs into a tableau (of the claimed size) by fixing a cyclic ordering of evs, and attaching in place of each frontier node $(w, \emptyset)$ in G^e the node $(w, D(w) \cap mdfr(e'))$ in $G^{e'}$ for the next eventuality e'; this is similar to the original construction for CTL [19], [17]. ■

Example 38. Consider the CTL-formula $\psi = G_1 \wedge G_2 \wedge G_3$ (writing $\Box, \Diamond$ for AX, EX for brevity) where

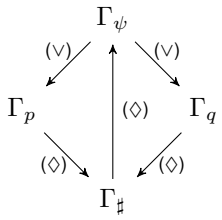
$$G_1 = AGF_1 \qquad G_2 = AGF_2 \qquad G_3 = AG\phi$$
$$F_1 = AF(p \wedge \Box\sharp) \qquad F_2 = AF(q \wedge \Box\sharp)$$
$$\phi = \neg(\sharp \wedge p) \wedge \neg(\sharp \wedge q) \wedge \neg(p \wedge q),$$

interpreted as usual over serial frames (i.e. coalgebras for the non-empty powerset functor), which means that we regard every sequent as implicitly containing $\Diamond\top$ (causing additional matches to Rule (1)). The algorithm constructs the following circular satisfiability proof, where $\Gamma := \Box G_1, \Box G_2, \Box G_3, \phi$ and where we indicate several consecutive applications of rules from a set $\mathcal{R}$ by writing $(\mathcal{R})^*$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{G_1 \wedge G_2 \wedge G_3}{\Gamma, (p \wedge \Box\sharp) \vee \Box F_1, (q \wedge \Box\sharp) \vee \Box F_2 := \Gamma_\psi}\,(\wedge,\nu,\mu)^*}{\Gamma, p, \Box\sharp, \Box F_2 := \Gamma_p}\,(\vee,\wedge)^* \qquad \dfrac{\Gamma, q, \Box\sharp, \Box F_1 := \Gamma_q}{\psi, \sharp, F_1}\,(\vee,\wedge)^*}{\psi, \sharp, F_2}\,(\Diamond) \qquad \dfrac{\Gamma_\sharp}{\Gamma_\sharp}\,(\Diamond)}{\Gamma, \sharp, \Box F_1, \Box F_2 := \Gamma_\sharp}\,(\wedge,\nu,\mu)^* \qquad \vdots}{\psi, F_1, F_2}\,(\Diamond)}{\Gamma_\psi}\,(\wedge,\nu)^*$$

Let S denote the set of labels of this proof and note that $E_S = S$. For instance, $\Gamma_\psi \in \hat{f}^n_{\{F_1,F_2\}}(E_S)$ for some n (namely, $n = 24$): Starting from Γ_ψ, it is possible to first reach $\Gamma_\sharp$ via Γ_p (while finishing F_1), pass Γ_ψ a second time and then finally branch right at the disjunction to reach Γ_q (and finish F_2).

We represent the relevant structure of the satisfiability proof in abstracted form as follows (where Γ_p, Γ_q and $\Gamma_\sharp$ are state nodes):



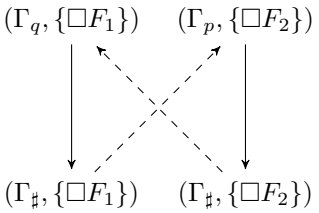
We write deferrals in abbreviated form for the remainder of this example, e.g. $\Box F_1$ denotes the deferral $(\Box X, F_1)$. The timed-out focussed DAG $G^{F_1}_{\mathbb{N}}$ is extracted from this structure as follows. The set $W^{F_1}_{\mathbb{N}}$ consists of nodes $(\Gamma_t, \emptyset, i)$ for $t \in \{p, q, \sharp\}$, $i \in \mathbb{N}$, as well as $(\Gamma_q, \{\Box F_1\}, i)$ where $\Gamma_q \in \hat{f}^i_{\{\Box F_1\}}$ and $(\Gamma_\sharp, \{\Box F_1\}, j)$ where $\Gamma_\sharp \in \hat{f}^j_{\{\Box F_1\}}$. The structure $L^{F_1}_{\mathbb{N}}$ contains transitions from $(\Gamma_q, \{\Box F_1\}, k)$ via $(\Gamma_\sharp, \{\Box F_1\}, l)$ to $(\Gamma_p, \emptyset, m)$ for suitable $k > l > m$. The focussed DAG G^{F_1} is obtained by selecting the nodes with the minimal time-out from $V^{F_1}_{\mathbb{N}}$, and has the shape

$$(\Gamma_q, \{\Box F_1\}) \to (\Gamma_\sharp, \{\Box F_1\}) \to (\Gamma_p, \emptyset).$$

Similarly, G^{F_2} has the shape

$$(\Gamma_p, \{\Box F_2\}) \to (\Gamma_\sharp, \{\Box F_2\}) \to (\Gamma_q, \emptyset).$$

Finally we combine the two DAGs to obtain a tableau. Transitions to nodes with empty focus are replaced by transitions to nodes with the same label but a new focus, i.e. the transitions $(\Gamma_\sharp, \{\Box F_1\}) \to (\Gamma_p, \emptyset)$ and $(\Gamma_\sharp, \{\Box F_2\}) \to (\Gamma_q, \emptyset)$ are replaced by $(\Gamma_\sharp, \{\Box F_1\}) \to (\Gamma_p, \{\Box F_2\})$ and $(\Gamma_\sharp, \{\Box F_2\}) \to (\Gamma_q, \{\Box F_1\})$, respectively, as indicated by the dashed arrows in the depiction of the tableau.



This is indeed a tableau; e.g. for the node $u = (\Gamma_\sharp, \{\Box F_2\})$ we have $u \in tp(D(u), n) = tp(\{\Box F_1, \Box F_2\}, n)$ for $n = 3$: We have a transition from u to $v = (\Gamma_q, \{\Box F_1\})$; note that Γ_q propositionally entails the conclusion of the single rule application to $\Gamma_\sharp$ and moreover finishes F_2, so it remains to show $v \in tp(\{\Box F_1\}, 2)$. We continue with two transitions via $(\Gamma_\sharp, \{\Box F_1\})$ (omitting details for this node) to $w = (\Gamma_p, \{\Box F_2\})$, where the last node finishes also F_2; that is, we end up having to show $w \in tp(\emptyset, 0)$, which holds by definition of tp.

We fix a tableau $L \subseteq W \times W$, and proceed to build a coalgebra over W. We intend that every state in this coalgebra satisfies

the formulas from its label, so we define a notion of intended extension of formulas:

Definition 39 (Pseudo-extension)**.** The *pseudo-extension* $\widehat{[\![\phi]\!]}$ of ϕ in W is

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

We extend this to sets Ψ of formulas by $\widehat{[\![\Psi]\!]} = \bigcap_{\psi\in\Psi} \widehat{[\![\psi]\!]}$.

The key condition bridging between tableaux and models is a strengthened form of *coherence* w.r.t. modal operators [32]:

Definition 40 (Strong coherence)**.** A T-coalgebra (W, ξ) is *strongly coherent* if for all $\heartsuit\phi \in \mathcal{F}$, $v \in W$, with $Cn(l(v)) = \{\Sigma_1, \dots, \Sigma_m\}$ and $L(v) = \{w_1, \dots, w_m\}$ as in Definition 36,

$$\heartsuit\phi \in l(v) \quad \text{implies} \quad \xi(v) \in [\![\heartsuit]\!](\widehat{[\![\phi]\!]} \cap L(v)_{\heartsuit\phi})$$

where

$$L(v)_{\heartsuit\phi} = \{w_i \mid \exists \Gamma \in \Sigma_i.\, w_i \in \widehat{[\![\Gamma]\!]}, \Gamma \in Rec_m(\phi, \heartsuit\phi, v)\}.$$

(Plain coherence requires only $\xi(v) \in [\![\heartsuit]\!]\widehat{[\![\phi]\!]}$.)

Lemma 41 (Strong existence lemma)**.** *Let L be a tableau. Then there exists a strongly coherent coalgebra over L.*

The proof of the strong existence lemma relies on one-step tableau completeness.

We fix a strongly coherent coalgebra $\mathcal{C} = (W, \xi)$, and show that in $\mathcal{C}$ every state satisfies the fixpoint literals that it claims to satisfy. This is easy for greatest fixpoints, and uses induction over the construction of the tracking proof transitionals for least fixpoints.

Definition 42 (Respect)**.** A formula ψ is *respected* if $\widehat{[\![\eta X.\psi]\!]} \subseteq [\![\eta X.\psi]\!]$ for each fixpoint literal $\eta X.\phi \leq \psi$.

Lemma 43. *Let $\psi \in \mathcal{F}$ be respected. Then $\widehat{[\![\psi]\!]} \subseteq [\![\psi]\!]$, $\widehat{[\![\nu X.\psi]\!]} \subseteq [\![\nu X.\psi]\!]$, and $\widehat{[\![\mu X.\psi]\!]} \subseteq [\![\mu X.\psi]\!]$.*

Proposition 44. *Every fixpoint literal is respected.*

From this, the truth lemma is immediate:

Lemma 45 (Truth lemma)**.** $\widehat{[\![\psi]\!]} \subseteq [\![\psi]\!]$ *for each* $\psi \in \mathcal{F}$.

Corollary 46 (Completeness)**.** *If a run of Algorithm 26 with input ϕ_0 returns 'Yes', then ϕ_0 is satisfiable.*

The model construction moreover implies a bound $2^{O(n)}$ on model size, the same as for CTL [17]:

Corollary 47. *Every satisfiable flat fixpoint formula ϕ_0 has a model of size at most $|\phi_0| \cdot 4^{|\phi_0|}$.*

The best previous bound known for ATL was $n^{O(n)} = 2^{O(n \log n)}$ [18].

VI. Complexity

As the global caching algorithm constructs only subsets of the Fischer-Ladner closure $\mathcal{FL}(\phi_0)$, it runs only through at most exponentially many iterations. Similarly, the propagation steps involve only exponentially many iterations in fixpoint computations. The algorithm therefore runs in ExpTime under mild assumptions on the set of modal rules – in fact, the same assumptions as used in previous work on reasoning in ExpTime-complete coalgebraic logics [26], [27] (and very similar ones as in work on reasoning in PSpace-complete ones [32], [37]). *We assume a reasonable size measure on* Λ inducing a *representation size* of formulas and sets of formulas for purposes of measuring the input size for the satisfiability problem. In particular we assume that numbers in probabilistic operators are *coded in binary* (the more stringent assumption compared to unary coding, as we are aiming for upper complexity bounds). The key requirement on the rule set is, then, that all rule matches to a given sequent can be represented by polynomially large codes, formally (rewording the exact conditions slightly in comparison to previous work):

Definition 48. The set $\mathcal{R}$ of modal rules is ExpTime-*tractable* if there exists a coding c of rules $(\Gamma_0/\Gamma_1 \dots \Gamma_n)$ as strings $c(\Gamma_0/\Gamma_1 \dots \Gamma_n)$ over some alphabet with the following properties. First, there exists a polynomial p such that for all P-sequents Γ, all rules $(\Gamma_0/\Gamma_1 \dots \Gamma_n)$, and all injective renamings σ such that $\Gamma_0\sigma \subseteq \Gamma$,

$$|c(\Gamma_0/\Gamma_1 \dots \Gamma_n)| \leq p(|\Gamma|)$$

(where $|\cdot|$ denotes string length on the left). Moreover, we can extract rules from codes in exponential time, and given $(\Gamma_0/\Gamma_1 \dots \Gamma_n)$ and a P-sequent Γ, we can compute, in exponential time, all injective σ such that $\Gamma_0\sigma \subseteq \Gamma$.

For all of our examples, ExpTime tractability has been established in previous work [32]; in most cases, it is a purely bureaucratic condition. E.g. for the relational μ-calculus, we can just code Rule (1) as itself. The only case in which tractability of the rule set is an actual issue is probabilistic fixpoint logic (Example 4.2); here, tractability relies on size bounds on solutions of linear inequalities [32].

Theorem 49. *If $\mathcal{R}$ is* ExpTime*-tractable, then Algorithm 26 decides the satisfiability problem of the associated flat coalgebraic μ-calculus in* ExpTime.

This bound is tight in all our example logics, i.e. the global caching algorithm has optimal complexity.

VII. Conclusion

We have presented a generic tableau method for flat coalgebraic μ-calculi that is single-pass and optimal, and supports global caching. Its main benefit is that it offers the chance of closing tableaux early, avoiding exponential run time in good cases. Our generic results instantiate to new algorithms for the flat fragments of the relational μ-calculus, probabilistic fixpoint logic, the alternating-time μ-calculus, and the monotone μ-calculus. Even for fragments of these such as CTL and alternating-time temporal logic ATL, our algorithm yields, to our best knowledge, the first time-out-free optimal single-pass decision procedure.

Future work concerns proof-theoretic consequences of our results in a dual sequent calculus, as well as the implementation of the algorithm within the coalgebraic reasoner COOL [40].

Acknowledgments: We wish to thank Rajeev Goré, Dirk Pattinson and Florian Widmann for helpful discussions, an anonymous benefactor for Example 38, and Erwin R. Catesbeiana for conceptual help on procrastinating eventualities.

REFERENCES

[1] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, Eds., *The Description Logic Handbook*. CUP, 2003.

[2] D. Kozen, "Results on the propositional μ-calculus," *Theoret. Comput. Sci.*, vol. 27, pp. 333–354, 1983.

[3] R. Alur, T. Henzinger, and O. Kupferman, "Alternating-time temporal logic," *J. ACM*, vol. 49, pp. 672–713, 2002.

[4] R. Parikh, "Propositional game logic," in *Foundations of Computer Science, FOCS 1983*. IEEE Computer Society, 1983.

[5] K. Larsen and A. Skou, "Bisimulation through probabilistic testing," *Inf. Comput.*, vol. 94, pp. 1–28, 1991.

[6] A. Heifetz and P. Mongin, "Probabilistic logic for type spaces," *Games and Economic Behavior*, vol. 35, pp. 31–53, 2001.

[7] R. Fagin and J. Y. Halpern, "Reasoning about knowledge and probability," *J. ACM*, vol. 41, pp. 340–367, 1994.

[8] R. Goré, "And-Or tableaux for fixpoint logics with converse: LTL, CTL, PDL and CPDL," in *Automated Reasoning, IJCAR 2014*, ser. LNCS, vol. 8562. Springer, 2014, pp. 26–45.

[9] L. Schröder and Y. Venema, "Flat coalgebraic fixed point logics," in *Concurrency Theory, CONCUR 2010*, ser. LNCS, vol. 6269. Springer, 2010, pp. 524–538.

[10] C. Cîrstea, C. Kupke, and D. Pattinson, "EXPTIME tableaux for the coalgebraic μ-calculus," in *Computer Science Logic, CSL 2009*, ser. LNCS, vol. 5771. Springer, 2009, pp. 179–193.

[11] L. Santocanale and Y. Venema, "Completeness for flat modal fixpoint logics," in *Logic for Programming, Artificial Intelligence, and Reasoning, LPAR 07*, ser. LNCS, vol. 4790. Springer, 2007, pp. 499–513.

[12] R. Goré and L. Nguyen, "EXPTIME tableaux for $\mathcal{ALC}$ using sound global caching," in *Description Logics, DL 2007*, ser. CEUR Workshop Proc., vol. 250, 2007.

[13] ——, "EXPTIME tableaux with global caching for description logics with transitive roles, inverse roles and role hierarchies," in *Automated Reasoning with Analytic Tableaux and Related Methods, TABLEAUX 2007*, ser. LNCS, vol. 4548. Springer, 2007, pp. 133–148.

[14] R. Goré and L. Postniece, "An experimental evaluation of global caching for $\mathcal{ALC}$ (system description)," in *Automated Reasoning, IJCAR 2008*, ser. LNCS, vol. 5195. Springer, 2008, pp. 299–305.

[15] M. Lange and C. Stirling, "Focus games for satisfiability and completeness of temporal logic," in *Logic in Computer Science, LICS 2001*. IEEE Computer Society, 2001, pp. 357–365.

[16] K. Brünnler and M. Lange, "Cut-free sequent systems for temporal logic," *J. Log. Algebr. Prog.*, vol. 76, pp. 216–225, 2008.

[17] E. A. Emerson, "Temporal and modal logic," in *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*. Elsevier, 1990, pp. 995–1072.

[18] S. Schewe, "Synthesis of distributed systems," Ph.D. dissertation, Universität des Saarlands, 2008.

[19] E. Emerson and J. Halpern, "Decision procedures and expressiveness in the temporal logic of branching time," *J. Comput. Sys. Sci.*, vol. 30, pp. 1–24, 1985.

[20] L. Zhang, U. Hustadt, and C. Dixon, "A resolution calculus for the branching-time temporal logic CTL," *ACM Trans. Comput. Log.*, vol. 15, 2014.

[21] O. Friedmann and M. Lange, "The modal μ-calculus caught off guard," in *Automated Reasoning with Analytic Tableaux and Related Methods, TABLEAUX 2011*, ser. LNCS, vol. 6793. Springer, 2011, pp. 149–163.

[22] V. Goranko and G. van Drimmelen, "Complete axiomatization and decidability of alternating-time temporal logic," *Theoret. Comput. Sci.*, vol. 353, pp. 93–117, 2006.

[23] D. Walther, C. Lutz, F. Wolter, and M. Wooldridge, "ATL is indeed EXPTIME-complete," *J. Log. Comput.*, vol. 16, pp. 765–787, 2006.

[24] V. Goranko and D. Shkatov, "Tableau-based decision procedures for logics of strategic ability in multiagent systems," *ACM Trans. Comput. Log.*, vol. 11, 2009.

[25] A. David, "TATL: Implementation of ATL tableau-based decision procedure," in *Automated Reasoning with Analytic Tableaux and Related Methods, TABLEAUX 2013*, ser. LNCS, vol. 8123. Springer, 2013, pp. 97–103.

[26] C. Cirstea, A. Kurz, D. Pattinson, L. Schröder, and Y. Venema, "Modal logics are coalgebraic," *Comput. J.*, vol. 54, pp. 31–41, 2011.

[27] 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 10*, ser. LNCS, vol. 6015. Springer, 2010, pp. 114–128.

[28] R. Goré, C. Kupke, D. Pattinson, and L. Schröder, "Global caching for coalgebraic description logics," in *Automated Reasoning, IJCAR 2010*, ser. LNCS, vol. 6173. Springer, 2010, pp. 46–60.

[29] D. Pattinson, "Expressive logics for coalgebras via terminal sequence induction," *Notre Dame J. Formal Logic*, vol. 45, pp. 19–33, 2004.

[30] L. Schröder, "A finite model construction for coalgebraic modal logic," *J. Log. Algebr. Prog.*, vol. 73, pp. 97–110, 2007.

[31] L. Schröder and D. Pattinson, "Rank-1 modal logics are coalgebraic," *J. Log. Comput.*, vol. 20, pp. 1113–1147, 2010.

[32] L. Schröder and D. Pattinson, "PSPACE bounds for rank-1 modal logics," *ACM Trans. Comput. Log.*, vol. 10, pp. 13:1–13:33, 2009.

[33] P. Wolper, "Temporal logic can be more expressive," *Inf. Control*, vol. 56, pp. 72–99, 1983.

[34] M. Pauly, "A modal logic for coalitional power in games," *J. Log. Comput.*, vol. 12, pp. 149–166, 2002.

[35] R. Parikh, "The logic of games and its applications," *Ann. Discr. Math.*, vol. 24, pp. 111–140, 1985.

[36] D. Peleg, "Concurrent dynamic logic," *J. ACM*, vol. 34, pp. 450–479, 1987.

[37] R. Myers, D. Pattinson, and L. Schröder, "Coalgebraic hybrid logic," in *Foundations of Software Science and Computation Structures, FoSSaCS 2009*, ser. LNCS, vol. 5504. Springer, 2009, pp. 137–151.

[38] L. Schröder, D. Pattinson, and C. Kupke, "Nominals for everyone," in *Int. Joint Conf. Artificial Intelligence, IJCAI 09*. AAAI, 2009, pp. 917–922.

[39] P. Abate, R. Goré, and F. Widmann, "One-pass tableaux for computation tree logic," in *Logic for Programming, Artificial Intelligence, and Reasoning, LPAR 07*, ser. LNCS. Springer, 2007, pp. 32–46.

[40] D. Gorín, D. Pattinson, L. Schröder, F. Widmann, and T. Wißmann, "COOL – a generic reasoner for coalgebraic hybrid logics (system description)," in *Automated Reasoning, IJCAR 2014*, ser. LNCS, vol. 8562. Springer, 2014, pp. 396–402.