The Complexity of Enriched μ-Calculi

Piero A. Bonatti[1⋆], Carsten Lutz[2], Aniello Murano[1⋆], and Moshe Y. Vardi[3⋆⋆]

[1] Università di Napoli "Federico II", Dipartimento di Scienze Fisiche, 80126 Napoli, Italy
[2] TU Dresden, Institute for Theoretical Computer Science, 01062 Dresden, Germany
[3] Microsoft Research and Rice University, Dept. of Computer Science, TX 77251-1892, USA

Abstract. The *fully enriched μ-calculus* is the extension of the propositional μ-calculus with inverse programs, graded modalities, and nominals. While satisfiability in several expressive fragments of the fully enriched μ-calculus is known to be decidable and EXPTIME-complete, it has recently been proved that the full calculus is undecidable. In this paper, we study the fragments of the fully enriched μ-calculus that are obtained by dropping at least one of the additional constructs. We show that, in all fragments obtained in this way, satisfiability is decidable and EXPTIME-complete. Thus, we identify a family of decidable logics that are maximal (and incomparable) in expressive power. Our results are obtained by introducing two new automata models, showing that their emptiness problems are EXPTIME-complete, and then reducing satisfiability in the relevant logics to this problem. The automata models we introduce are *two-way graded alternating parity automata* over infinite trees (2GAPT) and *fully enriched automata* (FEA) over infinite forests. The former are a common generalization of two incomparable automata models from the literature. The latter extend alternating automata in a similar way as the fully enriched μ-calculus extends the standard μ-calculus.

1 Introduction

The *μ-calculus* is a propositional modal logic augmented with least and greatest fixpoint operators [Koz83]. It is often used as a target formalism for embedding temporal and modal logics with the goal of transferring computational and model theoretic properties such as the EXPTIME upper complexity bound. *Description logics (DLs)* are a family of knowledge representation languages that originated in artificial intelligence [BM+03]. DLs currently receive considerable attention, which is mainly due to their use as an ontology language in prominent applications such as the semantic web [BHS02]. Notably, DLs have recently been standardized as the ontology language OWL by the W3C committee. It has been pointed out by several authors that, by embedding DLs into the μ-calculus, we can identify DLs that are of very high expressive power, but computationally well-behaved [CGL01,SV01,KSV02]. When putting this idea to work, we face the problem that modern DLs such as the ones underlying OWL include several constructs that cannot easily be translated into the μ-calculus. Most importantly, these

⋆ Supported in part by the European Network of Excellence REWERSE, IST-2004-506779.
⋆⋆ Supported in part by NSF grants CCR-0311326 and ANI-0216467, by BSF grant 9800096, and by Texas ATP grant 003604-0058-2003. Work done in part while this author was visiting the Isaac Newton Institute for Mathematical Science, Cambridge, UK, as part of a Special Programme on Logic and Algorithm.

	Inverse progr.	Graded mod.	Nominals	Complexity
fully enriched μ-calculus	x	x	x	undecidable
full graded μ-calculus	x	x		EXPTIME (1ary/2ary)
full hybrid μ-calculus	x		x	EXPTIME
hybrid graded μ-calculus		x	x	EXPTIME (1ary/2ary)
graded μ-calculus		x		EXPTIME (1ary/2ary)

Fig. 1. Enriched μ-calculi and previous results.

constructs are inverse programs, graded modalities, and nominals. Intuitively, inverse programs allow to travel backwards along accessibility relations [Var98], nominals are propositional variables interpreted as singleton sets [SV01], and graded modalities enable statements about the number of successors and predecessors of a state [KSV02]. All of the mentioned constructs are available in the DLs underlying OWL.

The extension of the μ-calculus with these constructs induces a family of enriched μ-calculi. These calculi may or may not enjoy the attractive computational properties of the original μ-calculus: on the one hand, it has been shown that satisfiability in a number of the enriched calculi is decidable and EXPTIME-complete [CGL01,SV01,KSV02]. On the other hand, it has recently been proved by Bonatti and Peron that satisfiability is undecidable in the *fully enriched μ-calculus*, i.e., the logic obtained by extending the μ-calculus with all of the above constructs simultaneously [BP04]. In computer science logic, it has always been a major research goal to identify decidable logics that are as expressive as possible. Thus, the above results raise the question of maximal decidable fragments of the fully enriched μ-calculus. In this paper, we study this question in a systematic way by considering all fragments of the fully enriched μ-calculus that are obtained by dropping at least one of inverse programs, graded modalities, and nominals. We show that, in all these fragments, satisfiability is decidable and EXPTIME-complete. Thus, we identify a whole family of decidable logics that have maximum (incomparable) expressivity.

The relevant fragments of the fully enriched μ-calculus are shown in Fig. 1 together with the complexity of their satisfiability problem. The results shown in gray are already known from the literature: EXPTIME-completeness of satisfiability in the full hybrid μ-calculus has been shown in [SV01]; under the assumption that the numbers inside graded modalities are coded in unary, the same result was proved for the full graded μ-calculus in [CGL01]; finally, the same was also shown for the (non-full) graded μ-calculus in [KSV02] under the assumption of binary coding. In this paper, we prove EXPTIME-completeness of the full graded μ-calculus and the hybrid graded μ-calculus. In both cases, we allow numbers to be coded in binary (techniques such as those of [CGL01] involve an exponential blow-up when numbers are coded in binary).

Our results are based on the automata-theoretic approach. We introduce *fully enriched automata (FEAs)*, which run on infinite forests and use a parity acceptance condition. Intuitively, these automata generalize alternating automata on infinite trees in a similar way as the fully enriched μ-calculus extends the standard μ-calculus: FEAs can move up to a node's predecessor (by analogy with inverse programs), move down to at least n or all but n successors (by analogy with graded modalities), and jump directly to the roots of the input forest (which are the analogues of nominals). We prove that

the emptiness problem is decidable for fully enriched automata and then show how to reduce to this problem satisfiability in the hybrid graded and the full graded μ-calculi, exploiting the forest model property enjoyed by these logics. Observe that decidability of the emptiness problem for FEAs does not contradict the undecidability of the fully enriched μ-calculus: the latter does not enjoy a forest model property [BP04], and hence satisfiability cannot be decided using forest-based FEAs.

To show that the emptiness problem for FEAs is in EXPTIME, we introduce an additional automata model: *two-way graded parity tree automata (2GAPTs)*. These automata are interesting in their own right because they generalize in a natural way two existing, but incomparable automata models: two-way alternating tree automata (2APT) [Var98] and graded parity tree automata (GAPT) [KSV02]. We give a polynomial reduction of the emptiness problem for FEAs to that for 2GAPTs, and then show containment in EXPTIME for the 2GAPT emptiness problem by a reduction to the emptiness of graded nondeterministic parity tree automata (GNPT) as introduced in [KSV02].

Due to space limitations, most of the proofs are omitted. The interested reader can find them in the accompanying technical report [BL+06].

2 Preliminaries

Let AP, Var, $Prog$, and Nom be finite and pairwise disjoint sets of *atomic propositions*, *propositional variables*, *atomic programs*, and *nominals*. A *program* is an atomic program a or its converse a^-. The set of *formulas of the fully enriched μ-calculus* is the smallest set such that (*i*) **true** and **false** are formulas; (*ii*) p and $\neg p$, for $p \in AP \cup Nom$, are formulas; (*iii*) $x \in Var$ is a formula; (*iv*) if φ_1 and φ_2 are formulas, α is a program, n is a non-negative integer, and y is a propositional variable, then the following are also formulas: $\varphi_1 \vee \varphi_2$, $\varphi_1 \wedge \varphi_2$, $\langle n, \alpha \rangle \varphi_1$, $[n, \alpha] \varphi_1$, $\mu y.\varphi_1(y)$, and $\nu y.\varphi_1(y)$. Observe that we use positive normal form, i.e., negation is applied only to atomic propositions.

We call μ and ν *fixpoint operators* and use λ to denote a fixpoint operator μ or ν. A propositional variable y occurs *free* in a formula if it is not in the scope of a fixpoint operator, and *bounded* otherwise. A *sentence* is a formula that contains no free variables. For a formula $\lambda y.\varphi(y)$, we write $\varphi(\lambda y.\varphi(y))$ to denote the formula that is obtained by one-step unfolding, i.e. replacing each free occurrence of y in φ with $\lambda y.\varphi(y)$. We refer often to the *graded modalities* $\langle n, \alpha \rangle \varphi_1$ and $[n, \alpha] \varphi_1$ as *atleast formulas* and *allbut formulas* and assume that the integers in these operators are given in binary coding: the contribution of n to the length of the formulas $\langle n, \alpha \rangle \varphi$ and $[n, \alpha] \varphi$ is $\lceil \log n \rceil$ rather than n. We refer to fragments of the fully enriched μ-calculus using the names from Fig. 1.

The semantics of the fully enriched μ-calculus is defined with respect to a *Kripke structure*, i.e., a tuple $K = \langle W, R, L \rangle$ where W is a non-empty set of *states*, $R : Prog \to 2^{W \times W}$ assigns to each atomic program a transition relation over W, and $L : AP \cup Nom \to 2^W$ assigns to each atomic proposition and nominal a set of states such that the sets assigned to nominals are singletons. To deal with inverse programs, we extend R as follows: for each $a \in Prog$, set $R(a^-) = \{(v, u) : (u, v) \in R(a)\}$. If $(w, w') \in R(\alpha)$, we say that w' is an α *successor* of w. Informally, an *atleast* formula $\langle n, \alpha \rangle \varphi$ holds at a state w of a Kripke structure K if φ holds at least in $n+1$ α successors of w. Dually, the *allbut* formula $[n, \alpha] \varphi$ holds in a state w of a Kripke structure K

if φ holds in all but at most n α successors of w. Note that $\neg\langle n,\alpha\rangle\varphi$ is equivalent to $[n,\alpha]\neg\varphi$, and that the modalities $\langle\alpha\rangle\varphi$ and $[\alpha]\varphi$ of the standard μ-calculus can be expressed as $\langle 0,\alpha\rangle\varphi$ and $[0,\alpha]\varphi$, respectively.

To formalize semantics, we introduce valuations. Given a Kripke structure $K = \langle W, R, L\rangle$ and a set $\{y_1,\ldots,y_n\}$ of variables in Var, a *valuation* $\mathcal{V} : \{y_1,\ldots,y_n\} \to 2^W$ is an assignment of subsets of W to the variables $y_1,\ldots,y_n$. For a valuation $\mathcal{V}$, a variable y, and a set $W' \subseteq W$, we denote by $\mathcal{V}[y \leftarrow W']$ the valuation obtained from $\mathcal{V}$ by assigning W' to y. A formula φ with free variables among $y_1,\ldots,y_n$ is interpreted over the structure K as a mapping φ^K from valuations to 2^W, i.e., $\varphi^K(\mathcal{V})$ denotes the set of points that satisfy φ under valuation $\mathcal{V}$. The mapping φ^K is defined inductively as follows:

- $\mathbf{true}^K(\mathcal{V}) = W$ and $\mathbf{false}^K(\mathcal{V}) = \emptyset$;
- for $p \in AP \cup Nom$, we have $p^K(\mathcal{V}) = L(p)$ and $(\neg p)^K(\mathcal{V}) = W \setminus L(p)$;
- for $y \in Var$, we have $y^K(\mathcal{V}) = \mathcal{V}(y)$;
- $(\varphi_1 \wedge \varphi_2)^K(\mathcal{V}) = \varphi_1^K(\mathcal{V}) \cap \varphi_2^K(\mathcal{V})$ and $(\varphi_1 \vee \varphi_2)^K(\mathcal{V}) = \varphi_1^K(\mathcal{V}) \cup \varphi_2^K(\mathcal{V})$;
- $(\langle n,\alpha\rangle\varphi)^K(\mathcal{V}) = \{w : |\{w' \in W : (w,w') \in R(\alpha) \text{ and } w' \in \varphi^K(\mathcal{V})\}| \geq n+1\}$;
- $([n,\alpha]\varphi)^K(\mathcal{V}) = \{w : |\{w' \in W : (w,w') \in R(\alpha) \text{ and } w' \notin \varphi^K(\mathcal{V})\}| \leq n\}$;
- $(\mu y.\varphi(y))^k(\mathcal{V}) = \bigcap\{W' \subseteq W : \varphi^K([y \leftarrow W']) \subseteq W'\}$;
- $(\nu y.\varphi(y))^k(\mathcal{V}) = \bigcup\{W' \subseteq W : W' \subseteq \varphi^K([y \leftarrow W'])\}$.

Let $K = \langle W, R, L\rangle$ be a Kripke structure and φ a sentence. For a state $w \in W$, we say that φ *holds* at w in K, denoted $K, w \models \varphi$, if $w \in \varphi^K$. K is a *model* of φ if there is a $w \in W$ such that $K, w \models \varphi$. Finally, φ is *satisfiable* if it has a model.

In the remainder of this section, we show that the full graded μ-calculus has a tree model property, and that the hybrid graded μ-calculus has a forest model property. A *forest* is a set $F \subseteq \mathbb{N}^+$ such that if $x \cdot c \in F$ where $x \in \mathbb{N}^+$ and $c \in \mathbb{N}$, then also $x \in F$. The elements of F are called *nodes*, and the strings consisting of a single natural number are the *roots* of F. For each root $r \in F$, the set $T = \{r \cdot x \mid x \in \mathbb{N}^* \text{ and } r \cdot x \in F\}$ is a *tree* of F (the tree *rooted in* r). For every $x \in F$, the nodes $x \cdot c \in F$ where $c \in \mathbb{N}$ are the *successors* of x, and x is their *predecessor*. The number of successors of x is called the *degree* of x, and is denoted by $deg(x)$. The degree of a forest is the maximum of the degrees of a node in the forest and the number of roots.

We call a Kripke structure $K = \langle W, R, L\rangle$ a *forest structure* if (*i*) W is a forest, (*ii*) $(w,v) \in \bigcup_{a \in Prog} R(a)$ iff $(w,v) \in W^2$ and w is either a predecessor or a successor of v, and (*iii*) $R(\alpha) \cap R(\beta) = \emptyset$ for all $\alpha, \beta \in Prog \cup \{a^- \mid a \in Prog\}$ with $\alpha \neq \beta$. K is *directed* if $(w,v) \in \bigcup_{a \in Prog} R(a)$ implies that v is a successor of w. If W consists of a single tree then we call K a *tree structure*.

We call $K = \langle W, R, L\rangle$ a *quasi forest structure* if $\langle W, R', L\rangle$ is a forest structure, where $R'(a) = R(a) \setminus (W \times \mathbb{N})$ for all $a \in Prog$ (i.e., K becomes a forest structure after deleting all the edges entering a root of W). K is *directed* if $\langle W, R', L\rangle$ is. The *degree* of K is the degree of W. Note that forest and tree structures are quasi forest structures. A *forest model* (resp. *tree model*, *quasi forest model*) of φ is a forest (resp. tree, quasi forest) structure $K = \langle W, R, L\rangle$ such that φ and the nominals in φ hold at some (not necessarily different) roots of W. In what follows, a formula φ *counts* up to b if the maximal integer in atleast and allbut restrictions used in φ is $b-1$.

Theorem 1. *Let φ be a sentence of the full graded μ-calculus such that φ has ℓ* atleast *subsentences and counts up to b. If φ is satisfiable, then φ has a tree model whose degree is at most $\ell(b+1)$.*

In contrast to the full graded μ-calculus, the hybrid graded μ-calculus does not enjoy the tree model property. This is for example witnessed by the formula

$$o \wedge \langle 0,a\rangle(p_1 \wedge \langle 0,a\rangle(p_2 \wedge \cdots \langle 0,a\rangle(p_{n-1} \wedge \langle 0,a\rangle o)\cdots))$$

which generates a cycle of length at most n if the atomic propositions are enforced to be mutually disjoint. However, we can follow [SV01] to show that every satisfiable formula of the hybrid graded μ-calculus has a quasi forest model.

Theorem 2. *Let φ be a sentence of the hybrid graded μ-calculus such that φ has k nominals, ℓ* atleast *subsentences and counts up to b. If φ is satisfiable, then φ has a directed quasi forest model K whose degree is at most* $\max\{k+1, \ell(b+1)\}$.

3 Enriched automata

Nondeterministic automata on infinite trees are a variation of nondeterministic automata on finite and infinite words, see [Tho90] for an introduction. *Alternating automata*, as first introduced in [MS87], are a generalization of nondeterministic automata. Intuitively, while a nondeterministic automaton that visits a node x of the input tree sends one copy of itself to each of the successors of x, an alternating automaton can send several copies of itself to the same successor. In the two-way paradigm [Var98], an automaton can send a copy of itself to its predecessor, too. In graded automata [KSV02], the automaton can specify a number n of successors to which copies of itself are sent, without specifying which successors these exactly are. The fully enriched automata that we are introducing in the next subsection work on infinite forests, include all of the above features, and additionally have the ability to send a copy of themselves to the roots of the forest.

3.1 Fully enriched automata

We start with some preliminaries. Let $F \subseteq \mathbb{N}^+$ be a forest and x a node in F. As a convention, we take $x \cdot \varepsilon = x$, $(x \cdot c) \cdot -1 = x$, and $\varepsilon \cdot -1$ as undefined. We call x a *leaf* if it has no successors. A *path* π in F is a minimal set $\pi \subseteq F$ such that some root r of F is contained in π and for every $x \in \pi$, either x is a leaf or there exists a unique $c \in F$ such that $x \cdot c \in \pi$. Given an alphabet Σ, a Σ-labeled forest is a pair $\langle F, V\rangle$, where F is a forest and $V : F \to \Sigma$ maps each node of F to a letter in Σ.

For a given set Y, let $B^+(Y)$ be the set of positive Boolean formulas over Y (i.e., Boolean formulas built from elements in Y using $\wedge$ and $\vee$), where we also allow the formulas **true** and **false** and $\wedge$ has precedence over $\vee$. For a set $X \subseteq Y$ and a formula $\theta \in B^+(Y)$, we say that X satisfies θ iff assigning **true** to elements in X and assigning false to elements in $Y \setminus X$ makes θ true. For $b > 0$, let $\langle [b]\rangle = \{\langle 0\rangle, \langle 1\rangle, \ldots, \langle b\rangle\}$, $[[b]] = \{[0], [1], \ldots, [b]\}$, and $D_b = \langle [b]\rangle \cup [[b]] \cup \{-1, \varepsilon, \langle root\rangle, [root]\}$.

A fully enriched automaton is an automaton in which the transition function δ maps a state q and a letter σ to a formula in $B^+(D_b \times Q)$. Intuitively, an atom $(\langle n\rangle, q)$ (resp. $([n], q)$) means that the automaton sends copies in state q to $n+1$ (resp. all but n) different successors of the current node, (ε, q) means that the automaton sends a copy (in state q) to the current node, $(-1, q)$ means that the automaton sends a copy to the predecessor of the current node, and $(\langle \mathit{root}\rangle, q)$ and $([\mathit{root}], q)$ mean that the automaton sends a copy to some, respectively all of the roots of the forest. When, for instance, the automaton is in state q, reads a node x, and

$$\delta(q, V(x)) = (-1, q_1) \wedge ((\langle \mathit{root}\rangle, q_2) \vee ([\mathit{root}], q_3)),$$

it sends a copy in state q_1 to the predecessor and either sends a copy in state q_2 to one of the roots or a copy in state q_3 to all roots.

Formally, a *fully enriched automaton* (FEA, for short) is a tuple $A = \langle \Sigma, b, Q, \delta, q_0, \mathcal{F}\rangle$, where Σ is the input alphabet, $b > 0$ is a counting bound, Q is a finite set of states, $\delta : Q \times \Sigma \to B^+(D_b \times Q)$ is a transition function, $q_0 \in Q$ is an initial state, and $\mathcal{F}$ is the acceptance condition. A *run* of A on an input Σ-labeled forest $\langle F, V\rangle$ is a tree $\langle T_r, r\rangle$ in which each node is labeled by an element of $F \times Q$. Intuitively, a node in T_r labeled by (x, q) describes a copy of the automaton in state q that reads the node x of F. Runs start in the initial state and satisfy the transition relation. Thus, a run $\langle T_r, r\rangle$ with root z has to satisfy the following: (*i*) $r(z) = (c, q_0)$ for some root c of F and (*ii*) for all $y \in T_r$ with $r(y) = (x, q)$ and $\delta(q, V(x)) = \theta$, there is a (possibly empty) set $S \subseteq D_b \times Q$, such that S satisfies θ, and for all $(d, s) \in S$, the following hold:

- If $d \in \{-1, \varepsilon\}$, then $x \cdot d$ is defined and there is $j \in \mathbb{N}$ such that $y \cdot j \in T_r$ and $r(y \cdot j) = (x \cdot d, s)$;
- If $d = \langle n\rangle$, then there are distinct $i_1, \ldots, i_{n+1} \in \mathbb{N}$ such that for all $1 \leq j \leq n+1$, there is $j' \in \mathbb{N}$ such that $y \cdot j' \in T_r$, $x \cdot i_j \in F$, and $r(y \cdot j') = (x \cdot i_j, s)$;
- If $d = [n]$, then there are distinct $i_1 \ldots, i_{deg(x)-n} \in \mathbb{N}$ such that for all $1 \leq j \leq deg(x) - n$, there is $j' \in \mathbb{N}$ such that $y \cdot j' \in T_r$, $x \cdot i_j \in F$, and $r(y \cdot j') = (x \cdot i_j, s)$;
- If $d = \langle \mathit{root}\rangle$, then for some root $c \in F$ and some $j \in \mathbb{N}$ such that $y \cdot j \in T_r$, it holds that $r(y \cdot j) = (c, s)$;
- If $d = [\mathit{root}]$, then for all roots $c \in F$ there exists $j \in \mathbb{N}$ such that $y \cdot j \in T_r$ and $r(y \cdot j) = (c, s)$.

A run $\langle T_r, r\rangle$ is *accepting* if all its infinite paths satisfy the acceptance condition. We consider here the *parity acceptance condition*, where $\mathcal{F} = \{F_1, F_2, \ldots, F_k\}$ is such that $F_1 \subseteq F_2 \subseteq \ldots \subseteq F_k = Q$. The number k of sets in $\mathcal{F}$ is called the *index* of the automaton. Given a run $\langle T_r, r\rangle$ and an infinite path $\pi \subseteq T_r$, let $\mathit{Inf}(\pi) \subseteq Q$ be such that $q \in \mathit{Inf}(\pi)$ iff there are infinitely many $y \in \pi$ for which $r(y) \in F \times \{q\}$. A path π *satisfies* a parity acceptance condition $\mathcal{F} = \{F_1, F_2, \ldots, F_k\}$ iff there is an even i for which $\mathit{Inf}(\pi) \cap F_i \neq \emptyset$ and $\mathit{Inf}(\pi) \cap F_{i-1} = \emptyset$. An automaton *accepts* a forest iff there exists an accepting run of the automaton on the forest. We denote by $\mathcal{L}(\mathcal{A})$ the set of all Σ-labeled forests that A accepts.

The *emptiness problem* for FEAs is to decide, given a FEA A, whether $\mathcal{L}(\mathcal{A}) = \emptyset$. To decide this problem, we first reduce it to the emptiness problem of a more restricted automata model: a *two-way graded alternating parity tree automaton (2GAPT)* is a

FEA that accepts trees (instead of forests) and cannot jump to the root of the input tree, i.e., it does not support directions $\langle root\rangle$ and $[root]$ in the transition relation. For each FEA A, there exists a 2GAPT A' that accepts a tree encoding of A's language. A Σ-labeled forest $\langle F, V\rangle$ is *encoded* by a $\Sigma \cup \{root\}$-labeled tree $\langle T, V'\rangle$ with root $z \notin F$ iff $root \notin \Sigma$, $T = \{z\} \cup \{z \cdot c \mid c \in F\}$, and V' satisfies:

- $V'(z) = \{root\}$,
- $V'(z \cdot x) = V(x)$ for all $x \in F$.

Then, we can prove the following.

Theorem 3. *Let A be a FEA running on Σ-labeled forests with n states, index k and counting bound b. There exists a 2GAPT A' running on $\Sigma \cup \{root\}$-labeled trees ($root \notin \Sigma$) with $3n + 1$ states, index k, and counting bound b such that A' accepts a labeled tree $\langle T, V\rangle$ iff A accepts the forest encoded by $\langle T, V\rangle$.*

3.2 Graded nondeterministic parity tree automata

To decide the emptiness problem of 2GAPTs, we use a reduction to the emptiness problem of graded nondeterministic parity tree automata as introduced in [KSV02]. In the following, we define these automata and state some results concerning them.

For an integer b, a *b-bound* is a pair in $B_b = \{(>, 0), (\leq, 0), (>, 1), (\leq, 1), \ldots, (>, b), (\leq, b)\}$. For a set Y, we use $B(Y)$ to denote the set of all Boolean formulas over atoms in Y. Each formula $\theta \in B(Y)$ induces a set $sat(\theta) \subseteq 2^Y$ such that $x \in sat(\theta)$ iff x satisfies θ. For an integer $b \geq 0$, a *b-counting constraint* for 2^Y is a relation $C \subseteq B(Y) \times B_b$. A tuple $t = \langle x_1, \ldots, x_m\rangle \in (2^Y)^m$ satisfies the b-counting constraint C if for all $\langle \theta, \xi\rangle \in C$, the tuple t satisfies ξ with respect to $sat(\theta)$, that is, when θ is paired with $(>, n)$, at least $n + 1$ elements of t should satisfy θ, and when θ is paired with $(\leq, n)$, at most n elements in the tuple satisfy θ. We use $\mathcal{C}(Y, b)$ to denote the set of all b-counting constraints for 2^Y.

A *graded nondeterministic parity tree automaton* (GNPT, for short) is a tuple $A = \langle \Sigma, b, Q, \delta, q_0, \mathcal{F}\rangle$ where Σ, b, q_0, and $\mathcal{F}$ are as in 2GAPT, $Q \subseteq 2^Y$ is the set of states (i.e., Q is encoded by a finite set of variables), and $\delta : Q \times \Sigma \to \mathcal{C}(Y, b)$ maps a state and a letter to a b-counting constraint for 2^Y. Given a GNPT A, a *run* of A on a Σ-labeled tree $\langle T, V\rangle$ rooted in z is a Q-labeled tree $\langle T, r\rangle$ such that $r(z) = q_0$ and for every $x \in T$, the tuple $\langle r(x \cdot 1), \ldots, r(x \cdot deg(x))\rangle$ satisfies $\delta(r(x), V(x))$. The run $\langle T, r\rangle$ is *accepting* if all its infinite paths satisfy the parity acceptance condition.

We need two special cases of GNPT: FORALL automata and SAFETY automata. In FORALL automata, for each $q \in Q$ and $\sigma \in \Sigma$ there is $s \in Q$ such that $\delta(q, \sigma) = \{\langle(\neg\theta_s), (\leq, 0)\rangle\}$, where $\theta_s \in B(Y)$ is such that $sat(\theta_s) = \{s\}$. Thus, a FORALL automaton is a notational variant of a deterministic tree automaton, where the transition function maps q and σ to $\langle s, \ldots, s\rangle$. In SAFETY automata, there is no acceptance condition, and all runs are accepting. Note that this does not mean that SAFETY automata accept all trees, as it may be that on some trees the automaton does not have a run. We will need the following results concerning GNPTs.

Lemma 1. [KSV02] *Given a* FORALL *GNPT A_1 with n_1 states and index k, and a* SAFETY *GNPT A_2 with n_2 states and counting bound b, we can define a GNPT A with $n_1 n_2$ states, index k, and counting bound b, such that $\mathcal{L}(A) = \mathcal{L}(A_1) \cap \mathcal{L}(A_2)$.*

Theorem 4. [KSV02] *Given a GNPT $A = \langle \Sigma, b, Q, \delta, q_0, \mathcal{F} \rangle$ with n states, index k, counting bound b, and $|\Sigma| = \ell$, the nonemptiness problem for A can be solved in time $n^k \ell (b+2)^{O(n(n+2+k \log nk))}$.*

4 The emptiness problem for 2GAPT

We show that emptiness of the language accepted by a 2GAPT can be decided in ExpTime. A corresponding lower bound is inherited from alternating tree automata [KVW00].

Let $A = \langle \Sigma, b, Q, \delta, q_0, \mathcal{F} \rangle$ be a 2GAPT. Recall that $D_b = \langle [b] \rangle \cup [[b]] \cup \{-1, \varepsilon\}$ and $\delta : Q \times \Sigma \to B^+(D_b \times Q)$. A *strategy tree* for A is a $2^{Q \times D_b \times Q}$-labeled tree $\langle T, \mathsf{str} \rangle$. Intuitively, the function str (from now on called *strategy*) maps each node of the tree to a set of transitions. For each label $w = \mathsf{str}(x)$, we define $head(w) = \{q : (q, c, q') \in w\}$ as the set of *sources* of w. A strategy tree $\langle T, \mathsf{str} \rangle$ is on a Σ-labeled tree $\langle T, V \rangle$, if $q_0 \in head(\mathsf{str}(root(T)))$ and for each node $x \in T$ and state q, the set $\{(c, q') : (q, c, q') \in \mathsf{str}(x)\}$ satisfies $\delta(q, V(x))$ (where $root(T)$ denotes the root of T). Intuitively, by choosing the atoms that are going to be satisfied for a node x, $\mathsf{str}(x)$ removes the nondeterminism in δ.

A *promise tree* for the automaton A on a Σ-labeled tree $\langle T, V \rangle$ is a $2^{Q \times Q}$-labeled tree $\langle T, \mathsf{pro} \rangle$. Intuitively, in a run that proceeds according to pro (in the following called *promise*), if a node $x \cdot i$ has $(q, q') \in \mathsf{pro}(x \cdot i)$ and the run visits its parent x in state q and proceeds by choosing an atom $(\langle n \rangle, q')$ or $([n], q')$, then $x \cdot i$ is among the successors of x that inherit q'. For each label $w = \mathsf{pro}(x)$, we also define $head(w) = \{q : (q, q') \in w\}$ as the set of *sources* of w.

Consider a 2GAPT A, a Σ-labeled tree $\langle T, V \rangle$, a strategy tree $\langle T, \mathsf{str} \rangle$ and a promise tree $\langle T, \mathsf{pro} \rangle$ for A on $\langle T, V \rangle$. A $(T \times Q)$-labeled tree $\langle T_r, r \rangle$ is *consistent* with str and pro if $\langle T_r, r \rangle$ suggests a possible run of A on $\langle T, V \rangle$ such that whenever the run $\langle T_r, r \rangle$ is in state q as it reads a node $x \in T$, the strategy $\mathsf{str}(x)$ is defined, the run proceeds according to the elements of $\mathsf{str}(x)$ having q as source, and it delivers requirements to each successor $x \cdot j$ according to the elements in $\mathsf{pro}(x \cdot j)$ also having q as source. Formally, $\langle T_r, r \rangle$ is consistent with str and pro iff the following hold:

- $r(root(T_r)) = (root(T), q_0)$;
- for each node y in T_r with $r(y) = (x, q)$, $\mathsf{str}(x)$ is defined and for all $(q, c, q') \in \mathsf{str}(x)$, the following hold:
  - If $c = -1$ or $c = \varepsilon$, then $x \cdot c$ is defined and there is $j \in \mathbb{N}$ such that $y \cdot j \in T_r$ and $r(y \cdot j) = (x \cdot c, q')$;
  - If $c = \langle n \rangle$ or $c = [n]$, then for each $j \in \mathbb{N}$ with $(q, q') \in \mathsf{pro}(x \cdot j)$, there is $j' \in \mathbb{N}$ such that $y \cdot j' \in T_r$ and $r(y \cdot j') = (x \cdot j, q')$.

Note that since the counting constraints in $\mathsf{str}(x)$ may not be satisfied, $\langle T_r, r \rangle$ may not be a legal run.

Consider a strategy tree $\langle T, \mathsf{str} \rangle$ and a promise tree $\langle T, \mathsf{pro} \rangle$ on a Σ-labeled tree $\langle T, V \rangle$. We say that pro *fulfills* str for V if the states promised to be visited by pro satisfy the obligations induced by str as it runs on V. Formally, pro fulfills str for V if for every node $x \in T$, the following hold:

- For every $(q, \langle n \rangle, q') \in \mathsf{str}(x)$, at least $n+1$ successors $x \cdot j$ of x have $(q, q') \in \mathsf{pro}(x \cdot j)$;
- for every $(q, [n], q') \in \mathsf{str}(x)$, at least $deg(x) - n$ successors $x \cdot j$ of x have $(q, q') \in \mathsf{pro}(x \cdot j)$.

Consider a 2GAPT A, a strategy tree $\langle T, \mathsf{str} \rangle$ and promise tree $\langle T, \mathsf{pro} \rangle$ on a Σ-labeled tree $\langle T, V \rangle$. A sequence $(x_0, q_0), (x_1, q_1) \ldots$ is a *trace* induced by str and pro if x_0 is the root of T (notice that q_0 is the initial state of A) and, for each $i \geq 0$, one of the following holds:

- $q_i \notin head(\mathsf{str}(x_i))$ and (x_i, q_i) is the last pair in the trace;
- there is $(q_i, c, q_{i+1}) \in \mathsf{str}(x_i)$ with $c = -1$ or $c = \varepsilon$, $x_i \cdot c$ defined, and $x_{i+1} = x_i \cdot c$;
- $\mathsf{str}(x_i)$ contains $(q_i, \langle n \rangle, q_{i+1})$ or $(q_i, [n], q_{i+1})$, there exists $j \in \mathbb{N}$ with $x_{i+i} = x_i \cdot j$, $x_{i+i} \in T$, and $(q, q') \in \mathsf{pro}(x_{i+1})$.

It is not difficult to see that a sequence of pairs of nodes of T and states of A starting with $(root(T), q_0)$ is a trace induced by a strategy and a promise for A on a Σ-labeled tree $\langle T, V \rangle$ if a run $\langle T_r, r \rangle$ on $\langle T, V \rangle$, which is consistent with both the strategy and the promise, has a path π labeled with the trace. We say that a strategy tree $\langle T, \mathsf{str} \rangle$ and a promise $\langle T, \mathsf{pro} \rangle$ are *good* for $\langle T, V \rangle$ if all the infinite traces induced by str and pro satisfy the acceptance condition $\mathcal{F}$. In [KSV02] it has been shown that a necessary and sufficient condition for a tree to be accepted by a one-way GAPT is to have a strategy tree and a promise tree good for the input tree, with the promise fulfilling the strategy. We establish the same result with respect to the notions of strategy tree and promise tree as introduced above for 2GAPTs.

Theorem 5. *A 2GAPT A accepts $\langle T, V \rangle$ iff there exist a strategy tree $\langle T, \mathsf{str} \rangle$ and a promise tree $\langle T, \mathsf{pro} \rangle$ good for $\langle T, V \rangle$ such that* pro *fulfills* str *for V.*

Strategy and promise trees allow us to define a notion of a run for alternating automata that has the same tree structure as the underlying input tree, unlike the run $\langle T_r, r \rangle$. Since we want to translate 2GAPT into GNPT, we still have the problem that paths in a run can go both up and down. To restrict our attention to unidirectional paths, we extend to our setting the notion of annotation as defined in [Var98]. Annotations allow decomposing a path of a run into a downward path and several finite paths (*detour*) that come back to their origin (possibly in a loop).

Let $A = \langle \Sigma, b, Q, \delta, q_0, \mathcal{F} \rangle$ be a 2GAPT with $\mathcal{F} = \{F_1, \ldots, F_k\}$. Recall that $D_b = \langle [b] \rangle \cup [[b]] \cup \{-1, \varepsilon\}$. For each state $q \in Q$, let $index(q)$ be the minimal i such that $q \in F_i$. Consider a strategy tree $\langle T, \mathsf{str} \rangle$ and a promise tree $\langle T, \mathsf{pro} \rangle$ for A on a Σ-labeled tree $\langle T, V \rangle$, an *annotation tree* for A on $\langle T, \mathsf{str} \rangle$ and $\langle T, \mathsf{pro} \rangle$ is $\langle T, \mathsf{ann} \rangle$ where the *annotation* ann is a mapping $\mathsf{ann} : T \to 2^{Q \times \{1,\ldots,k\} \times Q}$ such that for every node $x \in T$ the following conditions hold:

- If $(q, \varepsilon, q') \in \mathsf{str}(x)$ then $(q, index(q'), q') \in \mathsf{ann}(x)$;
- if $(q, j', q') \in \mathsf{ann}(x)$ and $(q', j'', q'') \in \mathsf{ann}(x)$, then $(q, min(j', j''), q'') \in \mathsf{ann}(x)$;
- if $x = y \cdot i$, $(q, -1, q') \in \mathsf{str}(x)$, $(q', j, q'') \in \mathsf{ann}(y)$, $\mathsf{str}(y)$ contains $(q'', \langle n \rangle, q''')$ or $(q'', [n], q''')$, and $(q'', q''') \in \mathsf{pro}(x)$, then $(q, min(index(q'), j, index(q''')), q''') \in \mathsf{ann}(x)$;

– if $y = x \cdot i$, $\mathsf{str}(x)$ contains $(q, \langle n \rangle, q')$ or $(q, [n], q')$, $(q, q') \in \mathsf{pro}(y)$, $(q', j, q'') \in \mathsf{ann}(y)$, and $(q'', -1, q''') \in \mathsf{str}(y)$, then $(q, min(index(q'), j, index(q''')), q''') \in \mathsf{ann}(x)$.

Given an annotation tree $\langle T, \mathsf{ann} \rangle$ for A on $\langle T, \mathsf{str} \rangle$ and $\langle T, \mathsf{pro} \rangle$, a *downward path* π induced by str, pro, and ann is a sequence $(x_0, q_0, t_0), (x_1, q_1, t_1), \ldots$ of triples, where $x_0 = root(T)$, q_0 is the initial state of A, and for each i, x_i is in T, q_i is in Q, and t_i is either an element of $\mathsf{str}(x_i)$ or $\mathsf{ann}(x_i)$, such that: (*i*) either t_i is (q_i, c, q_{i+1}) for some $c \in [[b]] \cup [\langle b \rangle]$, $(q_i, q_{i+1}) \in \mathsf{pro}(x_i \cdot d)$ for some $d \in \mathbb{N}$, and $x_{i+1} = x_i \cdot d$; or (*ii*) t_i is (q_i, d, q_{i+1}), for $d \in \{1, \ldots, k\}$, and $x_{i+1} = x_i$. In the first case, we consider $index(t_i)$ as the minimal j such that $q_{i+1} \in F_j$ and, in the second case, $index(t_i) = d$. Moreover, for a downward path π, we consider $index(\pi)$ as the minimal index $index(t_i)$ for all t_i occurring infinitely often in π. We say that a downward path π violates $\mathcal{F}$ if $index(\pi)$ is odd. Given an annotation tree $\langle T, \mathsf{ann} \rangle$ for A on $\langle T, \mathsf{str} \rangle$ and $\langle T, \mathsf{pro} \rangle$, we say that ann is *accepting* if there is no downward path induced by str, pro, and ann that violates $\mathcal{F}$. Notice that a downward path π can also end in a loop where the last t_i is given by ann and π is accepting if $index(t_i)$ is even.

Theorem 6. *A 2GAPT A accepts $\langle T, V \rangle$ iff there exist a strategy tree $\langle T, \mathsf{str} \rangle$ and a promise tree $\langle T, \mathsf{pro} \rangle$ on $\langle T, V \rangle$, and an annotation tree $\langle T, \mathsf{ann} \rangle$ on $\langle T, \mathsf{str} \rangle$ and $\langle T, \mathsf{pro} \rangle$ such that* pro *fulfills* str *for V and* ann *is accepting.*

In the following, we combine the input tree, the strategy, the promise, and the annotation into one tree $\langle T, (V, \mathsf{str}, \mathsf{pro}, \mathsf{ann}) \rangle$. Given a signature Σ for the input tree, let Σ' denote the extended signature for the combined trees, i.e., $\Sigma' = \Sigma \times 2^{Q \times D_b \times Q} \times 2^{Q \times Q} \times 2^{Q \times \{1, \ldots k\} \times Q}$.

Theorem 7. *Let A be a 2GAPT running on Σ-labeled trees with n states, index k and counting bound b. There exists a GNPT A' running on Σ'-labeled trees with $2^{n(2+k \log nk)}$ states, index nk, and b-counting constraints such that A' accepts a tree iff A accepts its projection on Σ.*

5 EXPTIME upper bounds for enriched μ-calculi

We establish EXPTIME upper bounds for satisfiability in the full graded μ-calculus and the hybrid graded μ-calculus. p For the full graded μ-calculus, we give a polynomial translation of formulas φ into a 2GAPT A_φ that, roughly speaking, accepts the tree models of φ. By Theorem 1, we can thus decide satisfiability of φ by checking non-emptiness of $L(\mathcal{L}(\mathcal{A}_\varphi))$. There is a minor technical difficulty to be overcome: Kripke structures have labeled edges, while the trees accepted by 2GAPTs do not. This problem can be dealt with by moving the label from each edge to the target node of the edge. For this purpose, we introduce a new propositional symbol p_α for each program α. Let the *tree encoding* of a tree structure $K = \langle W, R, L \rangle$ be the labeled tree $\langle W, L^* \rangle$ such that $L^*(w) = L(w) \cup \{p_\alpha \mid \exists (v, w) \in R(\alpha)$ with w successor of v in $W\}$.

Theorem 8. *Given a sentence φ of the full graded μ-calculus that has ℓ atleast subsentences and counts up to b, we can construct a 2GAPT A_φ such that A_φ*

– *accepts exactly the tree encodings of tree models of φ with degree at most $\ell(b+1)$,*

– *has $|\varphi|$ states, index $|\varphi|$, and counting bound b.*

In the case of the hybrid graded μ-calculus, two additional difficulties have to be addressed. First, FEAs accept forests while the hybrid μ-calculus has only a *quasi* forest model property. This problem can be solved by introducing in node labels new propositional symbols $\uparrow_o^\alpha$ (not occurring in the input formula) that represent an α-labeled edge from the current node to the (unique) root node labeled by nominal o. Second, we have to take care of the interaction between graded modalities and the implicit edges encoded via propositions $\uparrow_o^a$. To this end, we need to know the following information before constructing the FEA: which "relevant" formulas are satisfied by each nominal and which nominals are equivalent. This information is provided by a guess, which we define as follows. The *closure* $cl(\varphi)$ of a sentence φ of the full graded μ-calculus is the smallest set of sentences satisfying the following:

– $\varphi \in cl(\varphi)$;
– if $\psi_1 \wedge \psi_2 \in cl(\varphi)$ or $\psi_1 \vee \psi_2 \in cl(\varphi)$, then $\{\psi_1, \psi_2\} \subseteq cl(\varphi)$;
– if $\langle n, \alpha\rangle\psi \in cl(\varphi)$ or $[n, \alpha]\psi \in cl(\varphi)$, then $\psi, \psi \wedge p_\alpha \in cl(\varphi)$;
– if $\lambda x.\psi(x) \in cl(\varphi)$, then $\psi(\lambda x.\psi(x)) \in cl(\varphi)$.
– if $\psi \in cl(\varphi)$, then $\neg\psi \in cl(\varphi)$, where $\neg\psi$ denotes the formula obtained from ψ by dualizing all operators and replacing every literal (i.e., atomic proposition or negation thereof) with its negation.

For a sentence φ, we use $|\varphi|$ to denote the *length* of φ with numbers inside graded modalities coded in binary. Formally, $|\varphi|$ is defined by induction on the structure of φ in a standard way, with $|\langle n, \alpha\rangle\psi| = \lceil log\ n\rceil + 1 + |\psi|$, and similarly for $|[n, \alpha]\psi|$. As proved in [Koz83], for every sentence φ, the number of elements in $cl(\varphi)$ is linear in the length φ.

A *guess* for φ is a pair $(t, \sim)$ where t assigns a subset $t(o) \subseteq cl(\varphi)$ to each $o \in Nom$, and $\sim$ is an equivalence relation on the set of nominals occurring in φ such that the following conditions are satisfied, for all formulas $\psi \in cl(\varphi)$ and nominals o, o' occurring in φ: (*i*) $\psi \in t(o)$ or $\neg\psi \in t(o)$, (*ii*) $o \in t(o)$, and (*iii*) $o \sim o'$ implies $t(o) = t(o')$. We construct a separate FEA $A_{\varphi,G}$ for each guess G for φ. Since the number of guesses is exponential in the length of φ, we get an EXPTIME decision procedure by constructing all of the FEAs and checking whether some of them accept a nonempty language. *Forest encodings* of forest models are defined similar to tree encodings of tree models with the additional property that $\uparrow_o^\alpha \in L^*(w)$ iff there exists $(w, v) \in R(\alpha)$ such that v is a root of W and $o \in L^*(v)$.

Theorem 9. *Given a sentence φ of the hybrid graded μ-calculus that has ℓ* atleast *subsentences, counts up to b, contains k nominals, and a guess $G = (t, \sim)$ for φ, we can construct a FEA $A_{\varphi,G}$ such that $A_{\varphi,G}$*

– *accepts exactly the forest encodings of the quasi forest models of φ having degree at most $\max\{k + 1, \ell(b + 1)\}$, and*
– *has $\mathcal{O}(|\varphi|^2)$ states, index $|\varphi|$, and counting bound b.*

Given a sentence of the full graded μ-calculus with ℓ at-least subformulas, we get by Theorems 7 and 8 a GNPT A_φ with the number of states n and index k bounded by $|\varphi|$, and $|\Sigma|$ and the counting bound b bounded by $2^{|\varphi|}$. While the latter are exponential in

$|\varphi|$, only n and k appear in the exponents in the expression in Theorem 4. This yields the desired EXPTIME upper bound. The lower bound is due to the fact that the μ-calculus is EXPTIME-hard [FL79]. For the hybrid graded μ-calculus, we can argue similarly using Theorems 3, 7, and 9.

Theorem 10. *The satisfiability problems of the full graded μ-calculus and the hybrid graded μ-calculus are* EXPTIME*-complete even if the numbers in the graded modalities are coded in binary.*

References

[BHS02] F. Baader, I. Horrocks, and U. Sattler. Description logics for the semantic web. *KI – Künstliche Intelligenz*, 3, 2002.

[BM+03] F. Baader, D.L. McGuiness, D. Nardi, and P. Patel-Schneider. *The Description Logic Handbook: Theory, implementation and applications*. Cambridge Univ. Press, 2003.

[BC96] G. Bhat and R. Cleaveland. Efficient local model-checking for fragments of the modal mu-calculus. In *Proc. of TACAS'96*, LNCS 1055, pages 107-126, 1996.

[BL+06] P.A. Bonatti, C. Lutz, A. Murano and M.Y. Vardi. The Complexity of Enriched μ-calculi. Chair for Automata Theory, Institute for Theoretical Computer Science, Dresden University of Technology, 2006, LTCS-Report, LTCS-06-02, Germany, see http://lat.inf.tu-dresden.de/research/reports.html.

[BP04] P.A. Bonatti and A. Peron. On the undecidability of logics with converse, nominals, recursion and counting. *Artificial Intelligence*, 158(1):75-96, 2004.

[CGL01] D. Calvanese, G. De Giacomo, and M. Lenzerini. Reasoning in expressive description logics with fixpoints based on automata on infinite trees. In *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI'99)*, pages 84-89, 1999.

[FL79] M.J. Fischer and R.E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and Systems Sciences*, Vol.18, pages 194-211, 1979.

[Jut95] C.S. Jutla. Determinization and memoryless winning strategies. *Information and Computation*, 133(2):117–134, 1997.

[Koz83] D. Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, Vol.27, pages 333-354, 1983.

[KSV02] O. Kupferman, U. Sattler, and M.Y. Vardi. The complexity of the Graded μ-calculus. In *Proc. of the 18th CADE* LNAI 2392, pages 423-437, 2002.

[KVW00] O. Kupferman, M.Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. *Journal of the ACM*, Vol.47(2), pages 312-360, 2000.

[MS87] D.E. Muller and P.E. Schupp. Alternating automata on infinite trees. *Theoretical Computer Science*, Vol.54, pages 267–276, 1987.

[Saf89] S. Safra. Complexity of automata on infinite objects. *PhD thesis*, Weizmann Institute of Science, Rehovot, Israel, 1989.

[SV01] U. Sattler and M. Y. Vardi. The hybrid mu-calculus. In *Proc. of IJCAR'01*, Vol.2083 of LNAI, pages 76-91. Springer Verlag, 2001.

[Tho90] W. Thomas. Automata on Infinite Objects. In *Handbook of Theoretical Computer Science*, pages 133 – 191, 1990.

[Tho97] W. Thomas. Languages, automata, and logic. In *Handbook of Formal Language Theory*, volume III, pages 389-455, G. Rozenberg and A. Salomaa editors, 1997.

[Var98] M.Y. Vardi. Reasoning about the Past with Two-Way Automata. In *Proc. of ICALP'98*, LNCS 1443, pages. 628–641, 1998.