The Complexity of the Graded μ-Calculus

Orna Kupferman[1], Ulrike Sattler[2], Moshe Y. Vardi[3*]

[1] School of Computer Science and Engineering, Hebrew University, Jerusalem, Israel
orna@cs.huji.ac.il
[2] Institut für Theoretische Informatik, TU Dresden, Germany
sattler@tcs.inf.tu-dresden.de
[3] Department of Computer Science, Rice University, Houston, TX 77251-1892, U.S.A.
vardi@cs.rice.edu

Abstract. In classical logic, existential and universal quantifiers express that there exists at least one individual satisfying a formula, or that all individuals satisfy a formula. In many logics, these quantifiers have been generalized to express that, for a non-negative integer n, at least n individuals or all but n individuals satisfy a formula. In modal logics, *graded modalities* generalize standard existential and universal modalities in that they express, e.g., that there exist at least n accessible worlds satisfying a certain formula. Graded modalities are useful expressive means in knowledge representation; they are present in a variety of other knowledge representation formalisms closely related to modal logic.
A natural question that arises is how the generalization of the existential and universal modalities affects the satisfiability problem for the logic and its computational complexity, especially when the numbers in the graded modalities are coded in binary. In this paper we study the *graded μ-calculus*, which extends graded modal logic with fixed-point operators, or, equivalently, extends classical μ-calculus with graded modalities. We prove that the satisfiability problem for graded μ-calculus is EXPTIME-complete – not harder than the satisfiability problem for μ-calculus, even when the numbers in the graded modalities are coded in binary.

1 Introduction

In classical logic, existential and universal quantifiers express that there exists *at least one* individual satisfying a formula, or that *all* individuals satisfy a formula. In many logics, these quantifiers have been generalized to express that, for a non-negative integer n, *at least* n individuals or *all but* n individuals satisfy a formula. For example, predicate logic has been extended with so-called *counting quantifiers* $\exists^{\geq n}$ and $\exists^{\leq n}$ [GOR97,PST00]. In modal logics, *graded modalities* [Fin72,vD95,Tob01] generalize standard existential and universal modalities in that they express, e.g., that there exist at least n accessible worlds satisfying a certain formula. In description logics, *number restrictions* have always played a central role; e.g., they are present in almost all knowledge-representation systems based on description logic [PSMB$^+$91,BFH$^+$94,Hor98,HM01].

* Supported in Part by NSF grants CCR-9988322, IIS-9908435, IIS-9978135, and EIA-0086264, and by BSF grant 9800096.

Indeed, in a typical such system, one can describe cars as those vehicles having at least four wheels, and bicycles as those vehicles having exactly two wheels.

A natural question that arises is how the generalization of the existential and universal quantifiers affects the satisfiability problem for the logic and its computational complexity. The complexity of a variety of description logics with different forms of number restrictions has been investigated; see, e.g. [DLNdN91,HB91,DL94b,BS99]. It turned out that, in many cases, one can extend a logic with these forms of counting quantifiers without increasing its computational complexity. On the other hand, in some cases the extension makes the logic much more complex. A prominent example is the guarded fragment of first order logic, which becomes undecidable when extended with a very weak form of counting quantifiers (global functionality conditions on binary relations) [Grä99].

When investigating the complexity of a logic with a form of counting quantifiers, one must decide how the numbers in these quantifiers contribute to the length of a formula, i.e., to the input of a decision procedure. Assuming that these numbers are coded in unary (i.e., $|\exists^{\geq n} x.\varphi(x)| = n + |\varphi(x)|$) might seem odd, but is an assumption often made, for example in description and predicate logics. It reflects the way in which many decision procedures for these logic work: they explicitly generate n individuals for $\exists^{\geq n}$. In contrast, the assumption that the numbers are coded in binary (i.e., $|\exists^{\geq n} x.\varphi(x)| = \lceil \log n \rceil + |\varphi(x)|$) corresponds more closely to our intuition on the length of a formula, but it is not clear whether and how decision procedures can avoid the exponential blow up that a translation from the binary to the unary coding involves.

It is an interesting question whether the complexity of a logic is sensitive to the coding of numbers in counting quantifiers. It seems as if many logics are insensitive to the coding, i.e., both complexities coincide, even though one has to “work harder” for binary coding. For many logics with counting quantifiers, the complexity of the satisfiability problem is known for unary coding only, and is unknown for binary coding. For example, $\mathbf{C}^2$ (two-variable first-order logic with counting) is known to be NEXPTIME-complete if numbers in counting quantifiers are coded in unary [PST00]. While this coincides with the complexity of first-order two-variable logic without counting [GKV97], the complexity of $\mathbf{C}^2$ with binary coding is, to the best of our knowledge, unknown so far. Similarly, all the above mentioned complexity results for description and modal logics, with the exception of [Tob00,Tob01], assume unary coding of numbers.

In [Tob00,Tob01], Tobies studies *graded modal logic*, the extension of modal logic with graded modalities. He proves that the satisfiability problem for this logic is PSPACE-complete — not harder than the satisfiability problem for classical modal logic [Lad77], even when the numbers in the graded modalities are coded in binary. The binary coding requires additional technical machinery. Indeed, since the number of individuals that satisfy a graded modality is exponential in the length of the modality, one cannot simply generate the individuals, but use some form of book keeping to keep track and count the individuals required by counting quantifiers.

The *μ-calculus* [Koz83] extends modal logic with least and greatest fixpoint operators. The extension makes μ-calculus a highly expressive logic, of great theoretical and practical interest (cf. [Eme97]). In this paper, we study the *graded μ-calculus*, i.e., μ-calculus with graded modalities. We show that the satisfiability problem for graded μ-calculus is EXPTIME-complete — not harder than the satisfiability problem for classical μ-calculus [FL79], even if the numbers are coded in binary. Our result substantiates the above hypothesis that most logics are insensitive to the coding of numbers, and is interesting for two additional reasons. Firstly, many relevant description, modal, and dynamic logics are fragments of the graded μ-calculus; see, e.g., [Sch94,DL94b,DL94a]. Hence we obtain corresponding EXPTIME upper bounds for these fragments for free. Secondly, other relevant description, modal, and dynamic logics such as $\mathcal{DLR}_\mu$, $\mathcal{SHIQ}$, graded modal logics, and $\mathcal{DN}$ [CDL99,HST00,Fin72,De 95] are close "relatives" of the graded μ-calculus. Thus we could use the techniques developed here to prove EXPTIME upper bounds and develop optimal reasoning algorithms for these relatives using similar techniques.

Our techniques are based on the automata-theoretic approach [VW86,SE89,Var97]: to develop a decision procedure for a logic with the tree-model property, one first develops an appropriate notion of tree automata and studies their emptiness problem. The satisfiability problem for the logic is then reduced to the automata emptiness problem. We show here that the appropriate notion of automata is that of *graded alternating tree automata*, which generalize standard alternating tree automata by having the ability to count the number of successors that satisfy a certain condition[4] We show that graded μ-calculus has a tree model property and that, given a formula φ, we can construct a graded alternating automaton $\mathcal{A}_\varphi$ that accepts exactly the (tree abstractions of) models of φ. The size of $\mathcal{A}_\varphi$ is linear in $|\varphi|$, even if numbers in graded modalities are coded in binary. We then present an EXPTIME decision procedure for the emptiness of graded alternating automaton by an appropriate translation into *graded non-deterministic automata* (with an exponential blow-up in the size of the automaton), and show that emptiness of graded non-deterministic automata can be decided in polynomial time. Like other automata-based decision procedures, the techniques developed here can be re-used: once a suitable class of automata for a certain class of logics is designed (together with the corresponding emptiness test), these automata can be easily re-used for similar logics. In particular, our technique can be easily extended to handle in EXPTIME μ-calculus with *fractional modalities*, where we can express, e.g., that at least half of the accessible worlds satisfy some predicate, as well as all modalities that involve polynomially-checkable conditions on the number of accessible words that satisfy a formula.

2 The Graded μ-Calculus

The *graded μ-calculus* is a propositional modal logic augmented with least and greatest fixpoint operators [Koz83]. Specifically, we consider a μ-calculus where formulas are constructed from atomic propositions with Boolean connectives, the *graded modalities* $\langle n, \alpha \rangle$ ("exist at least n α-successors") and $[n, \alpha]$ ("all but at most n α-successors"), as

[4] Some variants of alternating automata that support counting are studied in the literature (c.f., [Wal96]). Unlike these variants, where counting is done in the transition function of the automaton, our graded automata count by maintaining binary counters that should satisfy counting constraints. This is essential for the efficient treatment of constraints coded in binary.

well as least (μ) and greatest (ν) fixpoint operators. We assume that μ-calculus formulas are written in positive normal form (negation is applied only to atomic propositions). Formally, given a set AP of atomic propositions, a set Var of propositional variables, and a set Prog of (atomic) programs, the set of formulas of the graded μ-calculus is the smallest set such that the following holds.

– $\mathbf{true}$, $\mathbf{false}$, p and $\neg p$, for $p \in \mathsf{AP}$, are formulas,
– $x \in \mathsf{Var}$ is a formula, and
– if φ_1 and φ_2 are formulas, α is a program, n is a non-negative integer, and x is a propositional variable, then $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$, $\langle n, \alpha \rangle \, \varphi_1$, $[n, \alpha] \, \varphi_1$, $\mu y.\varphi_1(y)$, and $\nu y.\varphi_1(y)$ are formulas.

A propositional variable x occurs *free* in a formula if it is not in the scope of a fixpoint operator. Note that x may occur both bound and free in a formula. A *sentence* is formula that contains no free variables. We use λ to denote a fixpoint operator μ or ν. For a λ-formula $\lambda x.\varphi(x)$, we write $\varphi(\lambda x.\varphi(x))$ to denote the formula that is obtained by replacing each free occurrence of x in φ with $\lambda x.\varphi(x)$. We refer to formulas of the form $\langle n, \alpha \rangle \, \varphi_1$ and $[n, \alpha] \, \varphi_1$ as *atleast* and *allbut* formulas, respectively.

We say that a formula φ *counts up to* b if the maximal integer occurring in graded modalities in φ is $b - 1$. We refer to b as the *counting bound* of φ. We assume that the integers in graded modalities are given in binary. The *length* of φ, denoted $|\varphi|$, reflects this fact. Formally, $|\varphi|$ is defined by induction on the structure of φ in a standard way, with $| \langle n, \alpha \rangle \, \varphi_1| = \lceil \log n \rceil + 1 + |\varphi_1|$, and similarly for $| \, [n, \alpha] \, \varphi_1|$.

We define the semantics of graded μ-calculus with respect to *Kripke structures*. The semantics is similar to the one for standard μ-calculus (see [Koz83]), with the exception of the graded modalities: a state w of a Kripke structure K satisfies the atleast formula $\langle n, \alpha \rangle \varphi$ iff at least $n + 1$ successors of w in α satisfy φ. Dually, w satisfies the allbut formula $[n, \alpha]\varphi$ iff all but at most n successors of w in α satisfy φ. Note that $\neg \langle n, \alpha \rangle \, \varphi$ is equivalent to $[n, \alpha] \, \neg\varphi$. Indeed, $\neg \langle n, \alpha \rangle \, \varphi$ means that less than $n+1$ successors of w satisfy φ, that is, at most n successors do not satisfy $\neg\varphi$. The least and greatest fixpoint operators are interpreted as in μ-calculus; thus, for example, $\mu y.p \vee \langle 1, \alpha \rangle y$ is satisfied in a point w if either w satisfies p or w has two different α-successors each of which either satisfies p or has two different α-successors etc., or equivalently, w is a root of a binary tree embedded in the transition relation of α in which each path eventually reaches a point that satisfies p. Note that the interpretation of a sentence is independent of valuations. A sentence ψ is called *satisfiable* iff there is a Kripke structure K and a state u of K such that u satisfies ψ.

The modalities $\langle n, \alpha \rangle \, \varphi$ and $[n, \alpha] \, \varphi$ are natural generalizations of the standard existential and universal next modalities. In particular, $\langle \alpha \rangle \varphi$ and $[\alpha]\varphi$ of modal logic are equivalent to $\langle 0, \alpha \rangle \, \varphi$ and $[0, \alpha] \, \varphi$, respectively, and the *number restrictions* $(\geq \; n \, r \, \varphi)$ and $(\leq \; n \, r \, \varphi)$ for a role r in description logics [HB91] are equivalent to $\langle n-1, r \rangle \, \varphi$ and $[n, r] \, \neg\varphi$, respectively (note that $(\geq \; 0 \, r \, \varphi)$ is equivalent to $\mathbf{true}$).

For technical convenience, we restrict our attention to formulas and Kripke structures in which only one program occurs. By adding atomic propositions associated with programs, one can reduce formulas and structures with several programs to our setting. Note that we can also add new atomic propositions that would take care of the counting done in graded modalities. Formally, if φ counts up to b, we add propositions $c_1, \ldots, c_b$, conjoin φ with a requirement that exactly one c_i holds in each point, that successors that are labeled by the same c_i agree on their label with respect to all the subformulas of φ, and replace an atleast modality $\langle n, \psi \rangle$ by a $\bigvee_{\{j_1,\ldots,j_{n+1}\} \subseteq \{1,\ldots,b\}} \bigwedge_{1 \leq i \leq n+1} \Diamond(\psi \wedge c_{j_i})$, and dually for allbut modalities. The μ-calculus formula we get is satisfiable iff φ is satisfiable, yet the length of each disjunct that replaces a graded modality is exponential in b. Since the bounds in the graded modalities are written in binary, the length of the formula we get is doubly exponential in the length of φ.

A *tree* is a set $T \subseteq \mathbb{N}^*$ such that if $x \cdot c \in T$ where $x \in \mathbb{N}^*$ and $c \in \mathbb{N}$, then also $x \in T$. The elements of T are called *nodes*, and the empty word ε is the *root* of T. For every $x \in T$, the nodes $x \cdot c$ where $c \in \mathbb{N}$ are the *children* of x. The number of children of x is called the *degree* of x, and is denoted $deg(x)$. The *degree* of a tree is the maximum degree of a node in the tree. A node is a *leaf* if it has no children. A *path* π of a tree T is a set $\pi \subseteq T$ such that $\varepsilon \in \pi$ and for every $x \in \pi$, either x is a leaf or there exists a unique $c \in \mathbb{N}$ such that $x \cdot c \in \pi$. Given an alphabet Σ, a *Σ-labeled tree* is a pair $\langle T, V \rangle$ where T is a tree and $V : T \to \Sigma$ maps each node of T to a letter in Σ.

In the full version, we show that the graded μ-calculus has the tree model property. Thus, if a formula φ is satisfiable, it is also satisfiable in a tree. Moreover, the number of atleast formulas in φ and its counting bound induce a sufficient degree for the tree. Formally, we have the following.

Theorem 1. *Consider a sentence ψ such that ψ has l atleast subsentences, each counting to at most b. If ψ is satisfiable, then ψ is satisfied in a tree whose degree is at most $l \cdot (b+1)$.*

3 Graded Automata

Automata over infinite trees (tree automata) run over Σ-labeled trees that have no leaves [Tho90]. *Alternating automata* generalize nondeterministic tree automata and were first introduced in [MS87]. Intuitively, while a nondeterministic automaton that visits a node x of the input tree send one copy of itself to each of the successors of x, an alternating automata can several copies of itself to the same successor.

3.1 Graded Alternating Parity Tree Automata

For a given set Y, let $\mathcal{B}^+(Y)$ 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, as usual, $\wedge$ has precedence over $\vee$. For a set $X \subseteq Y$ and a formula $\theta \in \mathcal{B}^+(Y)$, 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 $b \geq 0$, let $\langle [b] \rangle = \{\langle 0 \rangle, \langle 1 \rangle, \ldots, \langle b \rangle\}$ and $[[b]] = \{[0], [1], \ldots, [b]\}$, and let $D_b = \{\varepsilon\} \cup \langle [b] \rangle \cup [[b]]$. A *graded alternating tree automaton* is an automaton in which the transition function δ maps a state q and a letter σ to a formula in $\mathcal{B}^+(D_b \times Q)$. Intuitively, an atom (ε, q) means that the automaton sends a copy of itself in state q to the current node, an atom $(\langle n \rangle, q)$ means that the automaton sends copies in states q to $n+1$ different children of the current node, and $([n], q)$ means that the automaton sends copies in state q to all but n children of the current node. When, for instance, the automaton is in state q, reads a node x and $\delta(q, V(x)) = (\langle 3\rangle, q_1) \wedge (\varepsilon, q_2) \vee ([2], q_3)$, it can either send four copies in state q_1 to four different children of x and send a copy in state q_2 to x, or send copies in state q_3 to $deg(x) - 2$ children of x. So, while nondeterministic tree automata send exactly one copy to each child, graded automata can send several copies to the same child, they have ε transitions, and extend *symmetric automata* [JW95,Wil99] by specifying the number of children that need to satisfy an existential requirement or are exempt from satisfying a universal one.

Formally, a graded automaton is a tuple $\mathcal{A} = \langle \Sigma, b, Q, \delta, q_0, \alpha\rangle$, where Σ, Q, q_0, and α are as in alternating automata, b is a *counting bound*, and $\delta : Q \times \Sigma \to \mathcal{B}^+(D_b \times Q)$ is a transition function. A *run* of $\mathcal{A}$ on an input Σ-labeled tree $\langle T, V\rangle$ is a tree $\langle T_r, r\rangle$ (to be formally defined shortly) in which each node corresponds to a node of T and is labeled by an element of $\mathbb{N}^* \times Q$. A node in T_r, labeled by (x, q), describes a copy of the automaton that reads the node x of T and visits the state q. Note that many nodes of T_r can correspond to the same node of T; in contrast, in a run of a nondeterministic automaton on $\langle T, V\rangle$ there is a one-to-one correspondence between the nodes of the run and the nodes of the tree. The labels of a node and its children have to satisfy the transition function. Formally, the run $\langle T_r, r\rangle$ is an $(\mathbb{N}^* \times Q)$-labeled $\mathbb{N}$-tree such that $\varepsilon \in T_r$ and $r(\varepsilon) = (\varepsilon, q_0)$, and 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 $(c, s) \in S$, the following holds:

– If $c = \varepsilon$, then there is $j \in \mathbb{N}$ such that $y \cdot j \in T_r$ and $r(y \cdot j) = (x, s)$.
– If $c = \langle n\rangle$, then there are distinct $i_1, \ldots, i_{n+1} \in \mathbb{N}$ such that for all $1 \le j \le n+1$, there is $j' \in \mathbb{N}$ such that $y \cdot j' \in T_r$ and $r(y \cdot j') = (x \cdot i_j, s)$.
– If $c = [n]$, then there are distinct $i_1, \ldots, i_{deg(x)-n} \in \mathbb{N}$ such that for all $1 \le j \le deg(x) - n$, there is $j' \in \mathbb{N}$ such that $y \cdot j' \in T_r$ and $r(y \cdot j') = (x \cdot i_j, s)$.

Note that if $\theta =$ **true**, then y need not have children. This is the reason why T_r may have leaves. Also, since there exists no set S as required for $\theta =$ **false**, we cannot have a run that takes a transition with $\theta =$ **false**.

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 $\alpha = \{F_1, F_2, \ldots, F_k\}$ is such that $F_1 \subseteq F_2 \subseteq \cdots \subseteq F_k = Q$. The number k of sets in α is called the *index* of the automaton. Given a run $\langle T_r, r\rangle$ and an infinite path $\pi \subseteq T_r$, let $inf(\pi) \subseteq Q$ be such that $q \in inf(\pi)$ if and only if there are infinitely many $y \in \pi$ for which $r(y) \in \mathbb{N}^* \times \{q\}$. That is, $inf(\pi)$ contains exactly all the states that appear infinitely often in π. A path π satisfies a parity acceptance condition $\alpha = \{F_1, F_2, \ldots, F_k\}$ iff the minimal index i for which $inf(\pi) \cap F_i \neq \emptyset$ is even. An automaton accepts a tree if and only if there exists a run that accepts it. We denote by $\mathcal{L}(\mathcal{A})$ the set of all Σ-labeled trees that $\mathcal{A}$ accepts.

Theorem 2. *Given a sentence ψ of the graded μ-calculus*[5] *that counts up to b, we can construct a graded alternating parity automaton $\mathcal{A}_\psi$ such that*

[5] A graded μ-calculus sentence is *guarded* if for all $y \in \mathsf{Var}$, all the occurrences of y that are in a scope of a fixpoint modality λ are also in a scope of a graded modality that is itself in the scope of λ. Thus, a μ-calculus sentence is guarded if for all $y \in \mathsf{Var}$, all the occurrences of y are in the scope of a graded modality. For example, the formula $\mu y.(p \vee \langle 0\rangle y)$ is guarded

1. *$\mathcal{A}_\psi$ accepts exactly all trees that satisfy ψ.*
2. *$\mathcal{A}_\psi$ has $|\psi|$ states, index $|\psi|$, and counting bound b.*

Proof. The construction generalizes the one for μ-calculus sentences and parity automata [KVW00]. Given ψ, we define $\mathcal{A}_\psi = \langle 2^{\mathsf{AP}}, b, \mathsf{cl}(\psi), \delta, \psi, \alpha\rangle$, where for all $\sigma \in 2^{\mathsf{AP}}$, we define:

$$\begin{aligned}
\delta(p,\sigma) &= \textbf{true} \text{ if } p \in \sigma, & \delta(p,\sigma) &= \textbf{false} \text{ if } p \notin \sigma,\\
\delta(\neg p,\sigma) &= \textbf{true} \text{ if } p \notin \sigma, & \delta(\neg p,\sigma) &= \textbf{false} \text{ if } p \in \sigma,\\
\delta(\varphi_1 \wedge \varphi_2,\sigma) &= (\varepsilon,\varphi_1) \wedge (\varepsilon,\varphi_2), & \delta(\varphi_1 \vee \varphi_2,\sigma) &= (\varepsilon,\varphi_1) \vee (\varepsilon,\varphi_2),\\
\delta(\langle n\rangle\varphi,\sigma) &= (\langle n\rangle,\varphi), & \delta([n]\varphi,\sigma) &= ([n],\varphi),\\
\delta(\mu y.f(y),\sigma) &= \delta(f(\mu y.f(y)),\sigma), & \delta(\nu y.f(y),\sigma) &= \delta(f(\nu y.f(y)),\sigma)).
\end{aligned}$$

The acceptance condition α is defined as in the automata for standard μ-calculus, according to the *alternation level* of the formulas in $\mathsf{cl}(\psi)$. For details, see [BC96,KVW00].

3.2 Graded Nondeterministic Parity Tree Automata

For an integer b, a *b-bound* is pair in $B_b = \{(>,0),(\leq,0),(>,1),(\leq,1),\ldots,(>,b),(\leq,b)\}$. For a set X, a subset $P \subseteq X$, an $m > 0$, and a tuple $t = \langle x_1,\ldots,x_m\rangle \in X^m$, the *weight* of P in t, denoted $weight(P,t)$, is the number of elements in t that are members of P. That is, $weight(P,t) = |\{i : x_i \in P\}|$. For example, $weight(\{1,2,3\},\langle 1,2,2,4,2\rangle) = 4$. We say that t satisfies a b-bound $(>,n)$ with respect to P if $weight(P,t) > n$, and t satisfies a b-bound $(\leq,n)$ with respect to P if $weight(P,t) \leq n$.

For a set Y, we use $\mathcal{B}(Y)$ to denote the set of all Boolean formulas over atoms in Y. Each formula $\theta \in \mathcal{B}(Y)$ 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 $C \subseteq \mathcal{B}(Y) \times B_b$. For example, if $Y = \{y_1,y_2,y_3\}$, then we can have $C = \{\langle y_1 \vee \neg y_2,(\leq,3)\rangle,\langle y_3,(\leq,2)\rangle,\langle y_1 \wedge y_3,(>,1)\rangle\}$. 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 $sat(\theta)$. Thus, when $\theta \in \mathcal{B}(Y)$ 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.

For a constraint C, the *width* of C is the number of $\theta \in \mathcal{B}(Y)$ for which there is a b-bound ξ such that $\langle\theta,\xi\rangle \in C$. Note that θ may be paired with several b-bounds. Still, it is easy to replace C by an equivalent constraint C' (that is, a tuple t satisfies C iff t satisfies C') in which θ is paired with at most one constraint of the form $(>,n)$ and at most one constraint of the form $(\leq,n)$. We assume that we work with such *minimized* constraints. For two b-counting constraints C_1 and C_2, we denote by $C_1 \oplus C_2$ the minimization of $C_1 \cup C_2$. That is, if $\langle\theta,(>,n_1)\rangle \in C_1$ and $\langle\theta,(>,n_2)\rangle \in C_2$, then $C_1 \oplus C_2$ contains only $\langle\theta,(>,\max\{n_1,n_2\})\rangle$, and dually for constraints of the form $(\leq,n)$[6].

and the formula $\langle 0\rangle\mu y.(p \vee y)$ is not guarded. Given a graded μ-calculus formula, we can construct, in linear time, an equivalent guarded formula (see [BB87,KVW00] for a proof for μ-calculus, which is easily extendible to graded μ-calculus). Accordingly, we assume that all formulas are guarded. This is essential for the correctness of the construction in the proof.

[6] To keep the $\oplus$ operator efficient, we do not care for redundancies and contradictions that originate from the relation between the formulas in the constraints. For example, a minimized C may contain both $\langle\theta_1,(>,n)\rangle$ and $\langle\theta_2,(>,n)\rangle$ for θ_1 that implies θ_2, and it may contain both $\langle\theta,(>,n)\rangle$ and $\langle\theta,(\leq,n)\rangle$

We say that a constraint C is *short* if all the formulas θ that appear in C are of size linear in $|Y|$ and the width of C is at most $|Y|$. We use $\mathcal{C}(Y, b)$ to denote the set of all short b-counting constraints for 2^Y. We assume that the integers in the constraints are coded in binary. Thus, the size of $C \in \mathcal{C}(Y, b)$ is $O(|Y|^2 \lceil \log b \rceil)$.

Lemma 1. *Given a constraint $C \in \mathcal{C}(Y, b)$ and a set $S \subseteq 2^Y$, deciding whether there is a tuple $t \in (2^Y)^*$ such that t satisfies C can be done in space $(1 + \lceil \log(b + 1) \rceil)|Y|$ or time $(2b + 2)^{|Y|}$.*

Proof. Since the width of C is at most $|Y|$, an algorithm that guesses t (element by element) and updates a counter for each θ that participate in C requires space for storing the guess for the current element in 2^Y, and for storing the values of the counters. The algorithm terminates with a positive decision when the values of the counters are such that all the b-bounds in C are satisfied. There are at most $|Y|$ counters, each may count up to at most $b + 1$. Thus, the space required is $|Y| + |Y|\lceil \log(b + 1) \rceil$. In addition, since the length of each formula θ that participate in C is linear in $|Y|$, its valuation with respect to each element of the tuple can be done in space $\log |Y|$ [Lyn77].

A *graded nondeterministic parity tree automaton* (GNPT, for short) is $\mathcal{A} = \langle \Sigma, b, Q, \delta, q_0, \alpha \rangle$, where Σ and b, q_0, and α are as in GAPT, and the other components are as follows.

- The state space Q is encoded by a finite set Y of variables; that is, $Q \subseteq 2^Y$.
- The function $\delta : Q \times \Sigma \to \mathcal{C}(Y, b)$ maps a state and a letter to a b-counting constraint for 2^Y.

Note that, like GAPT, a GNPT is symmetric, in the sense it cannot distinguish between the different children of a node.

A *run* of the graded nondeterministic automaton $\mathcal{A}$ on a Σ-labeled tree $\langle T, V \rangle$ is a Q-labeled tree $\langle T, r \rangle$ such that $r(\varepsilon) = q_0$ and for every $x \in T$, the tuple $\langle r(x \cdot 1), r(x \cdot 2), \ldots, r(x \cdot deg(x)) \rangle$ satisfies $\delta(r(x), V(x))$. The run $\langle T, r \rangle$ is accepting if all its paths satisfy the parity acceptance condition.

We consider two special cases of GNPT.

- 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 $\theta_s \in \mathcal{B}(Y)$ is such that $sat(\theta_s) = \{s\}$. Thus, a forall automaton is a notational invariant 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.

Lemma 2. *Given a forall GNPT $\mathcal{A}_1$ with n_1 states and index k, and a safety GNPT $\mathcal{A}_2$ with n_2 states and counting bound b, we can define a GNPT $\mathcal{A}$ such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2)$. Moreover, $\mathcal{A}$ has $n_1 n_2$ states, index k, and counting bound b.*

4 The Nonemptiness Problem for GAPT

In this section we solve the nonemptiness problem for GAPT and conclude that the satisfiability problem for graded μ-calculus can be solved in EXPTIME. We first translate GAPT to GNPT, and then solve the nonemptiness problem for GNPT. In the case of standard μ-calculus, the solution to the satisfiability problem follows the same plan: we translate the formula to an alternating automaton $\mathcal{A}$, and then check the nonemptiness of $\mathcal{A}$ by first translating it to an equivalent nondeterministic automaton [MS95]. In our case, the automaton $\mathcal{A}$ is graded, so its translation into a nondeterministic automaton and the nonemptiness problem for the latter are more involved.

4.1 From GAPT to GNPT

Consider a GAPT $\mathcal{A} = \langle \Sigma, b, Q, \delta, q_0, \alpha \rangle$. Let $D_b = (\{\varepsilon\} \cup \langle [b] \rangle \cup [[b]])$. Recall that the transition function $\delta : Q \times \Sigma \to \mathcal{B}^+(D_b \times Q)$ maps a state and a letter to a formula in $\mathcal{B}^+(D_b \times Q)$. A *restriction* of δ is a partial function $\eta : Q \to 2^{D_b \times Q}$. For a letter $\sigma \in \Sigma$, we say that a restriction η is *relevant* to σ if for all $q \in Q$ for which $\delta(q, \sigma)$ is satisfiable (i.e., $\delta(q, \sigma)$ is not **false**), the set $\eta(q)$ satisfies $\delta(q, \sigma)$. If $\delta(q, \sigma)$ is not satisfiable, then $\eta(q)$ is undefined. Intuitively, by choosing the atoms that are going to be satisfied, η removes the nondeterminism in δ. Let F be the set of restrictions of δ. A *running strategy* of $\mathcal{A}$ for a Σ-labeled tree $\langle T, V \rangle$ is an F-labeled tree $\langle T, f \rangle$. We say that $\langle T, f \rangle$ is relevant with respect to $\langle T, V \rangle$ if for all $x \in T$, the restriction $f(x)$ is relevant to $V(x)$.

Consider a restriction η relevant to σ. For $q \in Q$, we say that a finite sequence $s = s_0, s_1, \ldots, s_{l+1}$ is a *step of* η *with* q *and* σ if $s_0 = q$, for all $0 \leq i < l$, we have $(\varepsilon, s_{i+1}) \in \eta(s_i)$, and $(\lambda, s_{l+1}) \in \eta(s_l)$, for $\lambda \in \langle b \rangle \cup [b]$. Thus, s is a step of η with q and σ if by following the restriction η at a node x labeled σ, a run that visits q can continue by first taking l subsequent ε-transitions and visiting $s_0, \ldots, s_l$, and then moving to a child of x in state s_{l+1}. We refer to (λ, s_{l+1}) as the last atom taken in the step. Note that l may be 0. We define the *value* of s, denoted $val(s)$, as the minimal i such that there is $0 < j \leq l + 1$ with $s_j \in F_i$. Note that when s contains only two states, its value is induced by s_1.

We say that a finite sequence $s = s_0, s_1, \ldots, s_l$ is an *ε-lasso of* η *with* q *and* σ if $s_0 = q$, for all $0 \leq i \leq l - 1$, we have $(\varepsilon, s_{i+1}) \in \eta(s_i)$, and there is $0 \leq c \leq l$ such that $(\varepsilon, s_c) \in \eta(s_l)$ Thus, s is an ε-lasso of η with q and σ if by following the restriction η at a node x labeled σ, there is $0 \leq c \leq l$ such that a run that visits q can eventually loop forever in $s_c, \ldots, s_l$ by taking subsequent ε-transitions. The value of s with a loop starting at c, denoted $val(s, c)$, is the minimal i such that there is $c \leq j \leq l$ with $s_j \in F_i$. We say that s is rejecting if there is $0 \leq c \leq l$ such that $val(s)$ is odd.

A *local promise* for the automaton $\mathcal{A}$ is a function $\rho : Q \to 2^Q$. We extend ρ to sets of states, thus for $P \subseteq Q$, we have $\rho(P) = \bigcup_{q \in P} \rho(q)$. Let G be the set of all local promises. A *promise* of $\mathcal{A}$ for a Σ-labeled tree $\langle T, V \rangle$ is a G-labeled tree $\langle T, g \rangle$. Intuitively, in a run that proceeds according to $\langle T, g \rangle$, if a node $y \cdot j$ has $s \in g(y \cdot j)(q)$ and the run visits its parent y in state q and proceeds by choosing an atom $\langle n \rangle s$ or $[n]s$, for some $0 \leq n \leq b$, then $y \cdot j$ is among the children of y that inherit s.

Consider a Σ-labeled tree $\langle T, V\rangle$, a running strategy $\langle T, f\rangle$ relevant to $\langle T, V\rangle$, and a promise $\langle T, g\rangle$. A $(T \times Q)$-labeled tree $\langle T_r, r\rangle$ is *consistent* with f and g if $\langle T_r, r\rangle$ suggests a possible run of $\mathcal{A}$ 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 restriction $f(x)(q)$ is defined, the run proceeds according to $f(x)(q)$, and it delivers requirements to each child $x \cdot j$ according to $g(x \cdot j)(q)$. Note that since the counting constraints in $f(x)(q)$ may not be satisfied, $\langle T_r, r\rangle$ may not be a legal run. Formally, $\langle T_r, r\rangle$ is consistent with f and g iff the following hold.

1. $\varepsilon \in T_r$ and $r(\varepsilon) = (\varepsilon, q_0)$.
2. Consider a node $y \in T_r$ with $r(y) = (x, q)$. Then, $f(x)(q)$ is defined, and for all $(c, s) \in f(x)(q)$, the following hold:
   – If $c = \varepsilon$, then there is $j \in \mathbb{N}$ such that $y \cdot j \in T_r$ and $r(y \cdot j) = (x, s)$.
   – If $c = \langle n\rangle$ or $c = [n]$, then for each $j \in \mathbb{N}$ with $s \in g(x \cdot j)(q)$, there is $j' \in \mathbb{N}$ such that $y \cdot j' \in T_r$ and $r(y \cdot j') = (x \cdot j, s)$.

For a node $x \in T$ and a state $q \in Q$, we say that x is *obliged to* q by f, g, and V if x is visited by q in some labeled tree $\langle T_r, r\rangle$ consistent with f and g.

Let $\Sigma' \subseteq \Sigma \times F \times G$ be such that for all $\langle \sigma, \eta, \rho\rangle \in \Sigma'$, we have that η is relevant to σ. For an infinite sequence $\langle \sigma_0, \eta_0, \rho_0\rangle, \langle \sigma_1, \eta_1, \rho_1\rangle, \ldots$ of triples in Σ' and a sequence (either finite or infinite) $q_0, q_1, \ldots$ of states, we say that $q_0, q_1, \ldots$ is a *trace* induced by $\langle \sigma_0, \eta_0, \rho_0\rangle, \langle \sigma_1, \eta_1, \rho_1\rangle, \ldots$ if q_0 is the initial state of $\mathcal{A}$ and there is a function $pos : \mathbb{N} \to \mathbb{N}$ such that $pos(0) = 0$ and for every $i \geq 0$, one of the following holds.

1. $\eta_{pos(i)}(q_i)$ is empty, in which case q_i is the last state in the trace,
2. there is $(\varepsilon, q_{i+1}) \in \eta_{pos(i)}(q_i)$ and $pos(i+1) = pos(i)$, or
3. $\eta_{pos(i)}(q_i)$ contains $(\langle n\rangle, q_{i+1})$ or $([n], q_{i+1})$, $q_{i+1} \in \rho_{pos(i)+1}(q_i)$, and $pos(i + 1) = pos(i) + 1$.

Intuitively, $q_0, q_1, \ldots$ is a trace induced by $\langle \sigma_0, \eta_0, \rho_0\rangle, \langle \sigma_1, \eta_1, \rho_1\rangle, \ldots$, if for every path $\pi \subseteq T$ and for every run $\langle T_r, r\rangle$ on a Σ-labeled tree in which π is labeled by $\sigma_0, \sigma_1, \ldots$, if $\langle T_r, r\rangle$ is consistent with a running strategy in which π is labeled $\eta_0, \eta_1, \ldots$ and a promise in which π is labeled $\rho_0, \rho_1, \ldots$, then $\langle T_r, r\rangle$ contains a path that visits the states $q_0, q_1, \ldots$.

Recall that $\Sigma' \subseteq \Sigma \times F \times G$. We refer to a Σ'-labeled tree as $\langle T, (V, f, g)\rangle$, where V, f, and g are the projections of the tree on Σ, F, and G, respectively. We say that a running strategy $\langle T, f\rangle$ and a promise $\langle T, g\rangle$ are *good* for $\langle T, V\rangle$ if all the infinite traces induced by paths in $\langle T, (V, f, g)\rangle$ satisfy the acceptance condition α.

Consider a Σ-labeled tree $\langle T, V\rangle$, a running strategy $\langle T, f\rangle$, and a promise $\langle T, g\rangle$. We say that g *fulfills* f for V if the states promised to be visited by g satisfy the obligations induced by f as it runs on V. Formally, g fulfills f for V if for every node $x \in T$, and state q such that x is obliged to q by f, g, and V, the following hold:

1. For every atom $\langle n\rangle s \in f(x)(q)$, at least $n+1$ children $x \cdot j$ of x have $s \in g(x \cdot j)(q)$.
2. For every atom $[n]s \in f(x)(q)$, at least $deg(x) - n$ children $x \cdot j$ of x have $s \in g(x \cdot j)(q)$.

Theorem 3. *A GAPT $\mathcal{A}$ accepts $\langle T, V\rangle$ iff there exist a running strategy $\langle T, f\rangle$ and a promise $\langle T, g\rangle$ such that f is relevant for V, f and g are good for $\langle T, V\rangle$, and g fulfills f for V.*

Intuitively, if f and g as above exist, the $(T \times Q)$-labeled trees that are consistent with f and g suggest legal accepting runs of $\mathcal{A}$ on $\langle T, V \rangle$.

Annotating input trees with restrictions and local promises enables us to transform GAPT to GNPT, with an exponential blow up:

Theorem 4. *Consider a GAPT $\mathcal{A}$ such that $\mathcal{A}$ runs on Σ-labeled trees. There is a GNPT $\mathcal{A}'$ such that $\mathcal{A}'$ runs on Σ'-labeled trees and the following hold:*

1. *$\mathcal{A}'$ accepts a tree iff $\mathcal{A}$ accepts its projection on Σ.*
2. *If $\mathcal{A}$ has n states, index k, and counting bound b, then $\mathcal{A}'$ has $2^{n(2+k \log nk)}$ states, index nk, and b-counting constraints.*

Proof. Let $\mathcal{A} = \langle \Sigma, b, Q, \delta, q_0, \alpha \rangle$ with $\alpha = \{F_1, \ldots, F_k\}$. The automaton $\mathcal{A}'$ is the intersection of two automata $\mathcal{A}'_1$ and $\mathcal{A}'_2$. The automaton $\mathcal{A}'_1$ is a forall GNPT and it accepts a tree $\langle T, (V, f, g) \rangle$ iff f and g are good for V. The automaton $\mathcal{A}'_2$ is a safety GNPT, and it accepts a tree $\langle T, (V, f, g) \rangle$ iff g fulfills f for V. Note that, since Σ' contains only triplets $\langle \sigma, \eta, \rho \rangle$ for which η is relevant to σ, it must be that f is relevant to V. Thus, by Theorem 3, it follows that $\mathcal{A}'$ accepts $\langle T, (V, f, g) \rangle$ iff $\mathcal{A}$ accepts $\langle T, V \rangle$.

In order to define $\mathcal{A}'_1$, we first define a nondeterministic co-parity word automaton $\mathcal{U}$ over Σ' such that $\mathcal{U}$ accepts a word if some trace it induces is infinite and violates the acceptance condition α. We define $\mathcal{U} = \langle \Sigma', S, M, s_0, F' \rangle$, where

– $S = (Q \times Q \times \{1, \ldots, k\}) \cup \{q_{acc}\}$. Intuitively, a state $\langle q, q_{prev}, v \rangle$ indicates that the current state of the trace is q, that it was reached by following a step whose last transition is from the state q_{prev}, and the value of the step is v (note that values are calculated with respect to α). Thus, q corresponds to states q_{i+1} in traces for which $pos(i + 1) = pos(i) + 1$. The number v is used for the acceptance condition. In addition, q_{prev} is used for checking the obligation of the current position, given a local promise in the input word.
– For every $\langle q, q_{prev}, v \rangle \in S$ and $\langle \sigma, \eta, \rho \rangle \in \Sigma'$, we distinguish between two cases. If $q \notin \rho(q_{prev})$, then the current position is not obliged to q and $M(\langle q, q_{prev}, v \rangle, \langle \sigma, \eta, \rho \rangle) = \emptyset$. Otherwise, we again distinguish between two cases: if there is a rejecting ε-lasso of η with q and σ, then $M(\langle q, q_{prev}, v \rangle, \langle \sigma, \eta, \rho \rangle) = \{q_{acc}\}$. Otherwise, $\langle q', q'_{prev}, v' \rangle \in M(\langle q, q_{prev}, v \rangle, \langle \sigma, \eta, \rho \rangle)$ iff there is a step $q, \ldots, q'_{prev}, q'$ of η with q and σ such that the value of the step is v'.
In addition, $M(q_{acc}, \langle \sigma, \eta, \rho \rangle) = \{q_{acc}\}$ for all $\langle \sigma, \eta, \rho \rangle \in \Sigma'$. Intuitively, $\mathcal{U}$ checks whether a possible step of η with q and σ can participate in a rejecting trace. If the current position is not obliged to the current state, no step of η can participate in a trace, so $\mathcal{U}$ gets stuck. Otherwise, if there is a rejecting ε-lasso of η with q and σ, a rejecting trace is found and $\mathcal{U}$ moves to an accepting sink. Otherwise, $\mathcal{U}$ guesses other possible steps of η with q and σ, and moves to a state which remembers the last two states visited in the step (possibly $q'_{prev} = q$), and the value of the step.
– $s'_0 = \langle q_0, q_0, l \rangle$, where l is such that $q_0 \in F_l$. Note that the choice of the second element is arbitrary, as the local promise at the root of the input tree is irrelevant.
– The co-parity condition is $F' = \{F'_1, F'_2, \ldots, F'_k\}$, where for $l \geq 2$, we have $F'_l = Q \times Q \times \{l\}$, and $F'_1 = (Q \times Q \times \{1\}) \cup \{q_{acc}\}$. That is, acceptance is determined with respect to the values of the steps taken along the trace. Also, since F' is a co-parity condition, the accepting sink q_{acc} is in F'_1.

In order to get $\mathcal{A}'_1$, we co-determinize $\mathcal{U}$ (note that $\mathcal{U}$ does not have ε-transitions) and expand it to a tree automaton on Σ'. That is, we first construct a deterministic parity word automaton $\tilde{\mathcal{U}}$ that complements $\mathcal{U}$, and then replace a transition $\tilde{M}(s,\tau) = s'$ in $\tilde{\mathcal{U}}$ by a transition $\tilde{M}_t(s,\tau) = \{\langle \neg\theta_{s'}, (\leq, 0)\rangle\}$ in $\mathcal{A}'_1$, where the states of $\tilde{\mathcal{U}}$ are encoded by some set Y_1 of variables and for every state s', the formula $\theta_{s'} \in \mathcal{B}(Y_1)$ holds only in the subset of Y_1 that encodes s'. By [Saf89,Tho97], the automaton $\tilde{\mathcal{U}}$ has $(nk)^{nk}$ states and index nk, thus so does $\mathcal{A}'_1$. Hence $|Y_1| = nk \log nk$.

It is left to define the safety GNPT $\mathcal{A}'_2$. Let $Q_{prev} = \{q_{prev} : q \in Q\}$ be a copy of Q in which each state is tagged with $prev$. The state space of $\mathcal{A}'_2$ is $Q' = 2^{Q \cup Q_{prev}}$. Intuitively, each state q' of $\mathcal{A}'_2$ corresponds to a pair $\langle P, P_{prev}\rangle \in Q \times Q$, with $P = q' \cap Q$ and P_{prev} is obtained from $q' \cap Q_{prev}$ by removing the *prev* tags. The element P of q' is a set of "commitments" that the current node should satisfy. The element P_{prev} is used for remembering the state of $\mathcal{A}$ that is visited in the parent node. When $\mathcal{A}'_2$ is in state $\langle P, P_{prev}\rangle$ and reads the letter $\langle \sigma, \eta, \rho\rangle$, it checks that all the commitments in P are covered by the local promise $\rho(P_{prev})$ in the input, and it delivers, for each $q \in P$, the requirements on the children as specified in $\eta(q)$.

Consider a state $\langle P, P_{prev}\rangle \in Q'$ and a letter $\langle \sigma, \eta, \rho\rangle \in \Sigma'$. For every $q \in P$, let $C^q_{\sigma,\eta}$ be the b-counting restriction in $\mathcal{C}(Q, b)$ imposed by $\eta(q)$. (If $\eta(q)$ is undefined, we do not care about $C^q_{\sigma,\eta}$, since, as we see shortly, in that case $\mathcal{A}'_2$ simply gets stuck.) Thus, $C^q_{\sigma,\eta} = \{\langle s, (>, n)\rangle : \langle n\rangle s \in \eta(q)\} \cup \{\langle \neg s, (\leq, n)\rangle : [n]s \in \eta(q)\}$. Intuitively, $C^q_{\sigma,\eta}$ restricts the tuple of the states that visit the children of the current node, which is visited by $\langle P, P_{prev}\rangle$, so that $\eta(q)$ is satisfied by the first elements of the states. In addition, the second element of the states in the tuple should be the encoding of P tagged with $prev$. This is done by the counting constraint $\{\langle \neg\theta^{prev}_P, (\leq, 0)\rangle\}$, where $\theta^{prev}_P \in \mathcal{B}(Q_{prev})$ is such that the only set that satisfies θ^{prev}_P is the encoding of P tagged with $prev$. Finally, for every $P \in 2^Q$, let $C^P_{\sigma,\eta} = (\oplus_{q \in P} C^q_{\sigma,\eta}) \cup \{\langle \neg\theta^{prev}_P, (\leq, 0)\rangle\}$. Then, $\mathcal{A}'_2 = \langle \Sigma', Q', \delta', \{q_0, q_0\}\rangle$, where for every $\langle P, P_{prev}\rangle \in Q'$ and $\langle \sigma, \eta, \rho\rangle \in \Sigma'$, we have that $\delta'(\langle P, P_{prev}\rangle, \langle \sigma, \eta, \rho\rangle)$ is empty if $\rho(P_{prev}) \not\subseteq P$ or there is $q \in P$ for which $\eta(q)$ is undefined, and is $C^P_{\sigma,\eta}$ otherwise. Note that Q' is defined with respect to the $2n$ variables $Q \cup Q_{prev}$. Also, all the formulas θ that are paired to constraints in $C^P_{\sigma,\eta}$ are either s or $\neg s$, for $s \in Q$, or $\neg\theta^{prev}_P$. Hence, the counting constraints in $\mathcal{A}'_2$ are in $\mathcal{C}(Q \cup Q_{prev}, b)$.

Now, by Lemma 2, we can define the the intersection $\mathcal{A}'$ of $\mathcal{A}'_1$ and $\mathcal{A}'_2$ as a GNPT with $2^{n(2+k \log nk)}$ states, index nk, and b-counting constraints.

4.2 The nonemptiness problem for GNPT

In a nondeterministic parity tree automaton $\mathcal{U} = \langle \Sigma, Q, M, q_0, \alpha\rangle$, the transition function $M : Q \times \Sigma \to 2^{Q^*}$ maps a state and a letter to a set of possible tuples for the children states. Thus, a run of nondeterministic tree automaton on a tree $\langle T, V\rangle$ is a Q-labeled tree $\langle T, r\rangle$ in which $r(\varepsilon) = q_0$ and for all $x \in T$, the tuple $\langle r(x \cdot 1), r(x \cdot 2), \ldots, r(x \cdot deg(x))\rangle \in M(r(x), V(x))$. The nonemptiness test for parity tree automata then uses the local test $is\_mother : 2^Q \times Q \to \{\textbf{true}, \textbf{false}\}$ that given a set $S \subseteq Q$ and a state q, returns **true** iff there is a tuple $t \in S^*$ and $\sigma \in \Sigma$ such that $t \in M(q, \sigma)$. It is easy to see how the is_mother test is used in a bottom-up nonemptiness algorithm for automata on finite trees, where in order to find the set S of states from which the automaton accepts some tree, one starts with the set S_0 of accepting states then define S_{i+1} as the set of states q such that either q is in S_i or $is\_mother(t,q) =$ **true**. In parity automata, the algorithm is more complicated, as one has to also keep track of the acceptance condition, but the same local test is used. Several nonemptiness algorithms for nondeterministic parity tree automata are known. In particular, the algorithms in [EJS93,KV98] use $O(n^k)$ calls to *is_mother*, where n is the size of Q and k is the index of the automaton.

Recall that in GNPT, a run $\langle T, r\rangle$ should satisfy $r(\varepsilon) = q_0$ and for all $x \in T$, the tuple $\langle r(x \cdot 1), r(x \cdot 2), \ldots, r(x \cdot deg(x))\rangle$ satisfies $\delta(r(x), V(x))$, which is a b-counting constraint. Thus, the nonemptiness test is similar, only that the local test $is\_mother : 2^Q \times Q \to \{\textbf{true}, \textbf{false}\}$ now returns **true** for a set $S \subseteq Q$ and a state q, iff there is $t \in S^*$ and $\sigma \in \Sigma$ such that t satisfies $\delta(q,\sigma)$. As with nondeterministic automata, the nonemptiness algorithm can do $O(n^k)$ calls to is_mother. Unlike the case for nondeterministic automata, however, here there is no simple transition function to consult when we perform the local test. In addition, we should take into an account the fact that the GNPT whose emptiness we check have larger alphabets than the GAPT we have started with.

Consider a GAPT $\mathcal{A} = \langle \Sigma, b, Q, \delta, q_0, \alpha\rangle$ with n states, index k, and counting bound b. Let us analyse carefully the complexity of the local is_mother test in the GNPT $\mathcal{A}'$ we constructed from $\mathcal{A}$ in Theorem 4. First, $\mathcal{A}'$ has counting constraints in $\mathcal{C}(Y', b)$, for Y' of size $n(2+k\log nk)$. Hence, by Lemma 1, given S, the check whether there is a tuple $t \in S^*$ such that t satisfies $\delta(q,\sigma')$, for a particular $\sigma' \in \Sigma'$, can be done in time $O(2b+2)^{n(2+k\log nk)})$. Now, $\Sigma' \subseteq \Sigma \times F \times G$, where F is the set of restrictions for δ and G is the set of all local promises. Let $|\Sigma| = l$. Recall that a restriction relevant to a letter $\sigma \in \Sigma$ maps a state $q \in Q$ to a subset of $D_b \times Q$ that satisfies $\delta(q,\sigma)$. We can restrict our attention to restrictions in which each state is paired with at most one element of $\langle\langle b\rangle\rangle$, one element of $[\langle b\rangle]$, and ε. Thus, $|F|$ is bounded by $(2b+4)^{n^2}$ and $|G|$ is bounded by 2^{n^2}. It follows that $|\Sigma'| \leq l(2b+4)^{n^2}2^{n^2}$, thus is_mother can be checked in time $l(b+2)^{O(n(n+2+k\log nk)}$. Since, as in [EJS93,KV98], the nonemptiness problem can be solved by $O(n^k)$ applications of is_mother, we have the following.

Theorem 5. *The nonemptiness problem for $\mathcal{A}'$ can be solved in time $n^k l(b+2)^{O(n(n+2+k\log nk))}$.*

For a graded μ-calculus formula ψ, we get, by Theorem 2, a GAPT $\mathcal{A}$ with n and k bounded by $|\psi|$, and the same counting bound b as ψ. While b and l may be exponential in $|\psi|$, only n and k appear in the exponents in the expression in Theorem 5. This implies the upper bound in the theorem below. The lower bound is due to the fact that the μ-calculus is known to be EXPTIME-hard [FL79].

Corollary 1. *The satisfiability problem for graded μ-calculus is EXPTIME-complete even if the numbers in the graded modalities are coded in binary.*

Note that the space and time bounds in Lemma 1 stay valid for counting constraints that involve richer bounds than $(>, n)$ and $(\leq, n)$. For example, we can handle bounds of the form $(>, \frac{1}{2})$ or $(\leq, \frac{1}{2})$, bounding the fraction of elements in the tuple that satisfy a predicate (of course, this is applicable only to structures where all points have only finitely many successors). In general, Lemma 1 can handle arbitrary polynomial predicates $\alpha \subseteq \mathbb{N}^2$, where a tuple $t \in (2^Y)^m$ satisfies such a constraint $\langle\theta,\alpha\rangle$ if $\alpha(weight(sat(\theta),t),m)$ holds. By defining the corresponding types of alternating automata, we can thus handle μ-calculus formulas with richer types of modalities.

References

[BB87] B. Banieqbal and H. Barringer. Temporal logic with fixed points. In *Temporal Logic in Specification*, volume 398 of *LNCS*, pages 62–74. Springer-Verlag, 1987.

[BC96] G. Bhat and R. Cleaveland. Efficient local model-checking for fragments of the modal μ-calculus. In *Proc. of TACAS-96, LNCS* 1055. Springer-Verlag, 1996.

[BFH$^+$94] F. Baader, E. Franconi, B. Hollunder, B. Nebel, and H.J. Profitlich. An empirical analysis of optimization techniques for terminological representation systems, or: Making KRIS get a move on. *Applied Artificial Intelligence*, 4:109–132, 1994.

[BS99] F. Baader and U. Sattler. Expressive number restrictions in description logics. *Journal of Logic and Computation*, 9(3):319–350, 1999.

[CDL99] D. Calvanese, G. De Giacomo, and M. Lenzerini. Reasoning in expressive description logics with fixpoints based on automata on infinite trees. In *IJCAI'99*, 1999.

[De 95] G. De Giacomo. *Decidability of Class-Based Knowledge Representation Formalisms*. PhD thesis, Università degli Studi di Roma "La Sapienza", 1995.

[DL94a] G. De Giacomo and M. Lenzerini. Boosting the correspondence between description logics and propositional dynamic logics. In *Proc. of AAAI-94*, 1994.

[DL94b] G. De Giacomo and M. Lenzerini. Concept language with number restrictions and fixpoints, and its relationship with mu-calculus. In *Proc. of ECAI-94*, 1994.

[DLNdN91] F. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. In *Proc. of KR-91*, 1991.

[EJS93] E.A. Emerson, C. Jutla, and A.P. Sistla. On model-checking for fragments of μ-calculus. In *Proc. 4th CAV*, LNCS 697, pages 385–396. Springer-Verlag, 1993.

[Eme97] E.A. Emerson. Model checking and the μ-calculus. In *Descriptive Complexity and Finite Models*, pages 185–214. American Mathematical Society, 1997.

[Fin72] K. Fine. In so many possible worlds. *Notre Dame Journal of Formal Logics*, 13:516–520, 1972.

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

[GKV97] E. Grädel, Ph. G. Kolaitis, and M. Y. Vardi. The decision problem for 2-variable first-order logic. *Bulletin of Symbolic Logic*, 3:53–69, 1997.

[GOR97] E. Grädel, M. Otto, and E. Rosen. Two-variable logic with counting is decidable. In *Proc. of LICS-97*, 1997.

[Grä99] E. Grädel. On the restraining power of guards. *Journal of Symbolic Logic*, 64, 1999.

[HB91] B. Hollunder and F. Baader. Qualifying number restrictions in concept languages. In *Proc. of KR-91*, pages 335–346, 1991.

[HM01] V. Haarslev and R. Möller. RACER system description. In *Proc. of IJCAR-01*, volume 2083 of *LNAI*. Springer-Verlag, 2001.

[Hor98] I. Horrocks. Using an Expressive Description Logic: FaCT or Fiction? In *Proc. of KR-98*, 1998.

[HST00] I. Horrocks, U. Sattler, and S. Tobies. Reasoning with individuals for the description logic shiq. In *Proc. of CADE-17*, LNCS 1831, Germany, 2000. Springer-Verlag.

[JW95] D. Janin and I. Walukiewicz. Automata for the modal μ-calculus and related results. In *Proc. of MFCS-95*, LNCS, pages 552–562. Springer-Verlag, 1995.

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

[KV98] O. Kupferman and M.Y. Vardi. Weak alternating automata and tree automata emptiness. In *Proc. STOC-98*, pages 224–233, 1998.

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

[Lad77] R. E. Ladner. The computational complexity of provability in systems of modal propositional logic. *SIAM Journal of Control and Optimization*, 6(3):467–480, 1977.

[Lyn77] N. Lynch. Log space recognition and translation of parenthesis languages. *Journal of the ACM*, 24:583–590, 1977.

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

[MS95] D.E. Muller and P.E. Schupp. Simulating alternating tree automata by nondeterministic automata: New results and new proofs of theorems of Rabin, McNaughton and Safra. *Theoretical Computer Science*, 141:69–107, 1995.

[PSMB$^+$91] P. Patel-Schneider, D. McGuinness, R. Brachman, L. Resnick, and A. Borgida. The CLASSIC knowledge representation system: Guiding principles and implementation rationale. *SIGART Bulletin*, 2(3):108–113, 1991.

[PST00] L. Pacholski, W. Szwast, and L. Tendera. Complexity results for first-order two-variable logic with counting. *SIAM Journal of Computing*, 29(4):1083–1117, 2000.

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

[Sch94] K. Schild. Terminological cycles and the propositional μ-calculus. In *Proc. of KR-94*, pages 509–520. Morgan Kaufmann, 1994.

[SE89] R.S. Streett and E.A. Emerson. An automata theoretic decision procedure for the propositional μ-calculus. *Information and Computation*, 81(3):249–264, 1989.

[Tho90] W. Thomas. Automata on infinite objects. In J. Van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 165–191. North Holland, 1990.

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

[Tob00] S. Tobies. The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. *Journal of Artificial Intelligence Research*, 12:199–217, 2000.

[Tob01] S. Tobies. PSPACE reasoning for graded modal logics. *Journal of Logic and Computation*, 11(1):85–106, 2001.

[Var97] M.Y. Vardi. What makes modal logic so robustly decidable? In *Descriptive Complexity and Finite Models*, pages 149–183. American Mathematical Society, 1997.

[vD95] W. van der Hoek and M. De Rijke. Counting objects. *Journal of Logic and Computation*, 5(3):325–345, 1995.

[VW86] M.Y. Vardi and P. Wolper. Automata-theoretic techniques for modal logics of programs. *Journal of Computer and System Science*, 32(2):182–221, 1986.

[Wal96] I. Walukiewicz. Monadic second order logic on tree-like structures. In *Proc. of STACS-96*, LNCS, pages 401–413. Springer-Verlag, 1996.

[Wil99] T. Wilke. CTL$^+$ is exponentially more succinct than CTL. In *Proc. of FSTTCS-99*, volume 1738 of *LNCS*, pages 110–121. Springer-Verlag, 1999.