July 22, 2008

THE COMPLEXITY OF ENRICHED μ-CALCULI

PIERO A. BONATTI [a,⋆], CARSTEN LUTZ [b], ANIELLO MURANO [c,⋆], AND MOSHE Y. VARDI [d,⋆⋆]

[a] Università di Napoli "Federico II", Dipartimento di Scienze Fisiche, 80126 Napoli, Italy
e-mail address: bonatti@na.infn.it

[b] TU Dresden, Institute for Theoretical Computer Science, 01062 Dresden, Germany
e-mail address: clu@tcs.inf.tu-dresden.de

[c] Università di Napoli "Federico II", Dipartimento di Scienze Fisiche, 80126 Napoli, Italy
e-mail address: murano@na.infn.it

[d] Microsoft Research and Rice University, Dept. of Computer Science, TX 77251-1892, USA
e-mail address: vardi@cs.rice.edu

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 these problems. The automata models we introduce are *two-way graded alternating parity automata* over infinite trees (2GAPTs) and *fully enriched automata* (FEAs) 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] and 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

⋆ 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.

LOGICAL METHODS
IN COMPUTER SCIENCE
DOI:10.2168/LMCS-???

	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)

Figure 1: Enriched μ-calculi and previous results.

into the μ-calculus, we can identify DLs that are of very high expressive power, but computationally well-behaved [CGL99, 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. The most important such 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 possibly 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 [CGL99, 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 expressivity.

The relevant fragments of the fully enriched μ-calculus are shown in Figure 1 together with the complexity of their satisfiability problem. The results shown in gray are already known from the literature: the EXPTIME lower bound for the original μ-calculus stems from [FL79]; it has been shown in [SV01] that satisfiability in the full hybrid μ-calculus is in EXPTIME; 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 [CGL99]; 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 (in contrast, the techniques used in [CGL99] involve an exponential blow-up when numbers are coded in binary).

Our results are based on the automata-theoretic approach and extends the techniques in [KSV02, SV01, Var98]. They involve introducing two novel automata models. To show that the full graded μ-calculus is in EXPTIME, we introduce *two-way graded parity tree automata (2GAPTs)*. These automata generalize in a natural way two existing, but incomparable automata models: two-way alternating parity tree automata (2APTs) [Var98] and (one-way) graded alternating parity tree automata (GAPTs) [KSV02]. The phrase "two-way" indicates that 2GAPTs (like 2APTs) can move up and down in the tree. The phrase "graded" indicates that 2GAPTs (like GAPTs) have the ability

to count the number of successors of a tree node that it moves to. Namely, such an automaton can move to at least n or all but n successors of the current node, without specifying which successors exactly these are. We show that the emptines problem for 2GAPT is in EXPTIME by a reduction to the emptiness of graded nondeterministic parity tree automata (GNPTs) as introduced in [KSV02]. This is the technically most involved part of this paper. To show the desired upper bound for the full graded μ-calculus, it remains to reduce satisfiability in this calculus to emptiness of 2GAPTs. This reduction is based on the tree model property of the full graded μ-calculus, and technically rather standard.

To show that the hybrid graded μ-calculus is in EXPTIME, we introduce *fully enriched automata (FEAs)* which run on infinite forests and, like 2GAPTs, use a parity acceptance condition. FEAs extend 2GAPTs by additionally allowing the automaton to send a copy of itself to some or all roots of the forest. This feature of "jumping to the roots" is in rough correspondence with the nominals included in the full hybrid μ-calculus. We show that the emptiness problem for FEAs is in EXPTIME using an easy reduction to the emptiness problem for 2GAPTs. To show that the hybrid graded μ-calculus is in EXPTIME, it thus remains to reduce satisfiability in this calculus to emptiness of FEAs. Since the correspondence between nominals in the μ-calculus and the jumping to roots of FEAs is only a rough one, this reduction is more delicate than the corresponding one for the full graded μ-calculus. The reduction is based on a forest model property enjoyed by the hybrid graded μ-calculus and requires us to work with the *two-way* automata FEAs although the hybrid graded μ-calculus does not offer inverse programs.

We remark that, intuitively, FEAs 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). Still, decidability of the emptiness problem for FEAs does not contradict the undecidability of the fully enriched μ-calculus since the latter does not enjoy a forest model property [BP04], and hence satisfiability cannot be decided using forest-based FEAs.

The rest of the paper is structured as follows. The subsequent section introduces the syntax and semantics of the fully enriched μ-calculus. The tree model property for the full graded μ-calculus and a forest model property for the hybrid graded μ-calculus are then established in Section 3. In Section 4, we introduce FEAs and 2GAPTs and show how the emptiness problem for the former can be polynomially reduced to that of the latter. In this section, we also state our upper bounds for the emptiness problem of these automata models. Then, Section 5 is concerned with reducing the satisfiability problem of enriched μ-calculi to the emptiness problems of 2GAPTs and FEAs. The purpose of Section 6 is to reduce the emptiness problem for 2GAPTs to that of GNPTs. Finally, we conclude in Section 7.

2. ENRICHED μ-CALCULI

We introduce the syntax and semantics of the fully enriched μ-calculus. Let Prop be a finite set of *atomic propositions*, Var a finite set of *propositional variables*, Nom a finite set of *nominals*, and Prog a finite set of *atomic programs*. We use Prog^- to denote the set of *inverse programs* $\{a^- \mid a \in \mathsf{Prog}\}$. The elements of $\mathsf{Prog} \cup \mathsf{Prog}^-$ are called *programs*. We assume $a^{--} = a$. The set of *formulas of the fully enriched μ-calculus* is the smallest set such that

- true and false are formulas;
- p and $\neg p$, for $p \in \mathsf{Prop}$, are formulas;
- o and $\neg o$, for $o \in \mathsf{Nom}$, are formulas;

- $x \in \mathsf{Var}$ is a formula;
- $\varphi_1 \vee \varphi_2$ and $\varphi_1 \wedge \varphi_2$ are formulas if φ_1 and φ_2 are formulas;
- $\langle n, \alpha\rangle\varphi$, and $[n, \alpha]\varphi$ are formulas if n is a non-negative integer, α is a program, and φ is a formula;
- $\mu y.\varphi(y)$ and $\nu y.\varphi(y)$ are formulas if y is a propositional variable and $\varphi(y)$ is a formula containing y as a free variable.

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 λy, and *bounded* otherwise. Note that y may occur both bounded and free in a formula. 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 often refer to the *graded modalities* $\langle n, \alpha\rangle\varphi$ and $[n, \alpha]\varphi$ 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 Figure 1. Hence, we say that a formula of the fully enriched μ-calculus is also a formula of the *hybrid graded μ-calculus*, *full hybrid μ-calculus*, and *full graded μ-calculus* if it does not have inverse programs, graded modalities, and nominals, respectively.

The semantics of the fully enriched μ-calculus is defined in terms of a *Kripke structure*, i.e., a tuple $K = \langle W, R, L\rangle$ where

- W is a non-empty (possibly infinite) set of *states*;
- $R : \mathsf{Prog} \to 2^{W\times W}$ assigns to each atomic program a binary relation over W;
- $L : \mathsf{Prop} \cup \mathsf{Nom} \to 2^W$ 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 atomic program a, we set $R(a^-) = \{(v, u) : (u, v) \in R(a)\}$. For a program α, if $(w, w') \in R(\alpha)$, we say that w' is an *α-successor* of w. With $\mathsf{succ}_R(w, \alpha)$ we denote the set of α-successors 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 $[n, \alpha]\neg\varphi$. Indeed,$\neg\langle n, \alpha\rangle\varphi$ holds in a state w if φ holds in less than $n+1$ α-successors of w, thus, at most n α-successors of w do not satisfy $\neg\varphi$, that is, $[n, \alpha]\neg\varphi$ holds in w. 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. The least and greatest fixpoint operators are interpreted as in the standard μ-calculus. Readers not familiar with fixpoints might want to look at [Koz83, SE89, BS06] for instructive examples and explanations of the semantics of the μ-calculus.

To formalize the semantics, we introduce valuations. Given a Kripke structure $K = \langle W, R, L\rangle$ and a set $\{y_1, \ldots, y_n\}$ of propositional 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 states that satisfy φ under valuation $\mathcal{V}$. The mapping φ^K is defined inductively as follows:

- $\mathsf{true}^K(\mathcal{V}) = W$ and $\mathsf{false}^K(\mathcal{V}) = \emptyset$;
- for $p \in \mathsf{Prop} \cup \mathsf{Nom}$, we have $p^K(\mathcal{V}) = L(p)$ and $(\neg p)^K(\mathcal{V}) = W \setminus L(p)$;

- for $y \in \mathsf{Var}$, we have $y^K(\mathcal{V}) = \mathcal{V}(y)$;
- $(\varphi_1 \wedge \varphi_2)^K(\mathcal{V}) = \varphi_1^K(\mathcal{V}) \cap \varphi_2^K(\mathcal{V})$
- $(\varphi_1 \vee \varphi_2)^K(\mathcal{V}) = \varphi_1^K(\mathcal{V}) \cup \varphi_2^K(\mathcal{V})$;
- $(\langle n, \alpha\rangle\varphi)^K(\mathcal{V}) = \{w : |\{w' \in W : (w, w') \in R(\alpha) \text{ and } w' \in \varphi^K(\mathcal{V})\}| > n\}$;
- $([n, \alpha]\varphi)^K(\mathcal{V}) = \{w : |\{w' \in W : (w, w') \in R(\alpha) \text{ and } w' \notin \varphi^K(\mathcal{V})\}| \leq n\}$;
- $(\mu y.\varphi(y))^K(\mathcal{V}) = \bigcap\{W' \subseteq W : \varphi^K(\mathcal{V}[y \leftarrow W']) \subseteq W'\}$;
- $(\nu y.\varphi(y))^K(\mathcal{V}) = \bigcup\{W' \subseteq W : W' \subseteq \varphi^K(\mathcal{V}[y \leftarrow W'])\}$.

Note that, in the clauses for graded modalities, α denotes a program, i.e., α can be either an atomic program or an inverse program. Also, note that no valuation is required for a sentence.

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 $w \in \varphi^K(\emptyset)$. 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.

3. Tree and Forest Model Properties

We show that the full graded μ-calculus has the tree model property, and that the hybrid graded μ-calculus has a forest model property. Regarding the latter, we speak of "a" (rather than "the") forest model property because it is an abstraction of the models that is forest-shaped, instead of the models themselves.

For a (potentially infinite) set X, we use X^+ (X^*) to denote the set of all non-empty (possibly empty) words over X. As usual, for $x, y \in X^*$, we use $x \cdot y$ to denote the concatenation of x and y. Also, we use ε to denote the empty word and by convention we take $x \cdot \varepsilon = x$, for each $x \in X^*$. Let $\mathbb{N}$ be a set of non-negative integers. A *forest* is a set $F \subseteq \mathbb{N}^+$ that is prefix-closed, that is, if $x \cdot c \in F$ with $x \in \mathbb{N}^+$ and $c \in \mathbb{N}$, then also $x \in F$. The elements of F are called *nodes*. For every $x \in F$, the nodes $x \cdot c \in F$ with $c \in \mathbb{N}$ are the *successors* of x, and x is their *predecessor*. We use $\mathsf{succ}(x)$ to denote the set of all successors of x in F. A *leaf* is a node without successors, and a *root* is a node without predecessors. A forest F is a *tree* if $F \subseteq \{c \cdot x \mid x \in \mathbb{N}^*\}$ for some $c \in \mathbb{N}$ (the *root* of F). The root of a tree F is denoted with $\mathsf{root}(F)$. If for some c, $T = F \cap \{c \cdot x \mid x \in \mathbb{N}^*\}$, then we say that T is *the tree of F rooted in c.*

We call a Kripke structure $K = \langle W, R, L\rangle$ a *forest structure* if

(*i*) W is a forest,

(*ii*) $\bigcup_{\alpha \in \mathsf{Prog} \cup \mathsf{Prog}^-} R(\alpha) = \{(w, v) \in W \times W \mid w \text{ is a predecessor or a successor of } v\}$.

Moreover, K is *directed* if $(w, v) \in \bigcup_{a \in \mathsf{Prog}} R(a)$ implies that v is a successor of w. If W is a tree, then we call K a *tree structure*.

We call $K = \langle W, R, L\rangle$ a *directed quasi-forest structure* if $\langle W, R', L\rangle$ is a directed forest structure, where $R'(a) = R(a) \setminus (W \times \mathbb{N})$ for all $a \in \mathsf{Prog}$, i.e., K becomes a directed forest structure after deleting all the edges entering a root of W. Let φ be a formula and $o_1, \ldots, o_k$ the nominals occurring in φ. 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 there are roots $c_0, \ldots, c_k \in W \cap \mathbb{N}$ with $K, c_0 \models \varphi$ and $L(o_i) = \{c_i\}$, for $1 \leq i \leq k$. Observe that the roots $c_0, \ldots, c_k$ do not have to be distinct.

Using a standard unwinding technique such as in [Var98, KSV02], it is possible to show that the full graded μ-calculus enjoys the tree model property, i.e., if a formula φ is satisfiable, it is also satisfiable in a tree model. We omit details and concentrate on the similar, but more difficult proof of the fact that the hybrid graded μ-calculus has a forest model property.

Theorem 3.1. *If a sentence φ of the full graded μ-calculus is satisfiable, then φ has a tree model.*

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 n if the atomic propositions p_i are forced to be mutually exclusive (which is easy using additional formulas). However, we can follow [SV01, KSV02] to show that the hybrid graded μ-calculus has a forest model property. More precisely, we prove that the hybrid graded μ-calculus enjoys the *quasi-forest model property*, i.e., if a formula φ is satisfiable, it is also satisfiable in a directed quasi-forest structure.

The proof is a variation of the original construction for the μ-calculus given by Streett and Emerson in [SE89]. It is an amalgamation of the constructions for the hybrid μ-calculus in [SV01] and for the hybrid graded μ-calculus in [KSV02]. We start with introducing the notion of a well-founded adorned pre-model, which augments a model with additional information that is relevant for the evaluation of fixpoint formulas. Then, we show that any satisfiable sentence φ of the hybrid graded μ-calculus has a well-founded adorned pre-model, and that any such pre-model can be unwound into a tree-shaped one, which can be converted into a directed quasi-forest model of φ.

To determine the truth value of a Boolean formula, it suffices to consider its subformulas. For μ-calculus formulas, one has to consider a larger collection of formulas, the so called Fischer-Ladner closure [FL79]. The *closure* $\mathsf{cl}(\varphi)$ of a sentence φ of the hybrid graded μ-calculus is the smallest set of sentences satisfying the following:

- $\varphi \in \mathsf{cl}(\varphi)$;
- if $\psi_1 \wedge \psi_2 \in \mathsf{cl}(\varphi)$ or $\psi_1 \vee \psi_2 \in \mathsf{cl}(\varphi)$, then $\{\psi_1, \psi_2\} \subseteq \mathsf{cl}(\varphi)$;
- if $\langle n,a\rangle\psi \in \mathsf{cl}(\varphi)$ or $[n,a]\psi \in \mathsf{cl}(\varphi)$, then $\psi \in \mathsf{cl}(\varphi)$;
- if $\lambda y.\psi(y) \in \mathsf{cl}(\varphi)$, then $\psi(\lambda y.\psi(y)) \in \mathsf{cl}(\varphi)$.

An *atom* is a subset $A \subseteq \mathsf{cl}(\varphi)$ satisfying the following properties:

- if $p \in \mathsf{Prop} \cup \mathsf{Nom}$ occurs in φ, then $p \in A$ iff $\neg p \notin A$;
- if $\psi_1 \wedge \psi_2 \in \mathsf{cl}(\varphi)$, then $\psi_1 \wedge \psi_2 \in A$ iff $\{\psi_1, \psi_2\} \subseteq A$;
- if $\psi_1 \vee \psi_2 \in \mathsf{cl}(\varphi)$, then $\psi_1 \vee \psi_2 \in A$ iff $\{\psi_1, \psi_2\} \cap A \neq \emptyset$;
- if $\lambda y.\psi(y) \in \mathsf{cl}(\varphi)$, then $\lambda y.\psi(y) \in A$ iff $\psi(\lambda y.\psi(y)) \in A$.

The set of atoms of φ is denoted $\mathsf{at}(\varphi)$. A *pre-model* $\langle K, \pi\rangle$ for a sentence φ of the hybrid graded μ-calculus consists of a Kripke structure $K = \langle W, R, L\rangle$ and a mapping $\pi : W \to \mathsf{at}(\varphi)$ that satisfies the following properties:

- there is $w_0 \in W$ with $\varphi \in \pi(w_0)$;
- for $p \in \mathsf{Prop} \cup \mathsf{Nom}$, if $p \in \pi(w)$, then $w \in L(p)$, and if $\neg p \in \pi(w)$, then $w \notin L(p)$;
- if $\langle n,a\rangle\psi \in \pi(w)$, then there is a set $V \subseteq \mathsf{succ}_R(w,a)$, such that $|V| > n$ and $\psi \in \pi(v)$ for all $v \in V$;
- if $[n,a]\psi \in \pi(w)$, then there is a set $V \subseteq \mathsf{succ}_R(w,a)$, such that $|V| \leq n$ and $\psi \in \pi(v)$ for all $v \in \mathsf{succ}_R(w,a) \setminus V$.

If there is a pre-model $\langle K, \pi\rangle$ of φ such that for every state w and all $\psi \in \pi(w)$, it holds that $K, w \models \psi$, then K is clearly a model of φ. However, the definition of pre-models does not guarantee that $\psi \in \pi(w)$ is satisfied at w if ψ is a least fixpoint formula. In a nutshell, the standard approach for dealing with this problem is to enforce that it is possible to trace the evaluation of a least fixpoint formula through K such that the original formula is not regenerated infinitely often. When tracing such evaluations, a complication is introduced by disjunctions and at least restrictions, which require us to make a choice on how to continue the trace. To address this issue, we adapt the notion of a choice function of Streett and Emerson [SE89] to the hybrid graded μ-calculus.

A *choice function* for a pre-model $\langle K, \pi\rangle$ for φ is a partial function ch from $W \times \mathsf{cl}(\varphi)$ to $\mathsf{cl}(\varphi) \cup 2^W$, such that for all $w \in W$, the following conditions hold:

- if $\psi_1 \vee \psi_2 \in \pi(w)$, then $\mathsf{ch}(w, \psi_1 \vee \psi_2) \in \{\psi_1, \psi_2\} \cap \pi(w)$;
- if $\langle n, a\rangle\psi \in \pi(w)$, then $\mathsf{ch}(w, \langle n, a\rangle\psi) = V \subseteq \mathsf{succ}_R(w, a)$, such that $|V| > n$ and $\psi \in \pi(v)$ for all $v \in V$;
- if $[n, a]\psi \in \pi(w)$, then $\mathsf{ch}(w, [n, a]\psi) = V \subseteq \mathsf{succ}_R(w, a)$, such that $|V| \leq n$ and $\psi \in \pi(v)$ for all $v \in \mathsf{succ}_R(w, a) \setminus V$.

An *adorned pre-model* $\langle K, \pi, \mathsf{ch}\rangle$ of φ consists of a pre-model $\langle K, \pi\rangle$ of φ and a choice function ch. We now define the notion of a derivation between occurrences of sentences in adorned pre-models, which formalizes the tracing mentioned above. For an adorned pre-model $\langle K, \pi, \mathsf{ch}\rangle$ of φ, the *derivation relation* $\rightsquigarrow \subseteq (W \times \mathsf{cl}(\varphi)) \times (W \times \mathsf{cl}(\varphi))$ is the smallest relation such that, for all $w \in W$, we have:

- if $\psi_1 \vee \psi_2 \in \pi(w)$, then $(w, \psi_1 \vee \psi_2) \rightsquigarrow (w, \mathsf{ch}(\psi_1 \vee \psi_2))$;
- if $\psi_1 \wedge \psi_2 \in \pi(w)$, then $(w, \psi_1 \wedge \psi_2) \rightsquigarrow (w, \psi_1)$ and $(w, \psi_1 \wedge \psi_2) \rightsquigarrow (w, \psi_2)$;
- if $\langle n, a\rangle\psi \in \pi(w)$, then $(w, \langle n, a\rangle\psi) \rightsquigarrow (v, \psi)$ for each $v \in \mathsf{ch}(w, \langle n, a\rangle\psi)$;
- if $[n, a]\psi \in \pi(w)$, then $(w, [n, a]\psi) \rightsquigarrow (v, \psi)$ for each $v \in \mathsf{succ}_R(w, a) \setminus \mathsf{ch}(w, [n, a]\psi)$;
- if $\lambda y.\psi(y) \in \pi(w)$, then $(w, \lambda y.\psi(y)) \rightsquigarrow (w, \psi(\lambda y.\psi(y)))$.

A least fixpoint sentence $\mu y.\psi(y)$ is *regenerated* from state w to state v in an adorned pre-model $\langle K, \pi, \mathsf{ch}\rangle$ of φ if there is a sequence $(w_1, \rho_1), \ldots, (w_k, \rho_k) \in (W \times \mathsf{cl}(\varphi))^*$, $k > 1$, such that $\rho_1 = \rho_k = \mu y.\psi(y)$, $w = w_1$, $v = w_k$, the formula $\mu y.\psi(y)$ is a sub-sentence of each ρ_i in the sequence, and for all $1 \leq i < k$, we have $(w_i, \rho_i) \rightsquigarrow (w_{i+1}, \rho_{i+1})$. We say that $\langle K, \pi, \mathsf{ch}\rangle$ is *well-founded* if there is no least fixpoint sentence $\mu y.\psi(y) \in \mathsf{cl}(\varphi)$ and infinite sequence $w_1, w_2, \ldots$ such that, for each $i \geq 1$, $\mu y.\psi(y)$ is regenerated from w_i to w_{i+1}. The proof of the following lemma is based on signatures, i.e., sequence of ordinals that guides the evaluation of least fixpoints. It is a minor variation of the one given for the original μ-calculus in [SE89]. Details are omitted.

Lemma 3.2. *Let φ be a sentence of the hybrid graded μ-calculus. Then:*

(1) *if φ is satisfiable, it has a well-founded adorned pre-model;*
(2) *if $\langle K, \pi, \mathsf{ch}\rangle$ is a well-founded adorned pre-model of φ, then K is a model of φ.*

We now establish the forest model property of the hybrid graded μ-calculus.

Theorem 3.3. *If a sentence φ of the hybrid graded μ-calculus is satisfiable, then φ has a directed quasi-forest model.*

Proof. Let φ be satisfiable. By item (1) of Lemma 3.2, there is a well-founded adorned pre-model $\langle K, \pi, \mathsf{ch}\rangle$ for φ. We unwind K into a directed quasi-forest structure $K' = \langle W', L', R'\rangle$, and define a corresponding mapping $\pi' : W' \to \mathsf{at}(\varphi)$ and choice function ch' such that $\langle K', \pi', \mathsf{ch}'\rangle$ is again a well-founded adorned pre-model of φ. Then, item (2) of Lemma 3.2 yields that K' is actually a model of φ.

Let $K = \langle W, L, R\rangle$, and let $w_0 \in W$ such that $\varphi \in \pi(w_0)$. The set of states W' of K' is a subset of $\mathbb{N}^+$ as required by the definition of (quasi) forest structures, and we define K' in a stepwise manner by proceeding inductively on the length of elements of W'. Simultaneously, we define π', ch', and a mapping $\tau : W' \to W$ that keeps track of correspondences between states in K' and K.

The base of the induction is as follows. Let $I = \{w_1, \ldots, w_k\} \subseteq W$ be a minimal subset such that $w_0 \in I$ and if o is a nominal in φ and $L(o) = \{w\}$, then $w \in I$. Define K' by setting:

- $W' := \{1, \ldots, k\}$;

- $R'(a) := \{(i,j) \mid (w_i, w_j) \in R(a), 1 \leq i \leq j \leq k\}$ for all $a \in \mathsf{Prog}$;
- $L'(p) := \{i \mid w_i \in L(p), 1 \leq i \leq k\}$ for all $p \in \mathsf{Prop} \cup \mathsf{Nom}$.

Define τ by setting $\tau(i) = w_i$ for $1 \leq i \leq k$. Then, $\pi'(w)$ is defined as $\pi(\tau(w))$ for all $w \in W'$, and ch' is defined by setting $\mathsf{ch}'(w, \psi_1 \vee \psi_2) = \mathsf{ch}(\tau(w), \psi_1 \vee \psi_2)$ for all $\psi_1 \vee \psi_2 \in \pi'(w)$. Choices for atleast and allbut formulas are defined in the induction step.

In the induction step, we iterate over all $w \in W'$ of maximal length, and for each such w extend K', π', ch', and τ as follows. Let $(\langle a_1, n_1\rangle\psi_1, v_1), \ldots, (\langle a_m, n_m\rangle\psi_m, v_m)$ be all pairs from $\mathsf{cl}(\varphi) \times W$ of this form such that for each $(\langle a_i, n_i\rangle\psi_i, v_i)$, we have $\langle a_i, n_i\rangle\psi_i \in \pi(w)$ and $v_i \in \mathsf{ch}(\tau(w), \langle a_i, n_i\rangle\psi_i)$. For $1 \leq i \leq m$, define

$$\sigma(v_i) = \begin{cases} j & \text{if } v_i = \tau(j), 1 \leq j \leq k \\ w \cdot i & \text{otherwise.} \end{cases}$$

To extend K', set

- $W' := W' \cup \{\sigma(v_1), \ldots, \sigma(v_m)\}$;
- $R'(a) := R'(a) \cup \{(w, \sigma(v_i)) \mid a_i = a, 1 \leq i \leq m\}$ for all $a \in \mathsf{Prog}$;
- $L'(p) := L'(p) \cup \{w \cdot i \in W \mid v_i \in L(p), 1 \leq i \leq m\}$ for all $p \in \mathsf{Prop} \cup \mathsf{Nom}$.

Extend τ and π' by setting $\tau(w \cdot i) = v_i$ and $\pi'(w \cdot i) = \pi(v_i)$ for all $w \cdot i \in W'$. Finally, extend ch' by setting

- $\mathsf{ch}'(w \cdot i, \psi_1 \vee \psi_2) := \mathsf{ch}(v_i, \psi_1 \vee \psi_2)$ for all $w \cdot i \in W'$ and $\psi_1 \vee \psi_2 \in \pi'(w \cdot i)$;
- $\mathsf{ch}'(w, \langle n, a\rangle\psi) := \{\sigma(v) \mid v \in \mathsf{ch}(\tau(w), \langle n, a\rangle\psi)\}$ for all $\langle n, a\rangle\psi \in \pi'(w)$;
- $\mathsf{ch}'(w, [n, a]\psi) := \{\sigma(v) \mid v \in \mathsf{ch}(\tau(w), [n, a]\psi) \cap \{v_1, \ldots, v_m\}\}$ for all $[n, a]\psi \in \pi'(w)$.

It is easily seen that K' is a directed quasi-forest structure. Since $\langle K, \pi, \mathsf{ch}\rangle$ is an adorned pre-model of φ, it is readily checked that $\langle K', \pi', \mathsf{ch}'\rangle$ is an adorned pre-model of φ as well. If a sentence $\mu y.\psi(y)$ is regenerated from x to y in (K', π', ch'), then $\mu y.\psi(y)$ is regenerated from $\tau(x)$ to $\tau(y)$ in (K, π, ch). It follows that well-foundedness of $\langle K, \pi, \mathsf{ch}\rangle$ implies well-foundedness of $\langle K', \pi', \mathsf{ch}'\rangle$. □

Note that the construction from this proof fails for the fully enriched μ-calculus because the unwinding of K duplicates states, and thus also duplicates incoming edges to nominals. Together with inverse programs and graded modalities, this may result in $\langle K', \pi'\rangle$ not being a pre-model of φ.

4. 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 the predecessor of x, too. In graded automata [KSV02], the automaton can send copies of itself to a number n of successors, without specifying which successors these exactly are. Our most general automata model is that of fully enriched automata, as introduced in the next subsection. These automata 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.

4.1. **Fully enriched automata.** We start with some preliminaries. Let $F \subseteq \mathbb{N}^+$ be a forest, x a node in F, and $c \in \mathbb{N}$. As a convention, we take $(x \cdot c) \cdot -1 = x$ and $c \cdot -1$ as undefined. 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 $c \in F$ 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. We call $\langle F, V\rangle$ a Σ-*labeled tree* if F is a tree.

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

$$\begin{array}{rcl} \langle\!\langle b \rangle\!\rangle & = & \{\langle 0\rangle, \langle 1\rangle, \ldots, \langle b\rangle\} \\ [[b]] & = & \{[0], [1], \ldots, [b]\} \\ D_b & = & \langle\!\langle b \rangle\!\rangle \cup [[b]] \cup \{-1, \varepsilon, \langle \mathsf{root}\rangle, [\mathsf{root}]\} \end{array}$$

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 in state q to the predecessor of the current node, and $(\langle \mathsf{root}\rangle, q)$ (resp. $([\mathsf{root}], q)$) means that the automaton sends a copy in state q to some root (resp. all roots). When, for instance, the automaton is in state q, reads a node x, and

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

it sends a copy in state q_1 to the predecessor and either sends a copy in state q_2 to some root 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 a finite 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 an acceptance condition. A *run* of A on an input Σ-labeled forest $\langle F, V\rangle$ is an $F \times Q$-labeled tree $\langle T_r, r\rangle$. 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 at a root and satisfy the transition relation. Thus, a run $\langle T_r, r\rangle$ has to satisfy the following conditions:

(*i*) $r(\mathsf{root}(T_r)) = (c, q_0)$ 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 is a set $M \subseteq \mathsf{succ}(x)$ of cardinality $n+1$ such that for all $z \in M$, there is $j \in \mathbb{N}$ such that $y \cdot j \in T_r$and $r(y \cdot j) = (z, s)$;
- If $d = [n]$, then there is a set $M \subseteq \mathsf{succ}(x)$ of cardinality n such that for all $z \in \mathsf{succ}(x) \setminus M$, there is $j \in \mathbb{N}$ such that $y \cdot j \in T_r$ and $r(y \cdot j) = (z, s)$;
- If $d = \langle \mathsf{root}\rangle$, 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 $d = [\mathsf{root}]$, then for each root $c \in F$ there exists $j \in \mathbb{N}$ such that $y \cdot j \in T_r$ and $r(y \cdot j) = (c, s)$.

Note that if $\theta = \mathsf{true}$, then y does not need to have successors. Moreover, since no set S satisfies $\theta = \mathsf{false}$, there cannot be any run that takes a transition with $\theta = \mathsf{false}$.

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* [Mos84, EJ91, Tho97], where $\mathcal{F} = \{\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_k\}$ is such that $\mathcal{F}_1 \subseteq \mathcal{F}_2 \subseteq \ldots \subseteq \mathcal{F}_k = Q$. 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 $\mathsf{Inf}(\pi) \subseteq Q$ be the set of states q such that $r(y) \in F \times \{q\}$ for infinitely many $y \in \pi$. A path π *satisfies* a parity acceptance condition $\mathcal{F} = \{\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_k\}$ if the minimal i with $\mathsf{Inf}(\pi) \cap \mathcal{F}_i \neq \emptyset$ is even. An automaton *accepts* a forest iff there exists an accepting run of the automaton on the forest. We denote by $\mathcal{L}(A)$ the set of all Σ-labeled forests that A accepts. The *emptiness problem* for FEAs is to decide, given a FEA A, whether $\mathcal{L}(A) = \emptyset$.

4.2. **Two-way graded alternating parity tree automata.** 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 transitions $\langle\mathsf{root}\rangle$ and $[\mathsf{root}]$. The emptiness problem for 2GAPTs is thus a special case of the emptiness problem for FEAs. In the following, we give a reduction of the emptiness problem for FEAs to the emptiness problem for 2GAPTs. This allows us to derive an upper bound for the former problem from the upper bound for the latter that is established in Section 6.

We show how to translate a FEA A into a 2GAPT A' such that $\mathcal{L}(A')$ consists of the forests accepted by A, encoded as trees. The encoding that we use is straightforward: the *tree encoding* of a Σ-labeled forest $\langle F, V\rangle$ is the $\Sigma \uplus \{\mathsf{root}\}$-labeled tree $\langle T, V'\rangle$ obtained from $\langle F, V\rangle$ by adding a fresh root labeled with $\{\mathsf{root}\}$ whose children are the roots of F.

Lemma 4.1. *Let A be a FEA running on Σ-labeled forests with n states, index k and counting bound b. There exists a 2GAPT A' that*

(1) *accepts exactly the tree encodings of forests accepted by A and*
(2) *has $\mathcal{O}(n)$ states, index k, and counting bound b.*

Proof. Suppose $A = \langle \Sigma, b, Q, \delta, q_0, \mathcal{F}\rangle$. Define A' as $\langle \Sigma \uplus \{\mathsf{root}\}, b, Q', \delta', q_0', \mathcal{F}'\rangle$, where Q' and δ' are defined as follows:

$$
\begin{array}{rcl}
Q' & = & Q \uplus \{q_0', q_r\} \uplus \{\mathsf{some}_q, \mathsf{all}_q \mid q \in Q\} \\
\delta'(q_0', \mathsf{root}) & = & (\langle 0\rangle, q_0) \wedge ([0], q_r) \\
\delta'(q_0', \sigma) & = & \mathsf{false} \text{ for all } \sigma \neq \{\mathsf{root}\} \\
\delta'(q_r, \mathsf{root}) & = & \mathsf{false} \\
\delta'(q_r, \sigma) & = & ([0], q_r) \text{ for all } \sigma \neq \{\mathsf{root}\} \\
\delta'(\mathsf{some}_q, \sigma) & = & \begin{cases} (-1, \mathsf{some}_q) & \text{if } \sigma \neq \mathsf{root} \\ (\langle 0\rangle, q) & \text{otherwise} \end{cases} \\
\delta'(\mathsf{all}_q, \sigma) & = & \begin{cases} (-1, \mathsf{all}_q) & \text{if } \sigma \neq \mathsf{root} \\ ([0], q) & \text{otherwise} \end{cases} \\
\delta'(q, \sigma) & = & \mathsf{tran}(\delta(q, \sigma)) \text{ for all } q \in Q \text{ and } \sigma \in \Sigma
\end{array}
$$

Here, $\mathsf{tran}(\beta)$ replaces all atoms $(\langle\mathsf{root}\rangle, q)$ in β with $(\varepsilon, \mathsf{some}_q)$, and all atoms $([\mathsf{root}], q)$ in β with $(\varepsilon, \mathsf{all}_q)$. The acceptance condition $\mathcal{F}'$ is identical to $\mathcal{F} = \{\mathcal{F}_1, \ldots, \mathcal{F}_k\}$, except that all $\mathcal{F}_i$ are extended with q_r and $\mathcal{F}_k$ is extended with q_0 and all states some_q and all_q. It is not hard to see that A' accepts $\langle T, V\rangle$ iff A accepts the forest encoded by $\langle T, V\rangle$. □

In Section 6, we shall prove the following result.

Theorem 4.2. *The emptiness problem for a 2GAPT $A = \langle \Sigma, b, Q, \delta, q_0, \mathcal{F} \rangle$ with n states and index k can be solved in time $(b+2)^{\mathcal{O}(n^3 \cdot k^2 \cdot \log k \cdot \log b^2)}$.*

By Lemma 4.1, we obtain the following corollary.

Corollary 4.3. *The emptiness problem for a FEA $A = \langle \Sigma, b, Q, \delta, q_0, \mathcal{F} \rangle$ with n states and index k can be solved in time $(b+2)^{\mathcal{O}(n^3 \cdot k^2 \cdot \log k \cdot \log b^2)}$.*

5. EXPTIME UPPER BOUNDS FOR ENRICHED μ-CALCULI

We use Theorem 4.2 and Corollary 4.3 to establish EXPTIME upper bounds for satisfiability in the full graded μ-calculus and the hybrid graded μ-calculus.

5.1. Full graded μ-calculus.

We give a polynomial translation of formulas φ of the full graded μ-calculus into a 2GAPT A_φ that accepts the tree models of φ. We can thus decide satisfiability of φ by checking non-emptiness of $\mathcal{L}(A_\varphi)$. There is a minor technical difficulty to be overcome: we use Kripke structures with 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. For a formula φ, let $\Gamma(\varphi)$ denote the set of all atomic propositions and all propositions p_α such that α is an (atomic or inverse) program occurring in φ. The *encoding* of a tree structure $K = \langle W, R, L \rangle$ is the $2^{\Gamma(\varphi)}$-labeled tree $\langle W, L^* \rangle$ such that

$$L^*(w) = \{p \in \mathsf{Prop} \mid w \in L(p)\} \cup \{p_\alpha \mid \exists (v,w) \in R(\alpha) \text{ with } w \text{ } \alpha\text{-successor of } v \text{ in } W\}.$$

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, where in particular $|\langle n, \alpha \rangle \psi| = |[n, \alpha]\psi| = \lceil \log\ n \rceil + 1 + |\psi|$. We say that a formula φ *counts* up to b if the maximal integer in *atleast* and *allbut* formulas used in φ is $b-1$.

Theorem 5.1. *Given a sentence φ of the full graded μ-calculus that counts up to b, we can construct a 2GAPT A_φ such that A_φ*

1. *accepts exactly the encodings of tree models of φ,*
2. *has $\mathcal{O}(|\varphi|)$ states, index $\mathcal{O}(|\varphi|)$, and counting bound b.*

The construction can be done in time $\mathcal{O}(|\varphi|)$.

Proof. The automaton A_φ verifies that φ holds at the root of the encoded tree. To define the set of states, we use the Fischer-Ladner closure $\mathsf{cl}(\varphi)$ of φ. It is defined analogously to the Fischer-Ladner closure $\mathsf{cl}(\cdot)$ for the hybrid graded μ-calculus, as given in Section 3. We define A_φ as

$\langle 2^{\Gamma(\varphi)}, b, \mathsf{cl}(\varphi), \delta, \varphi, \mathcal{F}\rangle$, where the transition function δ is defined by setting, for all $\sigma \in 2^{\Gamma(\varphi)}$,

$$\begin{array}{rcl}
\delta(p,\sigma) & = & (p \in \sigma) \\
\delta(\neg p,\sigma) & = & (p \notin \sigma) \\
\delta(\psi_1 \wedge \psi_2,\sigma) & = & (\varepsilon,\psi_1) \wedge (\varepsilon,\psi_2) \\
\delta(\psi_1 \vee \psi_2,\sigma) & = & (\varepsilon,\psi_1) \vee (\varepsilon,\psi_2) \\
\delta(\lambda y.\psi(y),\sigma) & = & (\varepsilon,\psi(\lambda y.\psi(y))) \\
\delta(\langle n,a\rangle\psi,\sigma) & = & ((-1,\psi) \wedge (\varepsilon,p_{a^-}) \wedge (\langle n-1\rangle,\psi \wedge p_a)) \vee (\langle n\rangle,\psi \wedge p_a) \\
\delta(\langle n,a^-\rangle\psi,\sigma) & = & ((-1,\psi) \wedge (\varepsilon,p_a) \wedge (\langle n-1\rangle,\psi \wedge p_{a^-})) \vee (\langle n\rangle,\psi \wedge p_{a^-}) \\
\delta([n,a]\psi,\sigma) & = & ((-1,\psi) \wedge (\varepsilon,p_{a^-}) \wedge ([n],\psi \wedge p_a)) \vee ([n-1],\psi \wedge p_a) \\
\delta([n,a^-]\psi,\sigma) & = & ((-1,\psi) \wedge (\varepsilon,p_a) \wedge ([n],\psi \wedge p_{a^-})) \vee ([n-1],\psi \wedge p_{a^-})
\end{array}$$

In case $n = 0$, the conjuncts (resp. disjuncts) involving "$n-1$" are simply dropped in the last two lines.

The acceptance condition of A_φ is defined in the standard way as follows (see e.g. [KVW00]). For a fixpoint formula $\psi \in \mathsf{cl}(\varphi)$, the alternation level of ψ is the number of alternating fixpoint formulas one has to "wrap ψ with" to reach a sub-*sentence* of φ. Formally, let $\psi = \lambda y.\psi'(y)$. The *alternation level* of ψ in φ, denoted $\mathsf{al}_\varphi(\psi)$ is defined as follows ([BC96]): if ψ is a sentence, then $\mathsf{al}_\varphi(\psi) = 1$. Otherwise, let $\xi = \lambda' z.\psi''(z)$ be the innermost μ or ν subformula of φ that has ψ as a strict subformula. Then, if z is free in ψ and $\lambda' \neq \lambda$, we have $\mathsf{al}_\varphi(\psi) = \mathsf{al}_\varphi(\xi) + 1$; otherwise, $\mathsf{al}_\varphi(\psi) = \mathsf{al}_\varphi(\xi)$.

Let d be the maximum alternation level of (fixpoint) subformulas of φ. Denote by G_i the set of all ν-formulas in $\mathsf{cl}(\varphi)$ of alternation level i and by B_i the set of all μ-formulas in $\mathsf{cl}(\varphi)$ of alternation level less than or equal to i. Now, define $\mathcal{F} := \{\mathcal{F}_0, \mathcal{F}_1, \ldots, \mathcal{F}_{2d}, Q\}$ with $\mathcal{F}_0 = \emptyset$ and for every $1 \leq i \leq d$, $\mathcal{F}_{2i-1} = \mathcal{F}_{2i-2} \cup B_i$ and $\mathcal{F}_{2i} = \mathcal{F}_{2i-1} \cup G_i$. Let π be a path. By definition of $\mathcal{F}$, the minimal i with $\mathsf{Inf}(\pi) \cap F_i \neq \emptyset$ determines the alternation level and type λ of the outermost fixpoint formula $\lambda y.\psi(y)$ that was visited infinitely often on π. The acceptance condition makes sure that this formula is a ν-formula. In other words, every μ-formula that is visited infinitely often on π has a super-formula that (i) is a ν-formula and (ii) is also visited infinitely often. □

Let φ be a sentence of the full graded μ-calculus with ℓ at-least subformulas. By Theorems 3.1, 4.2, and 5.1, the satisfiability of φ can be checked in time bounded by $2^{p(|\varphi|)}$ where $p(|\varphi|)$ is a polynomial (note that, in Theorem 4.2, n, k, $\log \ell$, and $\log b$ are all in $\mathcal{O}(|\varphi|)$). This yields the desired EXPTIME upper bound. The lower bound is due to the fact that the μ-calculus is EXPTIME-hard [FL79].

Theorem 5.2. *The satisfiability problem of the full graded μ-calculus is* EXPTIME*-complete even if the numbers in the graded modalities are coded in binary.*

5.2. **Hybrid graded μ-calculus.** We reduce satisfiability in the hybrid graded μ-calculus to the emptiness problem of FEAs. Compared to the reduction presented in the previous section, 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 $\uparrow_o^a$ which do not occur in the input formula and represent an edge labeled with the atomic program a from the current node to the (unique) root node labeled by nominal o. Let $\Theta(\varphi)$ denote the set of all atomic propositions and nominals occurring in φ and all propositions p_a and $\uparrow_o^a$ such that the atomic program a and the nominal o occur in φ. Analogously to encodings of trees in the previous section,

the *encoding* of a directed quasi-forest structure $K = \langle W, R, L\rangle$ is the $2^{\Theta(\varphi)}$-labeled forest $\langle W, L^*\rangle$ such that

$$\begin{aligned} L^*(w) \;=\; & \{p \in \mathsf{Prop} \cup \mathsf{Nom} \mid w \in L(p)\} \cup \\ & \{p_a \mid \exists (v,w) \in R(a) \text{ with } w \text{ successor of } v \text{ in } W\} \cup \\ & \{\uparrow_o^a \mid \exists (w,v) \in R(a) \text{ with } L(o) = \{v\}\}. \end{aligned}$$

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 fix some information about the structures accepted by FEAs already before constructing the FEA, namely (i) the formulas from the Fischer-Ladner closure that are satisfied by each nominal and (ii) the nominals that are interpreted as the same state. This information is provided by a so-called guess. To introduce guesses formally, we need to extend the Fischer-Ladner closure $\mathsf{cl}(\varphi)$ for a formula φ of the hybrid graded μ-calculus as follows: $\mathsf{cl}(\varphi)$ has to satisfy the closure conditions given for the hybrid graded μ-calculus in Section 3 and, additionally, the following:

- if $\psi \in \mathsf{cl}(\varphi)$, then $\neg\psi \in \mathsf{cl}(\varphi)$, where $\neg\psi$ denotes the formula obtained from ψ by dualizing all operators and replacing every literal (i.e., atomic proposition, nominal, or negation thereof) with its negation.

Let φ be a formula with nominals $O = \{o_1, \ldots, o_k\}$. A *guess* for φ is a pair $(t, \sim)$ where t assigns a subset $t(o) \subseteq \mathsf{cl}(\varphi)$ to each $o \in O$ and $\sim$ is an equivalence relation on O such that the following conditions are satisfied, for all $o, o' \in O$:

(*i*) $\psi \in t(o)$ or $\neg\psi \in t(o)$ for all formulas $\psi \in \mathsf{cl}(\varphi)$;
(*ii*) $o \in t(o)$;
(*iii*) $o \sim o'$ implies $t(o) = t(o')$;
(*iv*) $o \not\sim o'$ implies $\neg o \in t(o')$.

The intuition of a guess is best understood by considering the following notion of compatibility. A directed quasi-forest structure $K = (W, R, L)$ is *compatible* with a guess $G = (t, \sim)$ if the following conditions are satisfied, for all $o, o' \in O$:

- $L(o) = \{w\}$ implies that $\{\psi \in \mathsf{cl}(\varphi) \mid K, w \models \psi\} = t(o)$;
- $L(o) = L(o')$ iff $o \sim o'$.

We construct a separate FEA $A_{\varphi,G}$ for each guess G for φ such that φ is satisfiable iff $\mathcal{L}(A_{\varphi,G})$ is non-empty for some guess G. 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 at least one of them accepts a non-empty language.

Theorem 5.3. *Given a sentence φ of the hybrid graded μ-calculus that counts up to b and a guess G for φ, we can construct a FEA $A_{\varphi,G}$ such that*

(1) *if $\langle F, V\rangle$ is the encoding of a directed quasi-forest model of φ compatible with G, then $\langle F, V\rangle \in \mathcal{L}(A_{\varphi,G})$,*
(2) *if $\mathcal{L}(A_{\varphi,G}) \neq \emptyset$, then there is an encoding $\langle F, V\rangle$ of a directed quasi-forest model of φ compatible with G such that $\langle F, V\rangle \in \mathcal{L}(A_{\varphi,G})$, and*
(3) *$A_{\varphi,G}$ has $\mathcal{O}(|\varphi|^2)$ states, index $\mathcal{O}(|\varphi|)$, and counting bound b.*

The construction can be done in time $\mathcal{O}(|\varphi|^2)$.

Proof. Let φ be a formula of the hybrid graded μ-calculus and $G = (t, \sim)$ a guess for φ. Assume that the nominals occurring in φ are $O = \{o_1, \ldots, o_k\}$. For each formula $\psi \in \mathsf{cl}(\varphi)$, atomic program a, and $\sigma \in 2^{\Theta(\varphi)}$, let

- $\mathsf{nom}_\psi^a(\sigma) = \{o \mid \psi \in t(o) \wedge \uparrow_o^a \in \sigma\}$;

- $|\mathsf{nom}^a_\psi(\sigma)|^\sim$ denote the number of equivalence classes C of $\sim$ such that some member of C is contained in $\mathsf{nom}^a_\psi(\sigma)$.

The automaton $A_{\varphi,G}$ verifies compatibility with G, and ensures that φ holds in some root. As its set of states, we use

$$Q = \mathsf{cl}(\varphi) \cup \{q_0\} \cup \{\neg o_i \vee \psi, \mid 1 \leq i \leq k \wedge \psi \in \mathsf{cl}(\varphi)\} \cup \{\mathsf{ini}_i \mid 1 \leq i \leq k\}.$$

Set $A_{\varphi,G} = \langle 2^{\Theta(\varphi)}, b, Q, \delta, q_0, \mathcal{F}\rangle$, where the transition function δ and the acceptance condition $\mathcal{F}$ is defined in the following. For all $\sigma \in 2^{\Theta(\varphi)}$, define:

$$\begin{aligned}
\delta(q_0,\sigma) &= (\langle\mathsf{root}\rangle,\varphi) \wedge \bigwedge_{1\leq i\leq k} (\langle\mathsf{root}\rangle, o_i) \wedge \bigwedge_{1\leq i\leq k} ([\mathsf{root}], \mathsf{ini}_i)\\
\delta(\mathsf{ini}_i,\sigma) &= (\varepsilon, \neg o_i) \vee \bigwedge_{\gamma\in t(o_i)} (\varepsilon,\gamma)\\
\delta(\neg p,\sigma) &= (p \notin \sigma)\\
\delta(\psi_1 \wedge \psi_2,\sigma) &= (\varepsilon,\psi_1) \wedge (\varepsilon,\psi_2)\\
\delta(\psi_1 \vee \psi_2,\sigma) &= (\varepsilon,\psi_1) \vee (\varepsilon,\psi_2)\\
\delta(\lambda y.\psi(y),\sigma) &= (\varepsilon,\psi(\lambda y.\psi(y)))\\
\delta([n,a]\psi,\sigma) &= \mathsf{false} \text{ if } |\mathsf{nom}^a_{\neg\psi}(\sigma)|^\sim > n\\
\delta([n,a]\psi,\sigma) &= ([n - |\mathsf{nom}^a_{\neg\psi}(\sigma)|^\sim], \psi \wedge p_a) \wedge \bigwedge_{o\in\mathsf{nom}^a_\psi(\sigma)} ([\mathsf{root}], \neg o \vee \psi) \text{ if } |\mathsf{nom}^a_{\neg\psi}(\sigma)|^\sim \leq n\\
\delta(\langle n,a\rangle\psi,\sigma) &= (\langle n - |\mathsf{nom}^a_\psi(\sigma)|^\sim\rangle, \psi \wedge p_a) \wedge \bigwedge_{o\in\mathsf{nom}^a_\psi(\sigma)} ([\mathsf{root}], \neg o \vee \psi)
\end{aligned}$$

In the last line, the first conjunct is omitted if $|\mathsf{nom}^a_\psi(\sigma)|^\sim > n$. The first two transition rules check that each nominal occurs in at least one root and that the encoded quasi-forest structure is compatible with the guess G. Consider the last three rules, which are concerned with graded modalities and reflect the existence of implicit back-edges to nominals. The first of these rules checks for allbut formulas that are violated purely by back-edges. The other two rules consist of two conjuncts, each. In the first conjunct, we subtract the number of nominals to which there is an implicit a-edge and that violate the formula ψ in question. This is necessary because the $\langle\cdot\rangle$ and $[\cdot]$ transitions of the automaton do not take into account implicit edges. In the second conjunct, we send a copy of the automaton to each nominal to which there is an a-edge and that satisfies ψ. Observe that satisfaction of ψ at this nominal is already guaranteed by the second rule that checks compatibility with G. We nevertheless need the second conjunct in the last two rules because, without the jump to the nominal, we will be missing paths in runs of $A_{\varphi,G}$ (those that involve an implicit back-edge). Thus, it would not be guaranteed that these paths satisfy the acceptance condition, which is defined below. This, in turn, means that the evaluation of least fixpoint formulas is not guaranteed to be well-founded. This point was missed in [SV01], and the same strategy used here can be employed to fix the construction in that paper.

The acceptance condition of $A_{\varphi,G}$ is defined as in the case of the full graded μ-calculus: let d be the maximal alternation level of subformulas of φ, which is defined as in the case of the full graded μ-calculus. Denote by G_i the set of all the ν-formulas in $\mathsf{cl}(\varphi)$ of alternation level i and by B_i the set of all μ-formulas in $\mathsf{cl}(\varphi)$ of alternation depth less than or equal to i. Now, $\mathcal{F} = \{\mathcal{F}_0, \mathcal{F}_1, \ldots, \mathcal{F}_{2d}, Q\}$, where $\mathcal{F}_0 = \emptyset$ and for every $1 \leq i \leq d$ we have $\mathcal{F}_{2i-1} = \mathcal{F}_{2i-2} \cup B_i$, and $\mathcal{F}_{2i} = \mathcal{F}_{2i-1} \cup G_i$.

It is standard to show that if $\langle F, V\rangle$ is the encoding of a directed quasi-forest model K of φ compatible with G, then $\langle F, V\rangle \in \mathcal{L}(A_{\varphi,G})$. Conversely, let $\langle F, V\rangle \in \mathcal{L}(A_{\varphi,G})$. If $\langle F, V\rangle$ is

nominal unique, i.e., if every nominal occurs only in the label of a single root, it is not hard to show that $\langle F, V\rangle$ is the encoding of a directed quasi-forest model K of φ compatible with G. However, the automaton $A_{\varphi,G}$ does not (and cannot) guarantee nominal uniqueness. To establish Point (2) of the theorem, we thus have to show that whenever $\mathcal{L}(A_{\varphi,G}) \neq \emptyset$, then there is an element of $\mathcal{L}(A_{\varphi,G})$ that is nominal unique.

Let $\langle F, V\rangle \in \mathcal{L}(A_{\varphi,G})$. From $\langle F, V\rangle$, we extract a new forest $\langle F', V'\rangle$ as follows: Let r be a run of $A_{\varphi,G}$ on $\langle F, V\rangle$. Remove all trees from F except those that occur in r as witnesses for the existential root transitions in the first transition rule. Call the modified forest F'. Now modify r into a run r' on F': simply drop all subtrees rooted at nodes whose label refers to one of the trees that are present in F but not in F'. Now, r' is a run on F' because (i) the only existential root transitions are in the first rule, and these are preserved by construction of F' and r'; and (ii) all universal root transitions are clearly preserved as well. Also, r' is accepting because every path in r' is a path in r. Thus, $\langle F', V'\rangle \in \mathcal{L}(A_{\varphi,G})$ and it is easy to see that $\langle F', V'\rangle$ is nominal unique. □

Combining Theorems 3.3, Corollary 4.3, and Theorem 5.3, we obtain an EXPTIME-upper bound for the hybrid graded μ-calculus. Again, the lower bound is from [FL79].

Theorem 5.4. *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.*

6. THE EMPTINESS PROBLEM FOR 2GAPTS

We prove Theorem 4.2 and thus show that the emptiness problem of 2GAPTs can be solved in EXPTIME. The proof is by a reduction to the emptiness problem of graded nondeterministic parity tree automata (GNPTs) as introduced in [KSV02].

6.1. **Graded nondeterministic parity tree automata.**

We introduce the graded nondeterministic parity tree automata (GNPTs) of [KSV02]. For $b > 0$, a *b-bound* is a pair in

$$B_b = \{(>, 0), (\leq, 0), (>, 1), (\leq, 1), \ldots, (>, b), (\leq, b)\}.$$

For a set X, a subset P of X, and a (finite or infinite) word $t = x_1 x_2 \cdots \in X^* \cup X^\omega$, the *weight* of P in t, denoted $\mathsf{weight}(P, t)$, is the number of occurrences of symbols in t that are members of P. That is, $\mathsf{weight}(P, t) = |\{i : x_i \in P\}|$. For example, $\mathsf{weight}(\{1, 2\}, 1241) = 3$. We say that t satisfies a b-bound $(>, n)$ with respect to P if $\mathsf{weight}(P, t) > n$, and t satisfies a b-bound $(\leq, n)$ with respect to P if $\mathsf{weight}(P, t) \leq n$.

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 $\mathsf{sat}(\theta) \subseteq 2^Y$ such that $x \in \mathsf{sat}(\theta)$ iff x satisfies θ. For an integer $b \geq 0$, a *b-counting constraint* for 2^Y is a relation $C \subseteq B(Y) \times B_b$. 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 word $t = x_1 x_2 \cdots \in (2^Y)^* \cup (2^Y)^\omega$ satisfies the b-counting constraint C if for all $\langle \theta, \xi\rangle \in C$, the word t satisfies ξ with respect to $\mathsf{sat}(\theta)$, that is, when θ is paired with $\xi = (>, n)$, at least $n + 1$ occurrences of symbols in t should satisfy θ, and when θ is paired with $\xi = (\leq, n)$, at most n occurrences satisfy θ. For example, the word $t_1 = \emptyset\{y_1\}\{y_2\}\{y_1, y_3\}$ does not satisfy the constraint C above, as the number of sets in t_1 that satisfies $y_1 \wedge y_3$ is one. On the other hand, the word $t_2 = \{y_2\}\{y_1\}\{y_1, y_2, y_3\}\{y_1, y_3\}$ satisfies C. Indeed, three sets in t_2 satisfy $y_1 \vee \neg y_2$, two sets satisfy y_3, and two sets satisfy $y_1 \wedge y_3$.

We use $\mathcal{C}(Y, b)$ to denote the set of all b-counting constraints for 2^Y. We assume that the integers in constraints are coded in binary.

We can now define *graded nondeterministic parity tree automata* (GNPTs, for short). A GNPT 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 C for 2^Y such that the cardinality of C is bounded by $\log |Q|$. For defining runs, we introduce an additional notion. Let x be a node in a Σ-labeled tree $\langle T, V \rangle$, and let $x \cdot i_1, x \cdot i_2, \dots$ be the (finitely or infinitely many) successors of x in T, where $i_j < i_{j+1}$ (the actual ordering is not important, but has to be fixed). Then we use $\mathsf{lab}(x)$ to denote the (finite or infinite) word of labels induced by the successors, i.e., $\mathsf{lab}(x) = V(x \cdot i_1)V(x \cdot i_2) \cdots$. Given a GNPT A, a *run* of A on a Σ-labeled tree $\langle T, V \rangle$ rooted in z is then a Q-labeled tree $\langle T, r \rangle$ such that

- $r(z) = q_0$ and
- for every $x \in T$, $\mathsf{lab}(x)$ satisfies $\delta(r(x), V(x))$.

Observe that, in contrast to the case of alternating automata, the input tree $\langle T, V \rangle$ and the run $\langle T, r \rangle$ share the component T. The run $\langle T, r \rangle$ is *accepting* if all its infinite paths satisfy the parity acceptance condition. A GNPT *accepts* a tree iff there exists an accepting run of the automaton on the tree. We denote by $\mathcal{L}(A)$ the set of all Σ-labeled trees that A accepts.

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 a $q' \in Q$ such that $\delta(q, \sigma) = \{\langle (\neg \theta_{q'}), (\leq, 0) \rangle\}$, where $\theta_{q'} \in B(Y)$ is such that $\mathsf{sat}(\theta_{q'}) = \{\{q'\}\}$. Thus, a FORALL automaton is very similar to a (non-graded) deterministic parity tree automaton, where the transition function maps q and σ to $\langle q', \dots, q' \rangle$ (and the out-degree of trees is not fixed). 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 at all.

We need two simple results concerning GNPTs. The following has been stated (but not proved) already in [KSV02].

Lemma 6.1. *Given a* FORALL *GNPT A_1 with n_1 states and index k_1, and a* SAFETY *GNPT A_2 with n_2 states and counting bound b_2, we can define a GNPT A with $n_1 n_2$ states, index k_1, and counting bound b_2, such that $\mathcal{L}(A) = \mathcal{L}(A_1) \cap \mathcal{L}(A_2)$.*

Proof. We can use a simple product construction. Let $A_i = (\Sigma, b_i, Q_i, \delta_i, q_{0,i}, \mathcal{F}^{(i)})$ with $Q_i \subseteq 2^{Y_i}$ for $i \in \{1, 2\}$. Assume w.l.o.g. that $Y_1 \cap Y_2 = \emptyset$. We define $A = (\Sigma, b_2, Q, \delta, (q_{0,1} \cup q_{0,2}), \mathcal{F})$, where

- $Q = \{q_1 \cup q_2 \mid q_1 \in Q_1 \text{ and } q_2 \in Q_2\} \subseteq 2^Y$, where $Y = Y_1 \uplus Y_2$;
- for all $\sigma \in \Sigma$ and $q = q_1 \cup q_2 \in Q$ with $\delta_1(q_1, \sigma) = \{\langle (\neg \theta_q), (\leq, 0) \rangle\}$ and $\delta_2(q_2, \sigma) = C$, we set $\delta(q, \sigma) = C \cup \{\langle (\neg \theta'_q), (\leq, 0) \rangle\}$, where $\theta'_q \in B(Y)$ is such that $\mathsf{sat}(\theta'_q) = \{q' \in Q \mid q' \cap Q_1 = q\}$;
- $\mathcal{F} = \{\mathcal{F}_1, \dots, \mathcal{F}_k\}$ with $\mathcal{F}_i = \{q \in Q \mid q \cap Q_1 \in \mathcal{F}_i^{(1)}\}$ if $\mathcal{F}^{(1)} = \{\mathcal{F}_1^{(1)}, \dots, \mathcal{F}_k^{(1)}\}$.

It is not hard to check that A is as required. □

The following result can be proved by an analogous product construction.

Lemma 6.2. *Given* SAFETY *GNPTs A_i with n_i states and counting bounds b_i, $i \in \{1, 2\}$, we can define a* SAFETY *GNPT A with $n_1 n_2$ states and counting bound $b = \mathsf{max}\{b_1, b_2\}$ such that $\mathcal{L}(A) = \mathcal{L}(A_1) \cap \mathcal{L}(A_2)$.*

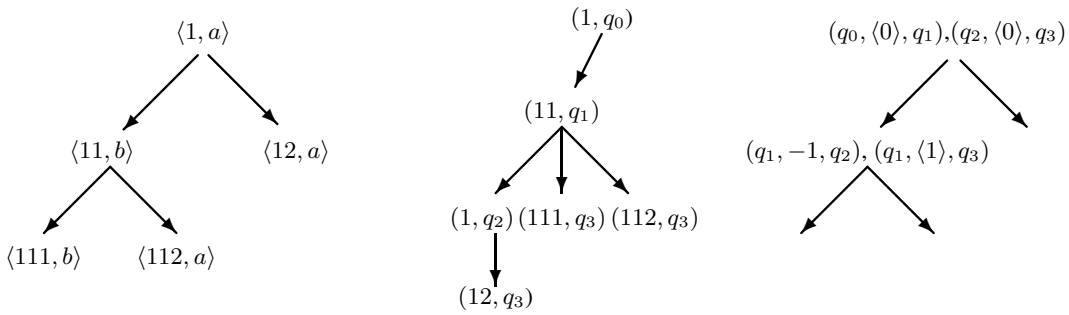

Figure 2: A fragment of an input tree, a corresponding run, and its strategy tree.

6.2. **Reduction to Emptiness of GNPTs.** We now show that the emptiness problem of 2GAPTs can be reduced to the emptiness problem of GNPTs that are only exponentially larger. Let $A = \langle \Sigma, b, Q, \delta, q_0, \mathcal{F}\rangle$ be a 2GAPT. We recall that δ is a function from $Q \times \Sigma$ to $B^+(D_b^- \times Q)$, with $D_b^- := \langle\langle b\rangle\rangle \cup [[b]] \cup \{-1, \varepsilon\}$. A *strategy tree* for A is a $2^{Q \times D_b^- \times Q}$-labeled tree $\langle T, \mathsf{str}\rangle$. Intuitively, the purpose of a strategy tree is to guide the automaton A by pre-choosing transitions that satisfy the transition relation. For each label $w = \mathsf{str}(x)$, we use $\mathsf{head}(w) = \{q \mid (q, c, q') \in w\}$ to denote the set of states for which str chooses transitions at x. Intuitively, if A is in state $q \in \mathsf{head}(w)$, str tells it to execute the transitions $\{(c, q') \mid (q, c, q') \in w\}$. In the following, we usually consider only the str part of a strategy tree. Let $\langle T, V\rangle$ be a Σ-labeled tree and $\langle T, \mathsf{str}\rangle$ a strategy tree for A, based on the same T. Then str is a *strategy for A on V* if for all nodes $x \in T$ and all states $q \in Q$, we have:

(1) $\delta(q_0, V(\mathsf{root}(T))) = \mathsf{true}$ or $q_0 \in \mathsf{head}(\mathsf{str}(\mathsf{root}(T)))$;
(2) if $q \in \mathsf{head}(\mathsf{str}(x))$, then the set $\{(c, q') : (q, c, q') \in \mathsf{str}(x)\}$ satisfies $\delta(q, V(x))$,
(3) if $(q, c, q') \in \mathsf{str}(x)$ with $c \in \{-1, \varepsilon\}$, then (i) $x \cdot c$ is defined and (ii) $\delta(q', V(x \cdot c)) = \mathsf{true}$ or $q' \in \mathsf{head}(\mathsf{str}(x \cdot c))$.

If A is understood, we simply speak of a strategy on V.

Example 6.3. Let $A = \langle \Sigma, b, Q, \delta, q_0, \mathcal{F}\rangle$ be a 2GAPT such that $\Sigma = \{a, b, c\}$, $Q = \{q_0, q_1, q_2, q_3\}$, and δ is such that $\delta(q, a) = (\langle 0\rangle, q_1) \vee (\langle 0\rangle, q_3)$ for $q \in \{q_0, q_2\}$, and $\delta(q_1, b) = ((-1, q_2) \wedge (\langle 1\rangle, q_3)) \vee ([1], q_1)$. Consider the trees depicted in Figure 2. From left to right, the first tree $\langle T, V\rangle$ is a fragment of the input tree, the second tree is a fragment of a run $\langle T_r, r\rangle$ of A on $\langle T, V\rangle$, and the third tree is a fragment of a strategy tree suggesting this run. In a label $\langle w, a\rangle$ of the input tree, w is the node name and $a \in \Sigma$ the label in the tree. In the run and strategy tree, only the labels are given, but not the node names.

Strategy trees do not give full information on how to handle transitions $(\langle n\rangle, q)$ and $([n], q)$ as they do not say which successors should be used when executing them. This is compensated by *promise trees*. A promise tree for A is a $2^{Q \times Q}$-labeled tree $\langle T, \mathsf{pro}\rangle$. Intuitively, if a run that proceeds according to pro visits a node x in state q and chooses a move $(\langle n\rangle, q')$ or $([n], q')$, then the successors $x \cdot i$ of x that inherit q' are those with $(q, q') \in \mathsf{pro}(x \cdot i)$. Let $\langle T, V\rangle$ be a Σ-labeled tree, str a strategy on V, and $\langle T, \mathsf{pro}\rangle$ a promise tree. We call pro a *promise for A on* str if the states promised to be visited by pro satisfy the transitions chosen by str, i.e., for every node $x \in T$, the following hold:

(1) for every $(q, \langle n \rangle, q') \in \mathsf{str}(x)$, there is a subset $M \subseteq \mathsf{succ}(x)$ of cardinality $n+1$ such that each $y \in M$ satisfies $(q, q') \in \mathsf{pro}(y)$;
(2) for every $(q, [n], q') \in \mathsf{str}(x)$, there is a subset $M \subseteq \mathsf{succ}(x)$ of cardinality n such that each $y \in \mathsf{succ}(x) \setminus M$ satisfies $(q, q') \in \mathsf{pro}(y)$;
(3) if $(q, q') \in \mathsf{pro}(x)$, then $\delta(q', V(x)) = \mathsf{true}$ or $q' \in \mathsf{head}(\mathsf{str}(x))$.

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

- 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 $x_{i+1} = x_i \cdot j \in T$, and $(q_i, q_{i+1}) \in \mathsf{pro}(x_{i+1})$.

Let $\mathcal{F} = \{\mathcal{F}_1, \dots, \mathcal{F}_k\}$. For each state $q \in Q$, let $\mathsf{index}(q)$ be the minimal i such that $q \in \mathcal{F}_i$. For a trace π, let $\mathsf{index}(\pi)$ be the minimal index of states that occur infinitely often in π. Then, π *satisfies* $\mathcal{F}$ if it has even index. The strategy str and promise pro are *accepting* if all the traces induced by str and pro satisfy $\mathcal{F}$.

In [KSV02], it was shown that a necessary and sufficient condition for a tree $\langle T, V \rangle$ to be accepted by a one-way GAPT is the existence of a strategy str on V and a promise pro on str that are accepting. We establish the same result for the case of 2GAPTs.

Lemma 6.4. *A 2GAPT A accepts $\langle T, V \rangle$ iff there exist a strategy* str *for A on V and a promise* pro *for A on* str *that are accepting.*

Proof. Let $A = \langle \Sigma, b, Q, \delta, q_0, \mathcal{F} \rangle$ be a 2GAPT with $\mathcal{F} = \{\mathcal{F}_1, \dots, \mathcal{F}_k\}$, and let $\langle T, V \rangle$ be the input tree. Suppose first that A accepts $\langle T, V \rangle$. Consider a two-player game on Σ-labeled trees, Protagonist vs. Antagonist, such that Protagonist is trying to show that A accepts the tree, and Antagonist is challenging that. A configuration of the game is a pair in $T \times Q$. The initial configuration is $(\mathsf{root}(T), q_0)$. Consider a configuration (x, q). Protagonist is first to move and chooses a set $P_1 = \{(c_1, q_1), \dots, (c_m, q_m)\} \subseteq D_b^- \times Q$ that satisfies $\delta(q, V(x))$. If $\delta(q, V(x)) = \mathsf{false}$, then Antagonist wins immediately. If P_1 is empty, Protagonist wins immediately. Antagonist responds by choosing an element (c_i, q_i) of P_1. If $c_i \in \{-1, \varepsilon\}$, then the new configuration is $(x \cdot c_i, q_i)$. If $x \cdot c_i$ is undefined, then Antagonist wins immediately. If $c_i = \langle n \rangle$, Protagonist chooses a subset $M \subseteq \mathsf{succ}(x)$ of cardinality $n+1$, Antagonist wins immediately if there is no such subset and otherwise responds by choosing an element y of M. Then, the new configuration is (y, q_i). If $c_i = [n]$, Protagonist chooses a subset $M \subseteq \mathsf{succ}(x)$ of cardinality at most n, Antagonist wins immediately if there is no such subset and otherwise responds by choosing an element y of $\mathsf{succ}(x) \setminus M$. Protagonist wins immediately if there is no such element. Otherwise, the new configuration is (y, q_i).

Consider now an infinite game Y, that is, an infinite sequence of immediately successive game configurations. Let $\mathsf{Inf}(Y)$ be the set of states in Q that occur infinitely many times in Y. Protagonist wins if there is an even $i > 0$ for which $\mathsf{Inf}(Y) \cap \mathcal{F}_i \neq \emptyset$ and $\mathsf{Inf}(Y) \cap \mathcal{F}_j = \emptyset$ for all $j < i$. It is not difficult to see that a winning strategy of Protagonist against Antagonist is essentially a representation of a run of A on $\langle T, V \rangle$ and vice versa. Thus, such a winning strategy exists iff A accepts this tree. The described game meets the conditions in [Jut95]. It follows that if Protagonist has a winning strategy, then it has a memoryless strategy, i.e., a strategy whose moves do not depend on the history of the game, but only on the current configuration.

Since we assume that A accepts the input tree $\langle T, V \rangle$, Protagonist has a memoryless winning strategy on $\langle T, V \rangle$. This winning strategy can be used to build a strategy str on V and a promise pro on str in the following way. For each $x \in T$, $\mathsf{str}(x)$ and $\mathsf{pro}(x)$ are the smallest sets such that, for

all configurations (x, q) occurring in Protagonist's winning strategy, if Protagonist chooses a subset $P_1 = \{(c_1, q_1), \dots, (c_m, q_m)\}$ of $D_b^- \times Q$ in the winning strategy, then we have

(i) $\{q\} \times P_1 \subseteq \mathsf{str}(x)$ and
(ii) for each atom (c_i, q_i) of P_1 with $c_i = \langle n \rangle$ (resp. $c_i = [n]$) if $M = \{y_1, \dots, y_{n+1}\}$ (resp. $M = \{y_1, \dots, y_n\}$) is the set of successors chosen by Protagonist after Antagonist has chosen (c_i, q_i), then we have $(q, q_i) \in \mathsf{pro}(y)$ for each $y \in M$ (resp. for each $y \in \mathsf{succ}(x) \setminus M$).

Using the definition of games and the construction of str, it is not hard to show that str is indeed a strategy on V. Similarly, it is easy to prove that pro is a promise on str. Finally, it follows from the definition of wins of Protagonist that str and pro are accepting.

Assume now that there exist a strategy str on V and a promise pro on str that are accepting. Using str and pro, it is straightforward to inductively build an accepting run $\langle T_r, r\rangle$ of A on $\langle T, V\rangle$:

- start with introducing the root z of T_r, and set $r(z) = (\mathsf{root}(T), q_0)$;
- if y is a leaf in T_r with $r(y) = (x, q)$ and $\delta(q, V(x)) \neq \mathsf{true}$, then do the following for all $(q, c, q') \in \mathsf{str}(x)$:
  - If $c = -1$ or $c = \varepsilon$, then add a fresh successor $y \cdot j$ to y in T_r and set $r(y \cdot j) = (x \cdot c, q')$;
  - If $c = \langle n \rangle$ or $c = [n]$, then for each $j \in \mathbb{N}$ with $(q, q') \in \mathsf{pro}(x \cdot j)$, add a fresh successor $y \cdot j'$ to y in T_r and set $r(y \cdot j') = (x \cdot j, q')$.

By Condition (3) of strategy trees, $y \cdot j$ is defined in the induction step. Using the properties of strategies on V and of promises on str, it is straightforward to show that $\langle T_r, r\rangle$ is a run. It thus remains to prove that $\langle T_r, r\rangle$ is accepting. Let π be a path in $\langle T_r, r\rangle$. By definition of traces induced by str and pro, the labeling of π is a trace induced by str and pro. Since str and pro are accepting, so is π. □

Strategy and promise trees together serve as a witness for acceptance of an input tree by a 2GAPT that, in contrast to a run $\langle T_r, r\rangle$, has the same tree structure as the input tree. To translate 2GAPTs into GNPTs, we still face the problem that traces in strategies and promises can move both up and down. To restrict attention to unidirectional paths, we extend to our setting the notion of annotation as defined in [Var98]. Annotations allow decomposing a trace of a strategy and a promise into a downward part and several finite parts that are *detours*, i.e., divert from the downward trace and come back to the point of diversion.

Let $A = \langle \Sigma, b, Q, \delta, q_0, \mathcal{F}\rangle$ be a 2GAPT. An *annotation tree* for A is a $2^{Q \times \{1,\dots,k\} \times Q}$-labeled tree $\langle T, \mathsf{ann}\rangle$. Intuitively, $(q, i, q') \in \mathsf{ann}(x)$ means that from node x and state q, A can make a detour and comes back to x with state q' such that i is the smallest index of all states that have been seen along the detour. Let $\langle T, V\rangle$ be a Σ-labeled tree, str a strategy on V, pro a promise on str, and $\langle T, \mathsf{ann}\rangle$ an annotation tree. We call ann an annotation for A on str and pro if for every node $x \in T$, the following conditions are satisfied:

(1) If $(q, \varepsilon, q') \in \mathsf{str}(x)$ then $(q, \mathsf{index}(q'), q') \in \mathsf{ann}(x)$;
(2) if $(q, j', q') \in \mathsf{ann}(x)$ and $(q', j'', q'') \in \mathsf{ann}(x)$, then $(q, \mathsf{min}(j', j''), q'') \in \mathsf{ann}(x)$;
(3) if (i) $x = y \cdot i$, (ii) $(q, -1, q') \in \mathsf{str}(x)$, (iii) $(q', j, q'') \in \mathsf{ann}(y)$ or $q' = q''$ with $\mathsf{index}(q') = j$, (iv) $(q'', \langle n \rangle, q''')$ or $(q'', [n], q''')$ is in $\mathsf{str}(y)$, and (v) $(q'', q''') \in \mathsf{pro}(x)$, then $(q, \mathsf{min}(\mathsf{index}(q'), j, \mathsf{index}(q''')), q''') \in \mathsf{ann}(x)$;
(4) if (i) $y = x \cdot i$, (ii) $(q, \langle n \rangle, q')$ or $(q, [n], q')$ is in $\mathsf{str}(x)$, (iii) $(q, q') \in \mathsf{pro}(y)$, (iv) $(q', j, q'') \in \mathsf{ann}(y)$ or $q' = q''$ with $\mathsf{index}(q') = j$, and (v) $(q'', -1, q''') \in \mathsf{str}(y)$, then $(q, \mathsf{min}(\mathsf{index}(q'), j, \mathsf{index}(q''')), q''') \in \mathsf{ann}(x)$.

Example 6.5. Reconsider the 2GAPT $A = \langle \Sigma, b, Q, \delta, q_0, \mathcal{F}\rangle$ from Example 6.3, as well as the fragments of the input tree $\langle T, V\rangle$ and the strategy str on $\langle T, V\rangle$ depicted in Figure 2. Assume

that there is a promise pro on str with $(q_0, q_1) \in \mathsf{pro}(11)$ telling the automaton that if it executes $(\langle 0\rangle, q_1)$ in state q_0 at node 1, it should send a copy in state q_1 to node 11. Using $\mathsf{str}(1)$ and Condition (4) of annotations, we can now deduce that, in any annotation ann on str and pro, we have $(q_0, j, q_2) \in \mathsf{ann}(1)$ with j the minimum of the indexes of q_0, q_1, and q_2.

Given an annotation tree $\langle T, \mathsf{ann}\rangle$ on str and pro, a *downward trace* π induced by str, pro, and ann is a sequence $(x_0, q_0, t_0), (x_1, q_1, t_1), \dots$ of triples, where $x_0 = \mathsf{root}(T)$, q_0 is the initial state of A, and for each $i \geq 0$, one of the following holds:

(†) t_i is $(q_i, c, q_{i+1}) \in \mathsf{str}(x_i)$ 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 $x_{i+1} = x_i \cdot d$

(‡) t_i is $(q_i, d, q_{i+1}) \in \mathsf{ann}(x_i)$ for some $d \in \{1, \dots, k\}$, and $x_{i+1} = x_i$.

In the first case, $\mathsf{index}(t_i)$ is the minimal j such that $q_{i+1} \in \mathcal{F}_j$ and in the second case, $\mathsf{index}(t_i) = d$. For a downward trace π, $\mathsf{index}(\pi)$ is the minimal $\mathsf{index}(t_i)$ for all t_i occurring infinitely often in π. Note that a downward trace π can loop indefinitely at a node $x \in T$ when, from some point $i \geq 0$ on, all the t_j, $j \geq i$, are elements of ann (and all the x_j are x). We say that a downward trace π *satisfies* $\mathcal{F} = \{\mathcal{F}_1, \dots, \mathcal{F}_k\}$ if $\mathsf{index}(\pi)$ is even. Given a strategy str, a promise pro on str, an annotation ann on str and pro, we say that ann is *accepting* if all downward traces induced by str, pro, and ann satisfy $\mathcal{F}$.

Lemma 6.6. *A 2GAPT A accepts $\langle T, V\rangle$ iff there exist a strategy* str *for A on V, a promise* pro *for A on* str*, and an annotation* ann *for A on* str *and* pro *such that* ann *is accepting.*

Proof. Suppose first that A accepts $\langle T, V\rangle$. By Lemma 6.4, there is a strategy str on V and a promise pro on str which are accepting. By definition of annotations on str and pro, it is obvious that there exists a unique smallest annotation ann on str and pro in the sense that, for each node x in T and each annotation ann′, we have $\mathsf{ann}(x) \subseteq \mathsf{ann}'(x)$. We show that ann is accepting. Let $\pi = (x_0, q_0, t_0), (x_1, q_1, t_1), \dots$ be a downward trace induced by str, pro, and ann. It is not hard to construct a trace $\pi' = (x'_0, q'_0), (x'_1, q'_1), \dots$ induced by str and pro that is accepting iff π is: first expand π by replacing elements in π of the form (‡) with the detour asserted by ann, and then project π on the first two components of its elements. Details are left to the reader.

Conversely, suppose that there exist a strategy str on V, a promise pro on str, and an annotation ann on str and pro such that ann is accepting. By Lemma 6.4, it suffices to show that str and pro are accepting. Let $\pi = (x_0, q_0), (x_1, q_1), \dots$ be a trace induced by str and pro. It is possible to construct a downwards trace π' induced by str, pro, and ann that is accepting iff π is: whenever the step from (x_i, q_i) to (x_{i+1}, q_{i+1}) is such that $x_{i+1} = x_i \cdot c$ for some $c \in \mathbb{N}$, the definition of traces induced by str and pro ensures that there is a $t_i = (q_i, c, q_{i+1}) \in \mathsf{str}(x_i)$ such that the conditions from (†) are satisfied; otherwise, we consider the maximal subsequence $(x_i, q_i), \dots, (x_j, q_j)$ of π such that $x_j = x_i \cdot c$ for some $c \in \mathbb{N}$, and replace it with $(x_i, q_i), (x_j, q_j)$. By definition of annotations, there is $t_i = (q_i, d, q_{i+1}) \in \mathsf{ann}(x_i)$ such that the conditions from (‡) are satisfied. Again, we leave details to the reader. □

In the following, we combine the input tree, the strategy, the promise, and the annotation into one tree $\langle T, (V, \mathsf{str}, \mathsf{pro}, \mathsf{ann})\rangle$. The simplest approach to representing the strategy as part of the input tree is to additionally label the nodes of the input tree with an element of $2^{Q \times D_b^- \times Q}$. However, we can achieve better bounds if we represent strategies more compactly. Indeed, it suffices to store for every pair of states $q, q' \in Q$, at most four different tuples (q, c, q'): two for $c \in \{\varepsilon, -1\}$ and two for the minimal n and maximal n' such that $(q, [n], q'), (q, \langle n'\rangle, q') \in \mathsf{str}(y)$. Call the set of all representations of strategies L_{str}. We can now define the alphabet of the combined trees. Given

an alphabet Σ for the input tree, let Σ' denote the extended signature for the combined trees, i.e., $\Sigma' = \Sigma \times L_{\mathsf{str}} \times 2^{Q\times Q} \times 2^{Q\times\{1,\dots,k\}\times Q}$.

Theorem 6.7. *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 $2^{\mathcal{O}(kn^2\cdot\log k\cdot\log b^2)}$ states, index nk, and b-counting constraints such that A' accepts a tree iff A accepts its projection on Σ.*

Proof. Let $A = \langle\Sigma, b, Q, \delta, q_0, \mathcal{F}\rangle$ with $\mathcal{F} = \{\mathcal{F}_1, \dots, \mathcal{F}_k\}$. The automaton A' is the intersection of three automata A_1, A_2, and A_3. The automaton A_1 is a SAFETY GNPT, and it accepts a tree $\langle T, (V, \mathsf{str}, \mathsf{pro}, \mathsf{ann})\rangle$ iff str is a strategy on V and pro is a promise on str. It is similar to the corresponding automaton in [KSV02], but additionally has to take into account the capability of 2GAPTs to travel upwards. The state set of A_1 is $Q_1 := 2^{(Q\times Q)\cup Q}$. Let $P \in Q_1$. Intuitively,

(a) pairs $(q, q') \in P$ represent obligations for pro in the sense that if a node x of an input tree receives state P in a run of A, then (q, q') is obliged to be in $\mathsf{pro}(x)$;
(b) states $q \in P$ are used to memorize $\mathsf{head}(\mathsf{str}(y))$ of the predecessor y of x.

This behaviour is easily implemented via A_1's transition relation. Using false in the transition function of A_1 and thus ensuring that the automaton blocks when encountering an undesirable situation, it is easy to enforce Conditions (2) to (3) of strategies, and Condition (3) of promises. The initial state of A_1 is $\{(q_0, q_0)\}$, which together with Condition (3) of promises enforces Condition (1) of strategies. It thus remains to treat Conditions (1) and (2) of promises. This is again straightforward using the transition function. For example, if $(q, \langle n\rangle, q') \in \mathsf{str}(x)$, then we can use the conjunct $\langle(q, q'), (>, n)\rangle$ in the transition. Details of the definition of A_1 are left to the reader. Clearly, the automaton A_1 has $2^{\mathcal{O}(n^2)}$ states and counting bound b.

The remaining automata A_2 and A_3 do not rely on the gradedness of GNPTs. The automaton A_2 is both a SAFETY and FORALL GNPT. It accepts a tree $\langle T, (V, \mathsf{str}, \mathsf{pro}, \mathsf{ann})\rangle$ iff ann is an annotation. More precisely, A_2 checks that all conditions of annotations hold for each node x of the input tree. The first two conditions are checked locally by analyzing the labels $\mathsf{str}(x)$ and $\mathsf{ann}(x)$. The last two conditions require to analyze $\mathsf{pro}(x)$, $\mathsf{str}(y)$, and $\mathsf{ann}(y)$, where y is the parent of x. To access $\mathsf{str}(y) \subseteq Q \times D_b^- \times Q$ and $\mathsf{ann}(y) \subseteq Q \times \{1, \dots, k\} \times Q$ while processing x, A_2 must memorize these two sets in its states. Regarding $\mathsf{str}(y)$, it suffices to memorize the representation from L_{str}. The number of such representations is $(4b^2)^{n^2}$, which is bounded by $2^{\mathcal{O}(n^2\cdot\log b^2)}$. There are 2^{kn^2} different annotations, and thus the overall number of states of A_2 is bounded by $2^{\mathcal{O}(kn^2\cdot\log b^2)}$.

The automaton A_3 is a FORALL GNPT, and it accepts a tree $\langle T, (V, \mathsf{str}, \mathsf{pro}, \mathsf{ann})\rangle$ iff ann is accepting. By Lemma 6.6, it thus follows that A' accepts $\langle T, (V, \mathsf{str}, \mathsf{pro}, \mathsf{ann})\rangle$ iff A accepts $\langle T, V\rangle$. The automaton A_3 extends the automaton considered in [Var98] by taking into account promise trees and graded moves in strategies.

We construct A_3 in several steps. We first define a nondeterministic parity word automaton (NPW) U over Σ'. An input word to U corresponds to a path in an input tree to A'. We build U such that it accepts an input word/path if this path gives rise to a downward trace that violates the acceptance condition $\mathcal{F}$ of A. An NPW is a tuple $\langle\Sigma, S, M, s_0, \mathcal{F}\rangle$, where Σ is the input alphabet, S is the set of states, $M : S \to 2^S$ is the transition function, $s_0 \in S$ is the initial state, and $\mathcal{F} = \{\mathcal{F}_1, \mathcal{F}_2 \dots, \mathcal{F}_k\}$ is a parity acceptance condition. Given a word $w = a_0a_1 \dots \in \Sigma^\omega$, a run $r = q_0q_1 \cdots$ of U on w is such that $q_0 = s_0$ and $q_{i+1} \in M(q_i, a_i)$ for all $i \geq 0$.

We define $U = \langle\Sigma', S, M, s_0, \mathcal{F}'\rangle$ such that $S = (Q \times Q \times \{1, \dots, k\}) \cup \{q_{acc}\}$. Intuitively, a run of U describes a downward trace induced by str, pro, and ann on the input path. Suppose that x is the i-th node in an input path to U, r is a run of U on that path, and the i-th state in r is

$\langle q, q_{prev}, j\rangle$. This means that r describes a trace in which the state of A on the node x is q, while the previous state at the parent y of x was q_{prev}. Thus, A has executed a transition $(\langle b\rangle, q)$ or $([b], q)$ to reach state q at x. For reaching the state q_{prev} at y, A may or may not have performed a detour at y as described by ann. The j in $\langle q, q_{prev}, j\rangle$ is the minimum index of q and any state encountered on this detour (if any).

We now define the transition function M formally. To this end, let $\langle q, q_{prev}, j\rangle \in S$ and let $\sigma = (V(x), \mathsf{str}(x), \mathsf{pro}(x), \mathsf{ann}(x))$. To define $M(\langle q, q_{prev}, j\rangle, \sigma)$, we distinguish between three cases:

(1) if $(q_{prev}, q) \notin \mathsf{pro}(x)$, then $M(\langle q, q_{prev}, j\rangle, \sigma) = \emptyset$;
(2) otherwise and if $H = \{c : (q, c, q) \in \mathsf{ann}(x)\}$ is non-empty and some member of H has an odd index, set $M(\langle q, q_{prev}, j\rangle, \sigma) = \{q_{acc}\}$;
(3) if neither (1) nor (2) apply, then we put $\langle q', q'_{prev}, j'\rangle \in M(\langle q, q_{prev}, j\rangle, \sigma)$ iff
   - $(q, c, q') \in \mathsf{str}(x)$, with $c \in \langle\langle b\rangle\rangle \cup [[b]]$, $q'_{prev} = q$ and $j' = \mathsf{index}(q')$; or
   - $(q, d, q'_{prev}) \in \mathsf{ann}(x)$ for some d, $(q'_{prev}, c, q') \in \mathsf{str}(x)$ for some $c \in \langle\langle b\rangle\rangle \cup [[b]]$, and $j' = \mathsf{min}(d, \mathsf{index}(q'))$.

In addition, $M(q_{acc}, \sigma) = \{q_{acc}\}$, for all $\sigma \in \Sigma'$. For (1), note that if $(q_{prev}, q) \notin \mathsf{pro}(x)$, then pro does not permit downwards traces in which A switches from q_{prev} to q when moving from the parent of x to x. Thus, the current run of U does not correspond to a downward trace, and U does not accept. The purpose of (2) is to check for traces that "get caught" at a node.

The initial state s_0 of U is defined as $\langle q_0, q_0, \ell\rangle$, where ℓ is such that $q_0 \in \mathcal{F}_\ell$. Note that the choice of the second element is arbitrary, as the local promise at the root of the input tree is irrelevant. Finally, the parity condition is $\mathcal{F}' = \{\mathcal{F}'_1, \mathcal{F}'_2, \ldots, \mathcal{F}'_{k+1}\}$, where $\mathcal{F}'_1 = \emptyset$, $\mathcal{F}'_2 = Q \times Q \times \{1\} \cup \{q_{acc}\}$ and for each ℓ with $2 < \ell \leq k+1$, we have $\mathcal{F}'_\ell = Q \times Q \times \{\ell - 1\}$. Thus, U accepts a word if this word corresponds to a path of the input tree on which there is a non-accepting trace.

In order to get A_3, we co-determinize the NPW U and expand it to a tree automaton, i.e., a FORALL GNPT on Σ'. That is, we first construct a deterministic parity word automaton $\widetilde{U}$ that complements U, and then replace a transition $\widetilde{M}(q, \sigma) = q'$ in $\widetilde{U}$ by a transition

$$M_t(q, \sigma) = \{\langle(\neg\theta_{q'}), (\leq, 0)\rangle\}$$

in A_3 where the states of $\widetilde{U}$ are encoded by some set Y of variables and for every state q', the formula $\theta_{q'} \in B(Y)$ holds only in the subset of Y that encodes q'. By [Saf89, Tho97], the automaton $\widetilde{U}$ has $(nk)^{nk} \leq 2^{nk \cdot \log nk}$ states and index nk, thus so does A_3.

By Lemma 6.2, we can intersect the two SAFETY automata A_1 and A_2 obtaining a SAFETY automaton with $2^{\mathcal{O}(kn^2 \cdot \log b^2)}$ states and counting bound b. Moreover, by Lemma 6.1, the obtained SAFETY automaton can be intersected with the FORALL automaton A_3 yielding the desired GNPT A' with $2^{\mathcal{O}(kn^2 \cdot \log k \cdot \log b^2)}$ states, counting bound b, and index nk. □

6.3. **Emptiness of GNPTs.** By extending results of [KV98, KVW00, KSV02], we provide an algorithm for deciding emptiness of GNPTs. The general idea is to translate GNPTs into alternating (non-graded) parity automata on words, and then to use an existing algorithm from [KV98] for deciding emptiness of the latter.

A *singleton-alphabet GNPT on full ω-trees (ω-1GNPT)* is a GNPT that uses a singleton alphabet $\{a\}$ and admits only a single input tree $\langle T_\omega, V\rangle$, where T_ω is the full ω-tree $\mathbb{N}^+$ and V labels every node with the only symbol a. Our first aim is to show that every GNPT can be converted

into an ω-1GNPT such that (non)emptiness is preserved. We first convert to a 1GNPT, which is a single-alphabet GNPT.

Lemma 6.8. *Let $A = \langle \Sigma, b, Q, \delta, q_0, \mathcal{F}\rangle$ be a GNPT. Then there is a 1GNPT $A' = \langle \{a\}, b, Q', \delta', q_0', \mathcal{F}'\rangle$ with $L(A) = \emptyset$ iff $L(A') = \emptyset$ and $|Q'| \leq |Q| \times |\Sigma| + 1$.*

Proof. Let $Q \subseteq 2^Y$. We may assume w.l.o.g. that $\Sigma \subseteq 2^Z$ for some set Z with $Z \cap Y = \emptyset$. Now define the components of A' as follows:

- $Q' = \{\{s\}\} \cup \{q \cup \sigma, \mid q \in Q \wedge \sigma \in \Sigma\} \subseteq 2^{Y'}$, where $Y' = Y \uplus Z \uplus \{s\}$;
- $q_0' = \{s\}$;
- $\delta'(\{s\}, a) = \{\langle \mathsf{true}, (\leq, 1)\rangle, \langle \bigwedge_{y \in q_0} y \wedge \bigwedge_{y \in Y \setminus q_0} \neg y, (>, 0)\rangle, \langle s, (\leq, 0)\rangle\}$;
- $\delta'(q, a) = \delta(q \cap Y, q \cap Z) \cup \{\langle s, (\leq, 0)\rangle\}$ for all $q \in Q$ with $q \neq \{s\}$;
- $\mathcal{F}' = \{\mathcal{F}_1', \ldots, \mathcal{F}_k'\}$ with $\mathcal{F}_i' = \{q \in Q' \mid q \cap Q \in \mathcal{F}_i\}$ if $\mathcal{F} = \{\mathcal{F}_1, \ldots, \mathcal{F}_k\}$.

It is easy to see that A accepts $\langle T, V\rangle$ iff A' accepts $\langle T', V'\rangle$, where T' is obtained from T by adding an additional root, and V' assigns the label a to every node in T'. Intuitively, the additional root enables A' to "guess" a label at the root of the original tree. Then, the label will be guessed iteratively. □

In the next step, we translate to ω-1GNPTs.

Lemma 6.9. *Let $A = \langle \{a\}, b, Q, \delta, q_0, \mathcal{F}\rangle$ be a 1GNPT. Then there exists an ω-1GNPT $A' = \langle \{a\}, b, Q', \delta', q_0, \mathcal{F}'\rangle$ such that $\mathcal{L}(A) = \emptyset$ iff $\mathcal{L}(A') = \emptyset$ and $|Q'| = |Q| + 1$.*

Proof. Define the components of A' as follows:

- $Q' = Q \cup \{\{\bot\}\} \subseteq 2^{Y'}$, where $Y' = Y \uplus \{\bot\}$;
- if $\delta(q, a) = \{\langle \theta_1, \xi_1\rangle, \ldots, \langle \theta_k, \xi_k\rangle\}$, set $\delta'(q, a) = \{\langle \theta_1 \wedge \neg\bot, \xi_1\rangle, \ldots, \langle \theta_k \wedge \neg\bot, \xi_k\rangle\}$, for all $q \in Q$ with $\bot \notin q$;
- $\delta'(q, a) = \{\langle \neg\bot, (\leq, 0)\rangle\}$ for all $q \in Q$ with $\bot \in q$.
- $\mathcal{F}' = \{\mathcal{F}_1', \ldots, \mathcal{F}_k'\}$ with $\mathcal{F}_1' = \mathcal{F}_1$, and $\mathcal{F}_i' = \mathcal{F}_i \cup \{q \in Q' \mid \bot \in q\}$, for $2 \leq i \leq k$, if $\mathcal{F} = \{\mathcal{F}_1, \ldots, \mathcal{F}_k\}$.

It is easy to see that $\mathcal{L}(A) \neq \emptyset$ iff A' accepts $\langle T_\omega, V\rangle$. Accepting runs can be translated back and forth. When going from runs of A to runs of A', this involves of the children of each node with nodes labeled $\{\bot\}$. □

We are now ready to translate GNPTs to alternating word automata. A *single-alphabet alternating parity word automaton (1APW)* is a tuple $A = \langle \{a\}, Q, \delta, q_0, \mathcal{F}\rangle$, where $\{a\}$ is the alphabet, Q, q_0, and $\mathcal{F}$ are as in FEAs, and $\delta : Q \times \{a\} \to B^+(Q)$. There is only a single possible input to a 1APW, namely the infinite word $aaa\cdots$. Intuitively, if A is in state q on the i-th position of this word and $\delta(q, a) = q' \vee (q \wedge q'')$, then A can send to position $i + 1$ either a copy of itself in state q' or one copy in state q and one in state q''. The input word is accepted iff there is an accepting run of A, where a *run* is a Q-labeled tree $\langle T_r, r\rangle$ such that

- $r(\mathsf{root}(T_r)) = q_0$;
- for all $y \in T_r$ with $r(y) = q$ and $\delta(q, a) = \theta$, there is a (possibly empty) set $S \subseteq Q$ such that S satisfies θ and for all $q' \in S$, there is $j \in \mathbb{N}$ such that $y \cdot j \in T_r$ and $r(y \cdot j) = q'$.

As for FEAs, a run $\langle T_r, r\rangle$ is *accepting* if all its infinite paths satisfy the acceptance condition.

For an ω-1GNPT $A = \langle \{a\}, b, Q, \delta, q_0, \mathcal{F}\rangle$, $q \in Q$, and $P \subseteq Q$, the function $\mathsf{is\_mother}_A(q, P)$ returns true if there is an infinite word $t \in P^\omega$ that satisfies the counting constraint $\delta(q, a)$, and false otherwise.

Lemma 6.10. *For every ω-1GNPT $A = \langle \{a\}, b, Q, \delta, q_0, \mathcal{F}\rangle$, the 1APW $A' = \langle \{a\}, Q, \delta', q_0, \mathcal{F}\rangle$ is such that $\mathcal{L}(A) = \emptyset$ iff $\mathcal{L}(A') = \emptyset$, where for all $q \in Q$,*

$$\delta'(q,a) = \bigvee_{P \subseteq Q \text{ s.t. } \mathsf{is\_mother}_A(q,P)} \bigwedge_{q \in P} q.$$

Proof. (sketch) First assume that $\langle T_\omega, V\rangle \in \mathcal{L}(A)$. Then there exists an accepting run $\langle T_\omega, r\rangle$ of A on $\langle T_\omega, V\rangle$. It is not difficult to verify that $\langle T_\omega, r\rangle$ is also an accepting run of A'. Conversely, assume that $a^\omega \in \mathcal{L}(A')$. Then there is an accepting run $\langle T_r, r\rangle$ of A'. We define an accepting run $\langle T_\omega, r'\rangle$ of A on $\langle T_\omega, V\rangle$ by inductively defining r'. Along with r', we define a mapping $\tau : T_\omega \to T_r$ such that $r'(x) = r(\tau(x))$ for all $x \in T_\omega$. To start, set $r'(\mathsf{root}(T_\omega)) = q_0$ and $\tau(\mathsf{root}(T_\omega)) = \mathsf{root}(T_r)$. For the induction step, let $x \in T_\omega$ such that $r'(y)$ is not yet defined for the successors y of x. Since $\langle T_r, r\rangle$ is a run of A' and by definition of δ', there is a $P \subseteq Q$ such that (i) $\mathsf{is\_mother}_A(r(\tau(x)), P)$ and (ii) for all $q \in P$, there is a successor y of $\tau(x)$ in T_r with $r(y) = q$. By (i), there is a word $t = q_1 q_2 \cdots \in P^\omega$ that satisfies the counting constraint $\delta(r(\tau(x)), a) = \delta(r'(x), a)$. For all $i \geq 1$, define $r'(x \cdot i) = q_i$ and set $\tau(x \cdot i)$ to some successor y of $\tau(x)$ in T_r such that $r(y) = q_i$ (which exists by (ii)). It is not hard to check that $\langle T_\omega, r'\rangle$ is indeed an accepting run of A on $\langle T_\omega, V\rangle$. □

For a 1APW $A = \langle \{a\}, Q, \delta, q_0, \mathcal{F}\rangle$, $q \in Q$, and $P \subseteq Q$, the function $\mathsf{is\_mother}_A(q, P)$ returns true if P satisfies the Boolean formula $\delta(q, a)$, and false otherwise.

Since the transition function of the automaton A' from Lemma 6.10 is of size exponential in the number of states of the ω-1GNPT A, we should not compute A' explicitly. Indeed, this is not necessary since all we need from A' is access to $\mathcal{F}$ and $\mathsf{is\_mother}_{A'}$ and, as stated in the next lemma, $\mathsf{is\_mother}_{A'}$ coincides with $\mathsf{is\_mother}_A$. The lemma is an immediate consequence of the definition of the 1APW in Lemma 6.10.

Lemma 6.11. *Let A and A' be as in Lemma 6.10, with state set Q. Then* $\mathsf{is\_mother}_A = \mathsf{is\_mother}_{A'}$.

To decide the emptiness of 1APWs, we use the algorithm from [KV98]. It is a recursive procedure that accesses the transition function of the 1APW only via is_mother. If started on a 1APW with n states and index k, it makes at most $2^{\mathcal{O}(k \log n)}$ calls to is_mother and performs at most $2^{\mathcal{O}(k \log n)}$ additional steps. To analyze its runtime requirements, we first determine the complexity of computing is_mother.[1]

Lemma 6.12. *Let $A = \langle \{a\}, b, Q, \delta, q_0, \mathcal{F}\rangle$ be an ω-1GNPT with n states and counting bound b. Then* $\mathsf{is\_mother}_A$ *can be computed in time $b^{\mathcal{O}(\log n)}$.*

Proof. Assume that we want to check whether $\mathsf{is\_mother}_A(q, P)$, for some $q \in Q$ and $P \subseteq Q$. Let $\theta_1, \ldots, \theta_k$ be all formulas occurring in $C := \delta(q, a)$. We construct a deterministic Büchi automaton $A' = \langle \Sigma', Q', q_0', \delta', F'\rangle$ on infinite words that accepts precisely those words $t \in P^\omega$ that satisfy C:

- $\Sigma' = P$;
- $Q' = \{0, \ldots, b\}^k$;
- $q_0' = \{0\}^k$;
- $\delta'((i_1, \ldots, i_k), p)$ is the vector $(j_1, \ldots, j_k)$, where for all $h \in \{1, \ldots, k\}$, we have $j_h = \min\{b, i_h + 1\}$ if $p \in \Sigma'$ satisfies θ_h, and $j_h = i_h$ otherwise;
- F' consists of those tuples $(i_1, \ldots, i_k)$ such that for all $h \in \{1, \ldots, k\}$,
  (1) there is no $\langle \theta_h, (\leq, r)\rangle \in C$ with $r < i_h$;
  (2) for all $\langle \theta_h, (>, r)\rangle \in C$, we have $i_h \geq r$.

[1] We remark that the analogous Lemma 1 of [KSV02] is flawed because it considers only trees of finite outdegree.

By definition of GNPTs, the cardinality of C is bounded by $\log n$. Thus, A' has $b^{\log n}$ states. It remains to note that the emptiness problem for deterministic Büchi word automata (is NLOGSPACE-complete [VW94] and) can be solved in linear time [Var07]. □

Now for the runtime of the algorithm. Let A be a GNPT with n states, counting bound b, and index k. To decide emptiness of A, we convert A into an ω-1GNPT A' with $n+1$ states, counting bound b, and index k, and then into a 1APW A'' with $n+1$ states and index k. By Lemma 6.12, we obtain the following result.

Theorem 6.13. *Let $A = \langle \Sigma, b, Q, \delta, q_0, \mathcal{F} \rangle$ be a GNPT with $|Q| = n$, and index k. Then emptiness of A can be decided in time $(b+2)^{\mathcal{O}(k \cdot \log n)}$.*

6.4. **Wrapping Up.** Finally, we are ready to prove Theorem 4.2, which we restate here for convenience.

Theorem 4.2. *The emptiness problem for a 2GAPT $A = \langle \Sigma, b, Q, \delta, q_0, \mathcal{F} \rangle$ with n states and index k can be solved in time $(b+2)^{\mathcal{O}(n^2 \cdot k^2 \cdot \log k \cdot \log b^2)}$.*

Proof. By Theorem 6.7, we can convert A into a GNPT A' with $2^{\mathcal{O}(kn^2 \cdot \log k \cdot \log b^2)}$ states, index nk, and counting bound b. Thus, Theorem 6.13 yields the desired result. □

A matching EXPTIME lower bound is inherited from nongraded, one-way alternating tree automata.

7. CONCLUSION

We have studied the complexity of μ-calculi enriched with inverse programs, graded modalities, and nominals. Our analysis has resulted in a rather complete picture of the complexity of such logics. In particular, we have shown that only the fully enriched μ-calculus is undecidable, whereas all its fragments obtained by dropping at least one of the enriching features inherit the attractive computational behavior of the original, non-enriched μ-calculus.

From the perspective of the description logic OWL, the picture is as follows. Undecidability of the fully enriched μ-calculus means that OWL extended with fixpoints is undecidable. The decidable μ-calculi identified in this paper give rise to natural fragments of OWL that remain decidable when enriched with fixpoints. Orthogonal to the investigations carried out in this paper, it would be interesting to understand whether there are any second-order features that can be added to OWL without losing decidability. In particular, decidability of OWL extended with transitive closure is still an open problem.

Acknowledgements. We are grateful to Orna Kupferman and Ulrike Sattler for helpful discussions of [SV01, KSV02].

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.

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

[BS06] J. Bradfield and C. Stirling. Modal μ-calculi, *Handbook of Modal Logic* (Blackburn, Wolter, and van Benthem, eds.), pages 722–756, Elsevier, 2006.

[CGL99] 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.

[EJ91] E. A. Emerson and C. S. Jutla. Tree automata, Mu-Calculus and determinacy. In *Proc. of the 32nd Annual Symposium on Foundations of Computer Science* (FOCS'01), IEEE Computer Society Press, pages 368–377, 1991.

[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*, Vol. 133(2), pages 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. Extended version at URL http://www.cs.huji.ac.il/ ornak/publications/cade02.pdf

[KV98] O. Kupferman and M.Y. Vardi. Weak alternating automata and tree automata emptiness. *Proc. of the 30th STOC*, ACM, 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*, Vol. 47(2), pages 312–360, 2000.

[Mos84] A. W. Mostowski. Regular expressions for infinite trees and a standard form of automata. In *Fifth Symposium on Computation Theory*, LNCS 208, pages 157–168, 1984.

[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*, LNAI 2083, pages 76–91. Springer Verlag, 2001.

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

[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.

[Var07] M.Y. Vardi. Automata-Theoretic Model Checking Revisited In *Proc. of the 8th VMCAI*, LNAI 4349, pages 137-150, 2007..

[VW94] M.Y. Vardi and P. Wolper. Reasoning about Infinite Computations. Information and Computation, Vol. 115(1), pages 1–37, 1994.