On CTL* with Graded Path Modalities*

Benjamin Aminof[1], Aniello Murano[2], and Sasha Rubin[2]

[1] Technische Universität Wien, Austria
[2] Università degli Studi di Napoli "Federico II", Italy

Abstract. Graded path modalities count the number of paths satisfying a property, and generalize the existential (E) and universal (A) path modalities of CTL*. The resulting logic is denoted GCTL*, and is a very powerful logic since (as we show) it is equivalent, over trees, to monadic path logic. We settle the complexity of the satisfiability problem of GCTL*, i.e., 2ExpTime-Complete, and the complexity of the model checking problem of GCTL*, i.e., PSpace-Complete. The lower bounds already hold for CTL*, and so we supply the upper bounds. The significance of this work is two-fold: GCTL* is much more expressive than CTL* as it adds to it a form of quantitative reasoning, and this is done at no extra cost in computational complexity.

1 Introduction

Quantitative Verification and Graded Modalities. Temporal logics are the cornerstone of the field of formal verification. In recent years, much attention has been given to extending these by quantitative measures of function and robustness, e.g., [18]. Unfortunately, these extensions often require one to reason about weighted automata for which much is undecidable [1, 2, 10]. One way to extend classical temporal logics at a lower cost is by counting quantifiers, known as graded modalities. Graded world modalities were introduced in formal verification as a useful extension of the standard existential and universal quantifiers in branching-time modal logics [7, 16, 19, 23]. These modalities allow one to express properties such as "there exist at least n successors satisfying a formula" or "all but n successors satisfy a formula". A prominent example is the extension of μ-calculus called $G\mu$-calculus [7, 19].

Despite its high expressive power, the μ-calculus (which extends modal logic by least and greatest fixpoint operators) is a low-level logic, making it "unfriendly" for users, who usually find it very hard to understand, let alone write, formulas involving even very modest nesting of fixed points. In contrast, CTL and CTL* are much more intuitive and user-friendly. An extension of CTL with graded *path modalities* called GCTL was defined in [5, 6]. Although there are

* Benjamin Aminof is supported by the Austrian National Research Network S11403-N23 (RiSE) of the Austrian Science Fund (FWF) and by the Vienna Science and Technology Fund (WWTF) through grant ICT12-059. Aniello Murano is partially supported by the FP7 EU project 600958-SHERPA. Sasha Rubin is a Marie Curie fellow of the Istituto Nazionale di Alta Matematica.

several positive results about GCTL this logic suffers from similar limitations as CTL, i.e., it cannot nest successive temporal operators and so cannot express fairness constraints. This dramatically limits the usefulness of GCTL and so we turn instead to GCTL* in which one can naturally and comprehensibly express complex properties of systems. Although the syntax and semantics of GCTL* were defined and justified in [6], only a rudimentary study of it was made. In particular, the complexity of the satisfiability and model checking problem for this logic was never established, and remained open since its introduction in 2009. Instead, research has focused on the much simpler fragment of GCTL.

Our results. We establish the exact complexity of the satisfiability and model checking problems for GCTL* to be 2ExpTime-Complete and PSpace-Complete, respectively. Thus, in both cases, the problems for GCTL* are not harder than for CTL*. This is very good news indeed since, as we also show, GCTL* is expressively equivalent, over trees, to monadic path logic, and is thus a powerful, yet relatively friendly logic. Along the way, we prove that GCTL* has the bounded-degree tree-model property, i.e., a satisfiable formula is satisfied in a tree whose branching degree is at most exponential in the size of the formula.

The importance of our results. We obtain that GCTL* has the following desirable combination of attributes:

a) GCTL* can naturally express properties of paths as well as count them. For example, the formula $\mathsf{E}^{\geq 2}\mathsf{G}(\mathit{request} \rightarrow (\mathit{request}\ \mathsf{U}\ \mathit{granted}))$ says: "there are at least two ways to schedule the computation such that every request is eventually granted". This cannot be expressed in CTL* nor in GCTL.

The naive semantics for $\mathsf{E}^{\geq n}\psi$ which states that "there are at least n different paths satisfying ψ" while at first glance may seem natural and desirable, when examined more carefully turns out to be undesirable, and less informative. For example, consider a faulty program in which requests are sometimes not granted. In GCTL* (unlike the naive counting) the formula $\mathsf{E}^{\geq 2}[\mathsf{F}(\mathit{request} \wedge \neg \mathsf{F}\mathit{granted})]$ requires at least two *incomparable* sequences of operations, each causing this faulty behaviour. Hence, it indicates whether the faulty behaviour is the result of multiple underlying problems, and is not confused by multiple paths that are extensions of a single faulty prefix. Furthermore, the naive counting very quickly leads to unnatural interpretations, as convincingly argued in [6].

This ability to easily count paths is a natural fit in various application domains. For example, in databases there is a close relationship between model-checking CTL* and XML navigation (see [4]). The logic GCTL* allows one to express quantitative requirements such as "client has at least 5 items in last-month orders". More generally, graded operators are common in description logics, which are prominently used for formal reasoning in AI (e.g., knowledge querying, planning with redundancies).

b) GCTL* is extremely expressive. Not only does GCTL* extend CTL* (and thus, unlike CTL, it can reason about fairness), we prove that it is expressively equivalent, over trees, to Monadic Path Logic (MPL) which is Monadic Second-Order Logic (MSOL) interpreted over trees but with set quantification restricted to branches.

c) GCTL* has relatively low complexity of satisfiability. Unfortunately, the complexity of satisfiability of MPL is non-elementary (this is already true for FOL). In sharp contrast, we prove that the complexity of satisfiability of GCTL* is 2ExpTime, and thus is no harder than for CTL*.

Technical Contributions. The upper bounds are obtained by exploiting an automata-theoretic approach for branching-time logics, combined with game theoretic reasoning at a crucial point. The automata-theoretic approach is suitable because GCTL* turns out to have the tree-model property. It is very hard to see how other techniques for deciding questions in logic (e.g. effective quantifier elimination, tableaux, composition) can be used to achieve optimal complexity results for GCTL*. Our proof is not just an easy adaptation of the classical decision procedure. We relate GCTL* to a new model of automata, i.e., *Graded Hesitant Tree Automata* (GHTA). These automata work on finitely-branching trees (not just k-ary trees) and their transition relations can count up to a given number (usual alternating automata only count up to 1).

Related Work. Counting modalities were first introduced by Fine [16] under the name *graded world modalities*. A systematic treatment of the complexity of various graded modal logics followed [9,14,21,23,24]. The extension of μ-calculus by graded world modalities was investigated in [7, 19]. Although these articles introduce automata that can count, our GHTA are more complicated since they have to deal with graded path modalities and not just graded world modalities. The extension of CTL* by the ability to say "there exist at least n successors satisfying ψ", called counting-CTL*, was defined in [22], and its connection with Monadic Path Logic studied using the composition method. It is unclear if that method, although elegant, can yield the complexity bounds we achieve (even for counting-CTL*). As shown in [6], $G\mu$-calculus cannot succinctly reason about paths, or even grandchildren of a given node (the same goes for counting-CTL*). The first work to deal with graded path modalities is [5] that introduced GCTL, the extension of CTL by these modalities. Graded path modalities over CTL were also studied in [15], using a different semantics than GCTLwhich is tailored for extending CTL, and it is unclear how one can extend their work to CTL*.

2 The GCTL* temporal logic

Let $\mathbb{N}$ denote the positive integers, and $[d] = \{1, 2, \ldots, d\}$ for $d \in \mathbb{N}$. An LTS (Labeled Transition System/Kripke structure) is a tuple $\mathsf{S} = \langle \Sigma, S, E, \lambda \rangle$, where Σ is a set of *labels*, S is a countable set of *states*, $E \subseteq S \times S$ is the *transition relation*, and $\lambda : S \mapsto \Sigma$ is the *labeling function*. Typically, $\Sigma = 2^{\mathrm{AP}}$ where AP is a finite set of *atomic propositions*. The *degree* of a state s is the cardinality of the set $\{t \in S : (s,t) \in E\}$ of its successors. We assume that E is total, i.e., that every state has a successor. A path in S is a finite or infinite sequence $\pi_0\pi_1 \cdots \in (S^*) \cup (S^\omega)$ such that $(\pi_{i-1}, \pi_i) \in E$ for all $1 \leq i < |\pi|$ ($|\pi|$ is the *length* of π). The set of (finite and infinite) paths in S is written $\mathsf{pth}(\mathsf{S})$, and the set of (finite and infinite) paths in S that start in a given state $q \in S$ is written $\mathsf{pth}(\mathsf{S}, q)$. Let $\preceq$ be the prefix ordering on paths. If $\pi \preceq \pi'$ say that π' is an *extension* of

π. For a set of paths X, denote by $\mathsf{min}(X)$ the minimal elements of X according to $\preceq$. A *Σ-labeled tree* T is a pair $\langle T, V\rangle$ where $T \subseteq \mathbb{N}^*$ is a $\prec$-downward closed set of strings over $\mathbb{N}$, and $V : T \to \Sigma$ is a labeling. We implicitly view a tree $\mathsf{T} = \langle T, V\rangle$ as the LTS $\langle \Sigma, T, E, V\rangle$ where $(t, s) \in E$ iff s is a son of t. If every node of a tree T has a finite degree then T is *finitely branching*. If every node has at most degree $k \in \mathbb{N}$, then T is *boundedly branching* or *has branching degree k*.

2.1 Syntax and Semantics of GCTL*

GCTL* extends CTL* by *graded path quantifiers* of the form $\mathsf{E}^{\geq g}$. We follow the definition of GCTL* from [6], but give a slightly simpler syntax. We assume that the reader is familiar with the logics CTL*, LTL, and CTL (see [20, 25]).

The semantics of GCTL* is defined for an LTS S. The GCTL* formula $\mathsf{E}^{\geq g}\psi$, for GCTL* path formula ψ, can be read as *"there exist at least g (minimal ψ-conservative) paths"*. Minimality was defined above, and so we now say, informally, what it means for a path to be ψ-conservative. An infinite path of S is ψ-conservative if it satisfies ψ, and a finite path of S is ψ-conservative if all its (finite and infinite) extensions in S satisfy ψ. Note that this notion uses a semantics of GCTL* over finite paths, and thus the semantics of GCTL* needs to be defined for finite paths (as well as infinite paths). As in [6], we use the weak-version of semantics of temporal operators for finite paths (defined in [11]). Intuitively, temporal operators are interpreted pessimistically (with respect to possible extensions of the path), e.g., $(\mathsf{S}, \pi) \models \mathsf{X}\psi$ iff $|\pi| \geq 2$ and $(\mathsf{S}, \pi_{\geq 1}) \models \psi$.

Syntax of GCTL*. Fix a set of atoms AP. The GCTL* *state (φ) and path (ψ) formulas* are built inductively from AP using the following grammar: $\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{E}^{\geq g}\psi$ and $\psi ::= \varphi \mid \neg\psi \mid \psi \vee \psi \mid \mathsf{X}\psi \mid \psi\mathsf{U}\psi \mid \psi\mathsf{R}\psi$.

In the first part, p varies over AP and g varies over $\mathbb{N}$ (and thus, technically, there are infinitely many rules in this grammar). As usual, X, U and R are called *temporal operators* and $\mathsf{E}^{\geq g}$ (for $g \in \mathbb{N}$) are called *path modalities* (also called *path quantifiers*). We write $\mathsf{F}\varphi$ instead of $\mathbf{true}\mathsf{U}\varphi$, and $\mathsf{G}\varphi$ instead of $\mathbf{false}\mathsf{R}\varphi$. The class of GCTL* *formulas* is the set of state formulas generated by the above grammar. The simpler class of *Graded* CTL *formulas* (GCTL) is obtained by requiring each temporal operator to be immediately preceded by a path quantifier. The logic LTL is the class of path formulas in which no path quantifier appears. The *degree* of the quantifier $\mathsf{E}^{\geq g}$ is the number g. The *degree* $\mathsf{deg}(\varphi)$, of a state formula φ, is the maximum of the degrees of the quantifiers appearing in φ. The *length* $|\varphi|$, of a formula φ, is defined inductively on the structure of φ as usual, and using $|\mathsf{E}^{\geq g}\psi|$ equal to $g + 1 + |\psi|$ (i.e., g is coded in unary).

Semantics of GCTL*. Given an LTS S and a state $s \in \mathsf{S}$, the definition of $(\mathsf{S}, s) \models \varphi$ is done inductively on the structure of φ, exactly as for CTL*, with the only change concerning the new path quantifier $\mathsf{E}^{\geq g}$. For $\varphi = \mathsf{E}^{\geq g}\psi$, where ψ is a GCTL* path formula, let $(\mathsf{S}, s) \models \varphi$ iff the cardinality of the set $\mathsf{min}(Con(\mathsf{S}, s, \psi))$ is at least g, where $Con(\mathsf{S}, s, \psi) := \{\pi \in \mathsf{pth}(\mathsf{S}, s) \mid \forall \pi' \in \mathsf{pth}(\mathsf{S}, s) : \pi \preceq \pi'$ implies $(\mathsf{S}, \pi') \models \psi\}$. The paths in $Con(\mathsf{S}, s, \psi)$ are called *ψ-conservative (in S starting at s)*, and paths in $\mathsf{min}(Con(\mathsf{S}, s, \psi))$ are called *minimal ψ-conservative*. It is not hard to see that, for total LTSs, the classic

logic CTL*coincides with the fragment of GCTL* in which the degree g of all quantifiers $\mathsf{E}^{\geq g}$ is 1.

If ψ is an LTL formula, we may write $\pi \models \psi$ instead of $(\mathsf{S}, \pi) \models \psi$. This is justifiable since the truth of ψ depends only on the path π independently of the rest of S. Two state formulas ϕ, ϕ' are *equivalent* if for all S and $s \in S$, we have $(\mathsf{S}, s) \models \phi$ iff $(\mathsf{S}, s) \models \phi'$. Two path formulas ψ, ψ' are *equivalent* if for all S and $\pi \in \mathsf{pth}(\mathsf{S})$, we have that $(\mathsf{S}, \pi) \models \psi$ if and only if $(\mathsf{S}, \pi) \models \psi'$. An LTS S with a designated state $q \in S$ is a *model* of a GCTL* formula φ, sometimes denoted $\mathsf{S} \models \varphi$, if $(\mathsf{S}, q) \models \varphi$. For a labeled tree T, the designated node is by default the root, and thus, $\mathsf{T} \models \varphi$ means that $(\mathsf{T}, \epsilon) \models \varphi$ (recall that ϵ designates the root of T). A GCTL* formula φ is *satisfiable* iff it has a model.

Example 1. We unpack the meaning of the GCTL* formula from the introduction $\mathsf{E}^{\geq 2}[\mathsf{F}(\mathit{request} \wedge \neg \mathsf{F}\mathit{granted})]$. Let ψ denote the path formula $\mathsf{F}(\mathit{request} \wedge \neg \mathsf{F}\mathit{granted})$. First, a finite or infinite path π satisfies ψ if at some point t the atom *request* holds, and at no later point on π does the atom *granted* hold. A finite π is ψ-conservative if and only if it satisfies ψ and the atom *granted* does not hold in any node of the subtree rooted at the end of π; and an infinite path is ψ-conservative if and only if it satisfies ψ. Thus, $\mathsf{E}^{\geq 2}\psi$ holds if and only if there exist two possibly finite paths, say π^1 and π^2, neither one a prefix of the other, both satisfying ψ (i.e., π^i has a request that is never granted on π^i), and such that if π^i is finite then that path has a request that is not granted in any possible extension of π^i.

2.2 Important Properties of GCTL*

Like CTL* (see [20]), one can think of a GCTL* path formula ψ over atoms AP as an LTL formula Ψ over atoms which themselves are GCTL* state formulas, as follows. A formula φ is a *state sub-formula* of ψ if i) φ is a state formula, and ii) φ is a sub-formula of ψ. A formula φ is a *maximal state sub-formula* of ψ if φ is a state sub-formula of ψ, and φ is not a proper sub-formula of any other state sub-formula of ψ. Let $max(\psi) = \{\varphi \mid \varphi \text{ is a maximal state sub-formula of } \psi\}$, and let $\overline{max(\psi)} = \bigcup_{\varphi \in max(\psi)} \{\varphi, \neg\varphi\}$ be the set of all maximal state sub-formulas of ψ and their negations. Every GCTL* path formula ψ can be viewed as the formula Ψ whose atoms are elements of $max(\psi)$. Note that Ψ is an LTL formula. For example, for $\psi = ((\mathsf{X}p)\ \mathsf{U}\ (\mathsf{E}^{\geq 2}\mathsf{X}q)) \vee p$, the state sub-formulas are $\{p, q, \mathsf{E}^{\geq 2}\mathsf{X}q\}$, and $max(\psi) = \{p, \mathsf{E}^{\geq 2}\mathsf{X}q\}$, and thus Ψ is the LTL formula $(\mathsf{X}\underline{p}\ \mathsf{U}\ \underline{\mathsf{E}^{\geq 2}\mathsf{X}q}) \vee \underline{p}$ over the atoms $\{p, \mathsf{E}^{\geq 2}\mathsf{X}q\}$ (here we underline sub-formulas that are treated as atoms). Given an LTS $\mathsf{S} = \langle 2^{\mathrm{AP}}, S, E, \lambda \rangle$ and a GCTL* path formula ψ, we define the *relabeling* of the LTS S by the values of the formulas in $max(\psi)$ as $\mathsf{S}_\psi = \langle max(\psi), S, E, L \rangle$ where $L(s)$ is the union of $\lambda(s)$ and the set of $\varphi \in max(\psi)$ such that $(\mathsf{S}, s) \models \varphi$.

Lemma 1. *For every* GCTL* *path formula* ψ *over* AP *there is an* LTL *formula* Ψ *over* $max(\psi)$ *such that for all* S *and all paths* π *in* S*:* $(\mathsf{S}, \pi) \models \psi$ *iff* $(\mathsf{S}_\psi, \pi) \models \Psi$.

It is not hard to see that GCTL* is not invariant under bisimulation (cf. [6]), and that it is invariant under unwinding (cf. [6]). The next theorem shows that GCTL* is a powerful logic. Indeed, it is equivalent, over trees, to Monadic Path Logic (MPL) which is MSO with quantification restricted to branches. Note that MPL is only defined over trees, while GCTL* (like CTL*) is defined over arbitrary LTS. This is the reason we compare their expressiveness over trees.

Theorem 1. GCTL* *is equivalent, over trees, to Monadic Path Logic.*

3 Graded Hesitant Tree Automata

In this section we define a new kind of automaton called Graded Hesitant Tree Automata. We also make use of the classical non-deterministic finite word automata (NFW) and non-deterministic Büchi word automata (NBW) (see [25]), alternating parity tree automata (APTA) (see [12]), and alternating hesitant tree automata (AHTA) (see [20]). We write $\langle \Sigma, Q, q_0, \delta, G\rangle$ for NBWs and $\langle \Sigma, Q, q_0, \delta, F\rangle$ for NFWs where Σ is the input alphabet, Q is the set of states, q_0 is the initial state, $\delta \subseteq Q \times \Sigma \times Q$ is the transition relation, $G \subseteq Q$ is the set of accepting states and $F \subseteq Q$ the set of final states. For a set X, let $\mathsf{B}^+(X)$ be the set of positive Boolean formulas over X, including the constants **true** and **false**. A set $Y \subseteq X$ satisfies a formula $\theta \in \mathsf{B}^+(X)$, written $Y \models \theta$, if assigning **true** to elements in Y and **false** to elements in $X \setminus Y$ makes θ true. Graded hesitant tree automata (GHTA) generalise AHTA[3]: a) they can work on finitely-branching trees (not just k-ary branching trees), and b) their transition relation allows the automaton to send multiple copies into the successors of the current node in a much more flexible way. Below we formally define AHTA and GHTA.

Definition of AHTA An *Alternating Hesitant Tree Automaton* (AHTA) is a tuple $\mathsf{A} = \langle \Sigma, D, Q, q_0, \delta, \langle G, B\rangle, \langle \mathsf{part}, \mathsf{type}, \preceq\rangle\rangle$ where Σ is a non-empty finite set of *input letters*; $D \subset \mathbb{N}$ is a finite non-empty set of *directions*, Q is the non-empty finite set of *states*, $q_0 \in Q$ is the *initial state*; the pair $\langle G, B\rangle \in 2^Q \times 2^Q$ is the *acceptance condition*[4] (we sometimes call the states in G *good states* and the states in B *bad states*); $\delta : Q \times \Sigma \to \mathsf{B}^+(D \times Q)$ is the *alternating transition function*; $\mathsf{part} \subset 2^Q$ is a partition of Q, $\mathsf{type} : \mathsf{part} \to \{trans, exist, univ\}$ is a function assigning the label *transient*, *existential* or *universal* to each element

[3] Strictly speaking, GHTA generalise the symmetric variant of AHTA. That is, for every language accepted by an AHTA and that is closed under the operation of permuting siblings, there is a GHTA that accepts the same language.

[4] The combination of a Büchi and a co-Büchi condition that hesitant automata use can be thought of as a special case of the parity condition with 3 colors. Thus, we could have defined Graded Parity Tree Automata instead (using the parity condition, our automata strictly generalise the ones in [5, 19]) However, we do not need the full power of the parity condition, and in order to achieve optimal complexity for model checking of GCTL* we need to be able to decide membership of our automata in a space efficient way, which cannot be done with the parity acceptance condition.

of the partition, and $\preceq \subset 2^Q \times 2^Q$ is a partial order on part. Moreover, the transition function δ is required to satisfy the following *hesitancy condition*: for every $\mathbb{Q} \in \mathsf{part}$, every $q \in \mathbb{Q}$, and every $\sigma \in \Sigma$: (i) for every $\mathbb{Q}' \in \mathsf{part}$ and $q' \in \mathbb{Q}'$, if q' occurs in $\delta(q, \sigma)$ then $\mathbb{Q}' \preceq \mathbb{Q}$; (ii) if $\mathsf{type}(\mathbb{Q}) \in \mathit{trans}$ then no state of $\mathbb{Q}$ occurs in the formula $\delta(q, \sigma)$; (iii) if $\mathsf{type}(\mathbb{Q}) \in \mathit{exist}$ (resp., $\mathsf{type}(\mathbb{Q}) \in \mathit{univ}$) then there is at most one element of $\mathbb{Q}$ in each disjunct of the DNF (resp., conjunct of CNF) of $\delta(q, \sigma)$.

An *input tree (for AHTA)* is a Σ-labeled tree $\mathsf{T} = \langle T, V \rangle$ with $T \subseteq D^*$. Since D is finite, such trees have fixed finite branching degree. A *run (or run tree)* of an alternating tree automaton A on input tree $\mathsf{T} = \langle T, V \rangle$ is a $(T \times Q)$-labeled tree $\langle T_r, r \rangle$, such that *(a)* $r(\varepsilon) = (\varepsilon, q_0)$ and *(b)* for all $y \in T_r$, with $r(y) = (x, q)$, there exists a *minimal* set $\mathsf{S} \subseteq D \times Q$, such that $\mathsf{S} \models \delta(q, V(x))$, and for every $(d, q') \in \mathsf{S}$, it is the case that $x \cdot d$ is a son of x, and there exists a son y' of y, such that $r(y') = (x \cdot d, q')$.

Note that if $\delta(q, V(x)) = \mathbf{true}$ then $S = \emptyset$ and the node y has no children; and if there is no S as required (for example if x does not have the required sons) then there is no run-tree with $r(y) = (x, q)$. Observe that disjunctions in the transition relation are resolved into different run trees, while conjunctions give rise to different sons of a node in a run tree. If v is a node of the run tree, and $r(v) = (u, q)$, call u the *location associated with* v, denoted $\mathit{loc}(v)$, and call q the *state associated with* v, denoted $\mathit{state}(v)$.

We now discuss the acceptance condition. Fix a run tree $\langle T_r, r \rangle$ and an infinite path π in it. Say that the path *visits* a state q at time i if $\mathit{state}(\pi_i) = q$. The hesitancy restriction (i) guarantees that the path π eventually gets trapped and visits only states in some element of the partition, i.e., there exists $\mathbb{Q} \in \mathsf{part}$ such that from a certain time i on, $\mathit{state}(\pi_j) \in \mathbb{Q}$ for all $j \geq i$. The condition (ii) ensures that this set is either existential or universal, i.e., $\mathsf{type}(\mathbb{Q}) \in \{\mathit{exist}, \mathit{univ}\}$. Thus, we say that the path π *gets trapped in an existential set* if $\mathsf{type}(\mathbb{Q}) = \mathit{exist}$, and otherwise we say that it *gets trapped in a universal set*. We can now define what it means for a path in a run tree to be *accepting*. A path that gets trapped in an existential set is *accepting* iff it visits some state of G infinitely often, and a path that gets trapped in a universal set is *accepting* iff it visits every state of B finitely often. A run $\langle T_r, r \rangle$ of an AHTA is *accepting* iff all its infinite paths are accepting. An automaton A accepts an input tree $\langle T, V \rangle$ iff there is an accepting run of A on $\langle T, V \rangle$. The *language* of A, denoted $\mathcal{L}(\mathsf{A})$, is the set of Σ-labeled D-trees accepted by A. We say that A is nonempty iff $\mathcal{L}(\mathsf{A}) \neq \emptyset$.

The *membership problem* of AHTA is the following decision problem: given an AHTA A with direction set D, and a finite LTS S in which the degree of each node is at most $|D|$, decide whether or not A accepts S. The *depth* of the AHTA is the size of the longest chain in $\prec$. The *size* $||\delta||$ of the transition function is the sum of the lengths of the formulas it contains. The *size* $||\mathsf{A}||$ of the AHTA is $|D| + |Q| + ||\delta||$. The partition, partial order and type function are not counted in the size of the automaton. The following is implicit in [20]:

Theorem 2. *The membership problem for AHTA can be solved in* $O(\partial \log^2(|S| \cdot ||\mathsf{A}||))$ *space where* ∂ *is the depth of* A *and* S *is the state set of* S.

Definition of GHTA We now introduce *Graded Hesitant Tree Automata* (GHTA). These can run on finitely-branching trees (not just trees of a fixed finite degree), and the transition function is graded, i.e., instead of a Boolean combination of direction-state pairs, it specifies a Boolean combination of distribution operations. There are two distribution operations: $\Diamond(q_1,...,q_k)$ and its dual $\Box(q_1,...,q_k)$. Intuitively, $\Diamond(q_1,...,q_k)$ specifies that the automaton picks k *different* sons $s_1,...,s_k$ of the current node and, for each $i \leq k$, sends a copy in state q_i to son s_i. Note that the states $q_1,...,q_k$ are not necessarily all different.

A GHTA A is a tuple $\langle \Sigma, Q, q_0, \delta, \langle G, B\rangle, \langle \mathsf{part}, \mathsf{type}, \preceq\rangle\rangle$ where all elements but δ are defined as for AHTA, and $\delta : Q \times \Sigma \to \mathsf{B}^+(\Diamond_Q \cup \Box_Q)$ is a transition function that maps a state and an input letter to a positive Boolean combination of elements in $\Diamond_Q = \{\Diamond(q_1,...,q_k) \mid (q_1,...,q_k) \in Q^k, k \in \mathbb{N}\}$ and $\Box_Q = \{\Box(q_1,...,q_k) \mid (q_1,...,q_k) \in Q^k, k \in \mathbb{N}\}$.

We show how to define the run of a GHTA A on a Σ-labeled finitely-branching tree $\mathsf{T} = \langle T, V\rangle$ by (locally) unfolding every $\Diamond_Q$ and $\Box_Q$ in $\delta(q, V(t))$ into a formula in $\mathsf{B}^+([d] \times Q)$ where d is the branching-degree of node t. For $k, d \in \mathbb{N}$, let $S(k,d)$ be the set of all ordered different k elements in $[d]$, i.e., $(s_1,...,s_k) \in S(k,d)$ iff for every $i \in [k]$ we have that $s_i \in [d]$, and that if $i \neq j$ then $s_i \neq s_j$. Observe that if $k > d$ then $S(k,d) = \emptyset$. For every $d \in \mathbb{N}$, define the function $expand_d : \mathsf{B}^+(\Diamond_Q \cup \Box_Q) \to \mathsf{B}^+([d] \times Q)$ that maps formula ϕ to the formula formed from ϕ by replacing every occurrence of a sub-formula of the form $\Diamond(q_1,...,q_k)$ by the formula $\bigvee_{(s_1,..,s_k)\in S(k,d)}(\bigwedge(s_i, q_i))$, and every occurrence of a sub-formula of the form $\Box(q_1,...,q_k)$ by the formula $\bigwedge_{(s_1,..,s_k)\in S(k,d)}(\bigvee(s_i, q_i))$. Observe that if $k > d$ then $\Diamond(q_1,...,q_k)$ becomes the constant formula **false**, and $\Box(q_1,...,q_k)$ becomes the constant formula **true**. The *run of a GHTA* A is defined as for an alternating tree automaton, except that one uses $expand_n(\delta(q, V(x)))$ instead of $\delta(q, V(x))$ for nodes x of T of degree n. Finally, the *hesitancy condition* defined above for AHTA is required to apply to the expanded transition function, i.e., insert the phrase "every $n \in \mathbb{N}$," before the phrase "and every $\sigma \in \Sigma$", and in items (i)-(iii) replace $\delta(q, \sigma)$ by $expand_n(\delta(q, \sigma))$. Acceptance is as for AHTA.

Lemma 2. *The emptiness problem for GHTA* A *over trees of branching degree at most* d *is decidable in time* $2^{O(d\cdot|Q|^3)}$*, where* Q *is the state set of* A*.*

Proof. Given a GHTA A with state set Q, convert it into an AHTA A' with the same state space by using the function $expand_d$ defined above to transform its transition relation into a non-graded one. This is possible since we assumed a bound d on the branching degree of the input trees, and thus the transformation $expand_d$ can be used in advance. This construction takes time that is $2^{O(|Q|\log d)}$. Recall that AHTA are a special case of alternating parity tree automata (APTA) with 3 priorities. Now apply the fact that the emptiness problem for APTA with p priorities over d-ary trees can be solved in time $2^{O(d\cdot|Q|^p)}$ [12].

4 From GCTL* to Graded Hesitant Automata

Elegant and optimal algorithms for solving the satisfiability and model-checking problems of CTL* were given using the automata-theoretic approach for branching-

time temporal logics [20]. Using this approach, one reduces satisfiability to the non-emptiness problem of a suitable tree automaton accepting all tree-models of a given temporal logic formula. We follow the same approach here, by reducing the satisfiability problem of GCTL* to the non-emptiness problem of GHTA. By Theorem 1, a GCTL* formula is satisfiable (in some, possibly infinite, labeled transition system) iff it has a finitely branching (though possibly unboundedly branching) tree model, which exactly falls within the abilities of GHTA. Our main technical result states that every GCTL* formula can be compiled into an exponentially larger GHTA (the rest of this section provides the proof):

Theorem 3. *Given a* GCTL* *formula* ϑ*, one can build a GHTA* A_ϑ *that accepts all the finitely-branching tree-models of* ϑ*. Moreover,* A_ϑ *has* $2^{O(|\vartheta|\cdot \mathsf{deg}(\vartheta))}$ *states, depth* $O(|\vartheta|)$*, and transition function of size* $2^{O(|\vartheta|\cdot \mathsf{deg}(\vartheta))}$*.*

An important observation that allows us to achieve an optimal construction is the following. Suppose that the formula $\mathsf{E}^{\geq g}\psi$ holds at some node w of a tree. Then, by definition, there are at least g different paths $\rho'^1,...,\rho'^g \in \mathsf{min}(Con(\mathsf{S}, w, \psi))$. Look at any g infinite extensions $\rho^1,...,\rho^g$ of these paths in the tree, and note that by the definition of ψ-conservativeness all these extensions must satisfy ψ. Also observe that for every $i \neq j$, the fact that ρ'^i, ρ'^j are different and minimal implies that the longest common prefix ρ'^{ij} of ρ^i and ρ^j is not ψ-conservative. As it turns out, the other direction is also true, i.e., if there are g infinite paths $\rho^1,...,\rho^g$ satisfying ψ, such that for every $i \neq j$ the common prefix ρ'^{ij} is not ψ-conservative, then there are g prefixes $\rho'^1,...,\rho'^g$ of $\rho^1,...,\rho^g$ respectively, such that $\rho'^1,...,\rho'^g \in \mathsf{min}(Con(\mathsf{S}, w, \psi))$. Note that this allows us to reason about the cardinality of the set $\mathsf{min}(Con(\mathsf{S}, w, \psi))$, by considering only the infinite paths $\rho^1,...,\rho^g$ and their common prefixes, without actually looking at the minimal ψ-conservative paths $\rho'^1,...,\rho'^g$. In reality, we do not even have to directly consider the common prefixes ρ'^{ij}. Indeed, since the property of being ψ-conservative is upward closed (with respect to the prefix ordering $\preceq$ of paths), showing that ρ'^{ij} is not ψ-conservative can be done by finding any extension of ρ'^{ij} that is not ψ-conservative. The following proposition formally captures this.

Proposition 1 *Given a* GCTL* *path formula* ψ *and a* 2^{AP}*-labeled tree* $\mathsf{T} = (T, V)$*, then* $\mathsf{T} \models \mathsf{E}^{\geq g}\psi$ *iff there are* g *distinct nodes* $y_1,...,y_g \in \mathsf{T}$ *(called* breakpoints*) such that for every* $1 \leq i, j \leq g$ *we have: (i) if* $i \neq j$ *then* y_i *is not a descendant of* y_j*; (ii) the path from the root to the father* x_i *of* y_i *is not* ψ*-conservative; (iii) there is an infinite path* ρ^i *in* T*, starting at the root and going through* y_i*, such that* $\rho^i \models \psi$*.*

We are in a position to describe our construction of a GHTA accepting all finitely-branching tree-models of a given GCTL* formula. Naturally, the main difficulty lies in handling the graded modalities. The basic intuition behind the way our construction handles formulas of the form $\varphi = \mathsf{E}^{\geq g}\psi$ is the following. Given an input tree, the automaton A_φ for this formula has to find at least g minimal ψ-conservative paths. At its core, A_φ runs g pairs of copies of itself in parallel. The reason these copies are not run independently is to ensure that

the two members of each pair are kept coordinated, and that different pairs do not end up making the same guesses (and thus overcounting the number of minimal ψ-conservative paths). The task of each of the g pairs is to detect some minimal ψ-conservative path that contributes 1 to the count towards g. This is done indirectly by using the characterization given by Proposition 1. Since this proposition requires checking if certain paths satisfy ψ, the automaton A_φ will access certain classic NBWs. We begin by establishing the existence of these:

Theorem 4. *Given an* LTL *formula ζ, there is an NBW $\mathbb{A}_\zeta$ (resp. NFW $\mathbb{B}_\zeta$), both of size $2^{O(\zeta)}$, accepting exactly all infinite (resp. finite) words that satisfy ζ.*

Lemma 3. *Given an* LTL *formula ζ, there is an NBW $\mathbb{A}^\zeta$ (of size $2^{O(\psi)}$) such that $\mathbb{A}^\zeta$ accepts a word w iff $w \models \zeta$, or $u \models \zeta$ for a prefix u of w. Moreover, $\mathbb{A}^\zeta$ has an accepting sink $\top$, such that if $r_0, r_1, \ldots$ is an accepting run of $\mathbb{A}^\zeta$ on w, and $i \geq 0$ satisfies $r_i \neq \top$, then a (finite or infinite) prefix u of w, of length $|u| > i$, satisfies ζ, and vice-versa (i.e., if a prefix u of w satisfies ζ, then there is an accepting run on w with $r_i \neq \top$ for all $i < |u|$).*

We can now finish the intuitive description of the construction of the automaton A_φ associated with a formula $\varphi = \mathsf{E}^{\geq g}\psi$. Let Ψ be the LTL formula resulting from applying Lemma 1 to ψ. In essence, A_φ guesses the g descendants $y_1, \ldots, y_g$ of the root of the input tree as given in Proposition 1. For every $1 \leq i \leq g$, the automaton uses one copy of $\mathbb{A}^{\neg\Psi}$ to verify that the path π, from the root to the father of y_i, is not ψ-conservative (by guessing some finite or infinite extension $\pi \preccurlyeq \pi'$ of it such that $\pi' \models \neg\Psi$), and one copy of $\mathbb{A}_\Psi$ to guess an infinite path π'' from the root through y_i such that $\pi'' \models \Psi$ (and is thus ψ-conservative).

4.1 The construction of GHTA A_ϑ for a GCTL* formula ϑ.

We induct on the structure of ϑ. Given a state sub-formula ϕ of ϑ (possibly including ϑ), for every formula $\theta \in \overline{max(\phi)}$, let $\mathsf{A}_\theta = \langle \Sigma, Q^\theta, q_0^\theta, \delta^\theta, \langle G^\theta, B^\theta \rangle, \langle \mathsf{part}^\theta, type^\theta, \preceq^\theta \rangle \rangle$ be a GHTA accepting the finitely-branching tree-models of θ. The proof of correctness plus the definition of the hesitancy structure, i.e., of $\langle \mathsf{part}^\theta, type^\theta, \preceq^\theta \rangle$, is in the full version (recall that the hesitancy structure is only used to decide in a space-efficient way membership, which is needed for our result that model-checking of GCTL* is in PSPACE). We build the GHTA A_ϕ accepting all finitely-branching tree-models of ϕ by suitably composing the automata of its maximal sub-formulas and their negations. Note that when composing these automata, we assume w.l.o.g. that the states of any occurrence of a constituent automaton of a sub-formula are disjoint from the states of any other occurrence of a constituent automaton (of the same or of a different sub-formula), as well as from any newly introduced states.[5] Formally:

[5] For example, when building an automaton for $\phi = \varphi_0 \vee \varphi_1$, in the degenerate case that $\varphi_0 = \varphi_1$ then A_{φ_1} is taken to be a copy of A_{φ_0} with its states renamed to be disjoint from those of A_{φ_0}. Also, the new state q_0 may be renamed to avoid a collision with any of the other states.

1. If $\phi = p \in AP$, then $\mathsf{A}_\phi = \langle \Sigma, \{q\}, q, \delta, \langle \emptyset, \emptyset \rangle, \langle \mathsf{part}, \mathsf{type}, \preceq \rangle \rangle$ where $\delta(q, \sigma) =$ **true** if $p \in \sigma$ and **false** otherwise.
2. If $\phi = \varphi_0 \vee \varphi_1$ then A_ϕ is obtained by nondeterministically invoking either A_{φ_0} or A_{φ_1}. Thus, $\mathsf{A}_\phi = \langle \Sigma, \bigcup_{i=0,1} Q^{\varphi_i} \cup \{q_0\}, q_0, \delta, \langle \bigcup_{i=0,1} G^{\varphi_i}, \bigcup_{i=0,1} B^{\varphi_i} \rangle, \beta \rangle$, where $\beta = \langle \mathsf{part}, \mathsf{type}, \preceq \rangle$, and for every $i \in \{0,1\}$, every $\sigma \in \Sigma$, and every $q \in Q^{\varphi_i}$ we have that: $\delta(q, \sigma) = \delta^{\varphi_i}(q, \sigma)$, and $\delta(q_0, \sigma) = \delta^{\varphi_0}(q_0^{\varphi_0}, \sigma) \vee \delta^{\varphi_1}(q_0^{\varphi_1}, \sigma)$.
3. If $\phi = \neg\varphi$, then A_ϕ is obtained by dualizing the automaton A_φ. Formally, the *dual of a GHTA* A is the GHTA obtained by dualizing the transition function of A (i.e., switch $\vee$ and $\wedge$, switch $\top$ and $\bot$, and switch $\Box$ and $\Diamond$), replacing the acceptance condition $\langle G, B \rangle$ with $\langle B, G \rangle$ (and toggling types).

Finally we deal with the case that $\phi = \mathsf{E}^{\geq g}\psi$. Observe that ψ is a path formula and, by Lemma 1, reasoning about ψ can be reduced to reasoning about the LTL formula Ψ whose atoms are elements of $max(\psi)$. Let $\Sigma' = 2^{max(\psi)}$. By Theorem 4, there is an NBW $\mathbb{A}_\Psi = \langle \Sigma', Q^+, q_0^+, \delta^+, G^+ \rangle$ accepting all infinite words in Σ'^ω satisfying Ψ. By Lemma 3, there is an NBW $\mathbb{A}^{\neg\Psi} = \langle \Sigma', Q^\neg, q_0^\neg, \delta^\neg, G^\neg \rangle$ accepting all infinite words in Σ'^ω that either satisfy $\neg\Psi$ or have a prefix that does. Note that the states of these automata are denoted Q^+ and $Q^\neg$. We let A_ϕ be $\langle \Sigma, Q, q_0, \delta, \langle G, B \rangle, \langle \mathsf{part}, \mathsf{type}, \preceq \rangle \rangle$, whose structure we now define.

The set of states. $Q = Q_1 \cup Q_2$, where $Q_1 = (Q^+ \cup \{\bot\})^g \times (Q^\neg \cup \{\bot\})^g \setminus \{\bot\}^{2g}$, and $Q_2 = \bigcup_{\theta \in \overline{max(\psi)}} Q^\theta$. The Q_1 states are used to run g copies of $\mathbb{A}^{\neg\Psi}$ and g copies of $\mathbb{A}_\Psi$ in parallel. Every state in Q_1 is a vector of $2g$ coordinates where coordinates $1,...,g$ (called Ψ *coordinates*) contain states of $\mathbb{A}_\Psi$, and coordinates $g+1,...,2g$ (called $\neg\Psi$ *coordinates*) contain states of $\mathbb{A}^{\neg\Psi}$. In addition, each coordinate may contain the special symbol $\bot$ indicating that it is *disabled*, as opposed to *active*. We disallow the vector $\{\bot\}^{2g}$ with all coordinates disabled. States in Q_2 are all those from the automata A_θ for every maximal state subformula of ψ, or its negation. These are used to run A_θ whenever A_ϕ guesses that θ holds at a node. Also, for every $1 \leq i \leq g$, we denote by $Q^i_{single} = \{(q_1,...,q_{2g}) \in Q_1 \mid q_i \neq \bot,$ and for all $j \leq g$, if $j \neq i$ then $q_j = \bot\}$ the set of all states in Q_1 in which the only active Ψ coordinate is i.

The initial state. $q_0 = (q_1,...,q_{2g})$ where for every $1 \leq i \leq g$ we have that $q_i = q_0^+$ and for every $g+1 \leq i \leq 2g$ we have that $q_i = q_0^\neg$.

The acceptance condition. $B = \cup_{\theta \in \overline{max(\psi)}} B^\theta$ and $G = G' \cup G'' \cup (\cup_{\theta \in \overline{max(\psi)}} G^\theta)$, where $G' = \{(q_1,...,q_{2g}) \in Q^i_{single} \mid q_i \in G^+\}$ is the set of all states in Q_1 in which the only active Ψ coordinate contains a good state, and $G'' = \{(q_1, \ldots, q_{2g}) \in Q_1 \mid \forall i. 1 \leq i \leq g \rightarrow q_i = \bot,$ and $\exists j. g+1 \leq j \leq 2g \wedge q_j \in G^\neg\}$ is the set of all states in Q_1 in which all the Ψ coordinates are inactive, and some $\neg\Psi$ coordinate contains a good state.

The transition function. δ is defined, for every $\sigma \in \Sigma$, as follows:

- For every $q \in Q_2$, let $\theta \in \overline{max(\psi)}$ be such that $q \in Q^\theta$, and define $\delta(q, \sigma) = \delta^\theta(q, \sigma)$. I.e., for states in Q_2, follow the rules of their respective automata.
- For every $q \in Q_1$, we define $\delta(q, \sigma) := \bigvee_{\sigma' \in \Sigma'} (J \wedge K \wedge L)$ where $J = \bigvee_{X \in Legal(q, \sigma')} \Diamond(X)$, $K = \bigwedge_{\theta \in \sigma'} \delta^\theta(q_0^\theta, \sigma)$, $L = \bigwedge_{\theta \notin \sigma'} \delta^{\neg\theta}(q_0^{\neg\theta}, \sigma)$, where $Legal(q, \sigma')$ is the set of all *legal distributions* of (q, σ'), and is defined later.

Informally, the disjunction $\bigvee_{\sigma' \in \Sigma'}$ corresponds to all possible guesses of the set of maximal subformulas of ψ that currently hold. Once a guess σ' is made, the copies of $\mathbb{A}^{\neg\Psi}$ and $\mathbb{A}_\Psi$ simulated by the states appearing in $Legal(q, \sigma')$ proceed as if the input node was labeled by the letter σ'. The conjunction $(\wedge_{\theta \in \sigma'} \delta^\theta(q_0^\theta, \sigma)) \wedge (\wedge_{\theta \notin \sigma'} \delta^{\neg\theta}(q_0^{\neg\theta}, \sigma))$ ensures that a guess is correct by launching a copy of A_θ for every subformula $\theta \in \sigma'$ that was guessed to hold, and a copy of $\mathsf{A}_{\neg\theta}$ for every subformula θ guessed not to hold.

We define *legal distribution*. Intuitively, a legal distribution of (q, σ') is a sequence $q^1,...,q^m$ of different states from Q_1 that "distribute" among them, without duplication, the coordinates active in q, while making sure that for every $1 \leq i \leq g$ coordinate i (which simulates a copy of $\mathbb{A}_\Psi$) does not get separated from the coordinate $i + g$ (which simulates its partner copy of $\mathbb{A}^{\neg\Psi}$) for as long as i is not the only active Ψ coordinate. As expected, every active coordinate j, in any of the states $q^1,...,q^m$, follows from q_j by using the transitions available in the automaton it simulates: $\mathbb{A}_\Psi$ if $j \leq g$, or $\mathbb{A}^{\neg\Psi}$ if $j > g$.

More formally, given a letter $\sigma' \in \Sigma'$, and a state $q = (q_1,...q_{2g}) \in Q_1$ in which the active coordinates are $\{i_1,...,i_k\}$, we say that a sequence $X = q^1,...,q^m$ (for some $m \geq 1$) of distinct states in Q_1 is a *legal distribution* of (q, σ') if the following conditions hold: *(i)* the coordinates active in the states $q^1,...,q^m$ are exactly $i_1,...,i_k$, i.e., $\{i_1,...,i_k\} = \cup\{i \in \{1,...,2g\} \mid \exists 1 \leq l \leq m \text{ s.t. } q_i^l \neq \bot\}$. *(ii)* if a coordinate i_j is active in some $q' \in X$ then it is not active in any other $q'' \in X$; *(iii)* if $1 \leq i_j < i_l \leq g$ are two active Ψ coordinates in some $q' \in X$, then $q'_{i_j+g}, q'_{i_l+g} \in Q^\neg \setminus \{\top\}$, i.e., the coordinates $i_j + g, i_l + g$ are also active in q' and do not contain the accepting sink of $\mathbb{A}^{\neg\Psi}$; *(iv)* if i_j is active in some $q' \in X$ then $(q_{i_j}, \sigma', q'_{i_j}) \in \delta^+$ if $i_j \leq g$, and $(q_{i_j}, \sigma', q'_{i_j}) \in \delta^\neg$ if $i_j > g$. I.e., active Ψ coordinates evolve according to the transitions of $\mathbb{A}_\Psi$, and active $\neg\Psi$ coordinates according to the those of $\mathbb{A}^{\neg\Psi}$.

Remark 1. We make two observations. First, the $2g$ copies of $\mathbb{A}^{\neg\Psi}$ and $\mathbb{A}_\Psi$ can not simply be launched from the root of the tree using a conjunction in the transition relation. The reason is that if this is done then there is no way to enforce property (i) of Proposition 1. Second, a cursory look may suggest that different copies of $\mathbb{A}^{\neg\Psi}$ and $\mathbb{A}_\Psi$ that are active in the current vector may be merged. Unfortunately, this cannot be done since $\mathbb{A}^{\neg\Psi}$ and $\mathbb{A}_\Psi$ are nondeterministic, and thus, different copies of these automata must be able to make independent guesses in the present in order to accept different paths in the future.

Proposition 2 *The automaton A_ϑ is a GHTA with depth $O(|\vartheta|)$ and $2^{O(|\vartheta| \cdot \mathsf{deg}(\vartheta))}$ many states, and the size of its transition function is $2^{O(|\vartheta| \cdot \mathsf{deg}(\vartheta))}$.*

5 Complexity of Satisfiability and MC of GCTL*

Theorem 5. *A satisfiable* GCTL* *formula ϑ has a tree model of branching degree at most $2^{O(|\vartheta| \cdot \mathsf{deg}(\vartheta))}$.*

Proof. Suppose ϑ is satisfiable. By Theorem 1, ϑ has a finitely-branching tree model. Observe, by Theorem 3, that $|Q| = 2^{O(|\vartheta| \cdot \mathsf{deg}(\vartheta))}$, where Q is the state set of the automaton A_ϑ defined in that proof. Hence, it is enough to prove that every tree model of ϑ has a subtree of branching degree $|Q|^2$ that also models ϑ.

To prove this claim, we use the membership game $G_{\mathsf{T},\mathsf{A}_\vartheta}$ of the input tree T and the automaton A_ϑ. There are two players, *automaton* and *pathfinder*. Player automaton moves by resolving disjunctions in the transition relation of A_ϑ, and is trying to show that T is accepted by A_ϑ. Player pathfinder moves by resolving conjunctions, and is trying to show that T is not accepted by A_ϑ. The game uses auxiliary tree structured arenas to resolve each transition of the automaton. This is a simple case of a *hierarchical parity game* [3]. As usual, player automaton has a winning strategy if and only if $\mathsf{T} \models \mathsf{A}_\vartheta$. By memoryless determinacy of parity games on infinite arenas, player automaton has a winning strategy if and only if he has a memoryless winning strategy. For a fixed memoryless strategy *str*, one can prove, by looking at the transition function of A_ϑ, that every play consistent with *str*, and every node t of the input tree T, only visits at most $|Q|^2$ sons of t, thus inducing a subtree which is the required boundedly-branching tree model.

Theorem 6. *The satisfiability problem for* GCTL* *over LTSs is 2*ExpTime-Complete*, and model checking* GCTL* *for finite LTSs is* PSpace-Complete*.*

Proof. The lower-bounds already hold for CTL*. Theorems 3, 5 and Lemma 2 give the upper-bound for satisfiability. For the upper-bound for model checking, given an LTS S (with largest degree d), and a GCTL* formula ϑ, using Theorem 3 construct the GHTA $\mathsf{A}_{\neg\vartheta}$, which has $2^{O(|\vartheta| \cdot \mathsf{deg}(\vartheta))}$ states, transition function of size $2^{O(|\vartheta| \cdot \mathsf{deg}(\vartheta))}$, and depth $O(|\vartheta|)$. As in the proof of Lemma 2, build an equivalent AHTA A' of size $d + (2^{O(|\vartheta| \cdot \mathsf{deg}(\vartheta))} \cdot |Q|^d) + 2^{O(|\vartheta| \cdot \mathsf{deg}(\vartheta))} = 2^{O(|\vartheta| \cdot \mathsf{deg}(\vartheta) + d \cdot |\vartheta| \cdot \mathsf{deg}(\vartheta))}$, and of depth $\partial = O(|\vartheta|)$. By Theorem 2, the membership problem of the AHTA A' on S can be solved in space $O(\partial \log^2(|S| \cdot ||\mathsf{A}'||))$ which is polynomial in $|\vartheta|$ and $|S|$ (using $\mathsf{deg}(\vartheta) \leq |\vartheta|$ and $d \leq |S|$).

6 Discussion

This work shows that GCTL* is an expressive logic (it is equivalent, over trees, to MPL and can express fairness and counting over paths) whose satisfiability and model-checking problems have the same complexity as that of CTL*.

GCTL* was defined in [5]. However, only the fragment GCTL was studied. As the authors note in the conference version of that paper, their techniques, that worked for GCTL, do not work for GCTL*. Moreover, they also suggested a line of attack that does not seem to work; indeed, it was left out of the journal version of their paper [6]. Instead, our method is a careful combination of the automata-theoretic approach to branching-time logics [20], a characterization of the graded path modality (Proposition 1), and a boundedly-branching tree model property whose proof uses game-theoretic arguments (Theorem 5). Moreover, our technique immediately recovers the main results about GCTL from [5], i.e., satisfiability for GCTL is ExpTime-Complete and the model checking

problem for GCTL is in PTime (Indeed, consider the construction in Theorem 3 of A_ϑ when ϑ it taken from the fragment GCTL of GCTL*, and in particular where it comes to a subformula ϕ of the form $\phi = \mathsf{E}^{\geq g}\psi$. Since ψ is either of the form $p\mathsf{U}q$ or $\mathsf{X}p$, the number of new states added at this stage is a constant. Thus, the number of states of A_ϑ is linear in the size of ϑ). In other words, our technique suggests a powerful new way to deal with graded path modalities.

When investigating the complexity of a logic with a form of counting quantifiers, one must decide how the numbers in these quantifiers contribute to the length of a formula, i.e., to the input of a decision procedure. In this paper we assume that these numbers are coded in unary, rather than binary. There are a few reasons for this. First, the unary coding naturally appears in description and predicate logics [8]. As pointed out in [19], this reflects the way in which many decision procedures for these logics work: they explicitly generate n individuals for $\exists^{\geq n}$. Second, although the complexity of the binary case is sometimes the same as that of the unary case, the constructions are significantly more complicated, and are thus much harder to implement [6, 7]. At any rate, as the binary case is useful in some circumstances we plan to investigate this in the future.

Comparison with (some) other approaches. Although showing that satisfiability of GCTL* is decidable is not hard (for example, by reducing to MSOL), identifying the exact complexity is much harder. Indeed, there is no known satisfiability-preserving translation of GCTL* to another logic that would yield the optimal 2ExpTime upper bound. We discuss two such candidate translations. First, in this article we show a translation from GCTL* to MPL. Unfortunately, the complexity of satisfiability of MPL is non-elementary. Second, there is no reason to be optimistic that a translation from GCTL* to $G\mu$-calculus (whose satisfiability is ExpTime-Complete) would yield the optimal complexity since a) already the usual translation from CTL* to μ-calculus does not yield optimal complexity [13], and b) the translation given in [6] from GCTL to $G\mu$-calculus does not yield optimal complexity. Moreover, the usual translation from CTL* to μ-calculus uses automata, and thus automata for GCTL* (from which we get our results directly) have to be developed anyway.

Future work. Recall that the graded μ-calculus was used to solve questions (such as satisfiability) for the description logic $\mu\mathcal{ALCQ}$ [7]. Similarly, our techniques for GCTL* might be useful for solving questions in $\mathcal{ALCQ}$ combined with temporal logic, such as for the graded extension of CTL$^*_{\mathcal{ALC}}$ [17]. Second, the GCTL model checking algorithm from [6] has been implemented in the NuSMV model-checker to provide more than one counter-example when a GCTL formula is not satisfied. We are thus optimistic that existing CTL* model-checkers can be fruitfully extended to handle GCTL*.

References

1. S. Almagor, U. Boker, and O. Kupferman. What's decidable about weighted automata? In *Intl. Symp. on Automated Technology for Verification and Analysis*, volume 6996 of *LNCS*, pages 482–491, 2011.

2. B. Aminof, O. Kupferman, and R. Lampert. Rigorous approximated determinization of weighted automata. In *Symp. on Logic in Computer Science*, pages 345–354, 2011.
3. B. Aminof, O. Kupferman, and A. Murano. Improved model checking of hierarchical systems. *Information and Computation*, 210:68–86, 2012.
4. M. Arenas, P. Barceló, and L. Libkin. Combining Temporal Logics for Querying XML Documents. In *ICDT*, LNCS 4353, pages 359–373, 2007.
5. A. Bianco, F. Mogavero, and A. Murano. Graded computation tree logic. In *Symp. on Logic in Computer Science*, pages 342–351. IEEE, 2009.
6. A. Bianco, F. Mogavero, and A. Murano. Graded computation tree logic. *ACM Transactions on Computational Logic*, 13(3):25, 2012.
7. P.A. Bonatti, C. Lutz, A. Murano, and M.Y. Vardi. The Complexity of Enriched Mu-Calculi. *Logical Methods in Computer Science*, 4(3):1–27, 2008.
8. D. Calvanese, G. De Giacomo, and M. Lenzerini. Reasoning in expressive description logics with fixpoints based on automata on infinite trees. In *International Joint Conference on Artificial Intelligence*, pages 84–89, 1999.
9. M. de Rijke. A note on graded modal logic. *Studia Logica*, 64(2):271–283, 2000.
10. M. Droste, W. Kuich, and H. Vogler. *Handbook of Weighted Automata*. Springer, 2009.
11. C. Eisner, D.Fisman, J. Havlicek, Y. Lustig, A. McIsaac, and D. V. Campenhout. Reasoning with temporal logic on truncated paths. In *Computer Aided Verification*, LNCS 2725, pages 27–39, 2003.
12. E.A. Emerson and C.S. Jutla. The complexity of tree automata and logics of programs. *SIAM Journal of Computing*, 29(1):132–158, 1999.
13. E.A. Emerson and A.P. Sistla. Deciding branching time logic. In *Symposium on Theory of Computing*, pages 14–24, 1984.
14. A. Ferrante, A. Murano, and M. Parente. Enriched μ-calculi module checking. *Logical Methods in Computer Science*, 4(3), 2008.
15. A. Ferrante, M. Napoli, and M. Parente. Model Checking for Graded CTL. *Fundamenta Informaticae*, 96(3):323–339, 2009.
16. K. Fine. In So Many Possible Worlds. *Notre Dame Journal of Formal Logic*, 13:516–520, 1972.
17. V. Gutiérrez-Basulto, J.C. Jung, and C. Lutz. Complexity of branching temporal description logics. In *Euro. Conf. on Artificial Intelligence*, pages 390–395, 2012.
18. T.A. Henzinger. Quantitative reactive modeling and verification. *Comput. Sci.*, 28(4):331–344, November 2013.
19. O. Kupferman, U. Sattler, and M.Y. Vardi. The Complexity of the Graded μ-Calculus. In *Conference on Automated Deduction*, LNCS 2392, pages 423–437. Springer, 2002.
20. O. Kupferman, M.Y. Vardi, and P. Wolper. An Automata Theoretic Approach to Branching-Time Model Checking. *Journal of the ACM*, 47(2):312–360, 2000.
21. V. Malvone, F. Mogavero, A. Murano, and L. Sorrentino. On the counting of strategies. In *Int. Symp. on Temporal Representation and Reasoning*, 2015.
22. F. Moller and A. Rabinovich. Counting on CTL*: On the expressive power of monadic path logic. *Information and Computation*, 184(1):147–159, 2003.
23. S. Tobies. PSPACE Reasoning for Graded Modal Logics. *Journal of Logic and Computation*, 11(1):85–106, 2001.
24. W. van der Hoek and JJ.Ch. Meyer. Graded modalities in epistemic logic. In *Symp. on Logical Foundations of Computer Science*, pages 503–514, 1992.
25. M. Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, 1994.