Alternating Nonzero Automata

Paulin Fournier
LS2N, Université de Nantes, France

Hugo Gimbert
CNRS, LaBRI, Université de Bordeaux, France

Abstract

We introduce a new class of automata on infinite trees called *alternating nonzero automata*, which extends the class of non-deterministic nonzero automata. The emptiness problem for this class is still open, however we identify a subclass, namely limited choice, for which we reduce the emptiness problem for alternating nonzero automata to the same problem for non-deterministic ones, which implies decidability. We obtain, as corollaries, algorithms for the satisfiability of a probabilistic temporal logic extending both CTL* and the qualitative fragment of pCTL*.

2012 ACM Subject Classification Theory of computation → Complexity theory and logic

Keywords and phrases zero-automata, probabilities, temporal logics

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2018.13

Funding Both authors received support for this work from the French ANR projet "Stoch-MC".

Acknowledgements The authors wish to thank Mikołaj Bojańczyk, Henryk Michalewski and Matteo Mio for interesting discussions on TMSO+ZERO as well as zero- and nonzero-automata; and the referees for their helpful comments.

1 Introduction

The theory of automata on infinite trees is rooted in Rabin's seminal theorem which establishes an effective correspondence between the monadic second order logic (MSO) theory of the infinite binary tree and the non-deterministic automata on this tree [18]. In this correspondence, the satisfiability of the logic is dual to the emptiness of the algorithm and both these algorithmic problems are mutually reducible to one another.

This elegant setting has been partially extended to probabilistic logics [13, 6, 14, 15, 2] and automata with probabilistic winning conditions [18, 17, 1, 7, 2]. In this paper we make another step in this direction: we show a correspondence between the logic $\mathrm{CTL}^*[\exists, \forall, \mathbb{P}_{>0}, \mathbb{P}_{=1}]$ and nonzero alternating automata with limited choice. Moreover we show that the emptiness problem of the automata is decidable and obtain as a corollary the decidability of the satisfiability of the logic.

Automata. Alternating nonzero automata are an alternating version of *non-deterministic nonzero automata* introduced in [3], which themselves are equivalent to *non-deterministic zero automata* introduced in [2].

An alternating nonzero automaton takes as input a binary tree. Some states of the automaton are controlled by Eve, while other states are controlled by Adam, and the player controlling the current state chooses the next transition. Some transitions are *local transitions*, in which case the automaton stays on the same node of the input tree while other are *split*

transitions in which case the automaton proceeds to the left son or to the right son of the current node with equal probability $\frac{1}{2}$.

This interaction between Eve and Adam is seen as a game where Eve and Adam play according to some strategies. Once the strategies are fixed, one obtains a Markov chain whose trajectories are all possible plays consistent with the strategies. The winner is determined with respect to winning conditions introduced in [2, 3], using a total order on the set of states (used to compute the limsup of a play which is the largest state seen infinitely often during the play) and three subsets of states, respectively called the *sure*, *almost-sure* and *positive states*. Eve wins if and only if the three acceptance conditions hold:

sure winning: every play has limsup in sure states; and

almost-sure winning: almost-every play has limsup in almost-sure states; and

positive winning: whenever the play enters a positive state there is positive probability that the play never exits positive states.

The input tree is accepted by the alternating automaton iff Eve has a winning strategy.

Alternating nonzero automata generalize both classical alternating automata with parity conditions [8, 16] (when all states are almost-sure and positive) as well as non-deterministic nonzero automata [3] (in case Eve controls all states).

We do not know whether the emptiness problem for these automata is decidable or not, however we show that the answer is positive for the subclass of alternating nonzero automata with *limited choice for Adam*. In these automata, some choices of Adam are canonical, at most one in every state, and Adam may perform at most a bounded number of non-canonical choices during a single play.

First, we show that the emptiness problem for alternating nonzero automata with limited choice for Adam is in NEXPTIME$\cap$ co-NEXPTIME (Theorem 22). The proof is an EXPTIME reduction to the emptiness problem for non-deterministic automata. This proof relies on the positional determinacy of the acceptance games for Eve (Lemma 10) and a characterization of positional winning strategies for Eve (Lemmas 12, 13 and 17).

Second, we show that in the particular case where the sure winning condition is a Büchi condition, emptiness of non-deterministic nonzero automata is in PTIME (Theorem 3). It follows that, in case of a trivial sure winning condition, emptiness of alternating nonzero automata with limited choice for Adam is in EXPTIME (Theorem 22).

Logic. The temporal logic CTL* introduced by Emerson and Halpern [9] and its fragments CTL and LTL are prominent tools to specify properties of discrete event systems.

A variant of CTL* is the logic pCTL* [11] in which the universal and existential path quantifiers are *replaced* by probabilistic path quantifiers which set upper or lower bounds on the probability of a path property in a Markov chain. For example the formula $\mathbb{P}_{\geq \frac{1}{2}}(FGa)$ specify that with probability at least $\frac{1}{2}$ eventually all the visited states are labelled with a. To our knowledge, the satisfiability problem for this logic is an open problem.

However, for the qualitative fragment of pCTL*, where only two probabilistic quantifiers $\mathbb{P}_{>0}$ and $\mathbb{P}_{=1}$ are available, the satisfiability is decidable [6]. In a variant of pCTL* called pECTL the path subformula are replaced by deterministic Büchi automaton, and the satisfiability of the qualitative fragment is 2-EXPTIME complete [6], the same complexity as for CTL* [19].

Remark that neither pCTL* nor pECTL includes the path operators $\forall$ and $\exists$, thus these two logics are incomparable in expressivity with CTL*. For example, on the alphabet $\{a, b\}$, the CTL* formula $\phi_1 = \forall FG\neg b$, and the pCTL* formula $\phi_2 = \mathbb{P}_{=1}(FG\neg b)$ specify, that

every branch, respectively *almost-every* branch, of the model has finitely many b. Neither ϕ_1 can be expressed in pCTL* nor ϕ_2 can be expressed in CTL*.

In this paper, we consider the logic CTL$^*[\exists, \forall, \mathbb{P}_{>0}, \mathbb{P}_{=1}]$ which is an extension of both CTL* and qualitative pCTL*and establish several properties of this logic.

The satisfiability by an arbitrary Σ-labelled Markov chain reduces to the satisfiability by $(\Sigma \cup \{\circ\})$-labelled a binary tree with $\circ$ a fresh letter (Theorem 24).

The satisfiability of CTL$^*[\exists, \forall, \mathbb{P}_{>0}, \mathbb{P}_{=1}]$ reduces to the emptiness of alternating nonzero automata with finite choice for Adam thus it is decidable in 3-NEXPTIME$\cap$co-3-NEXPTIME. In the variant ECTL$[\exists, \forall, \mathbb{P}_{>0}, \mathbb{P}_{=1}]$, where path formula are deterministic Büchi automata, this reduction gives a 2-NEXPTIME$\cap$ co-2-NEXPTIME complexity and for the fragment CTL$[\exists, \forall, \mathbb{P}_{>0}, \mathbb{P}_{=1}]$ the complexity is NEXPTIME$\cap$ co-NEXPTIME (Theorem 23).

For the fragments CTL$^*[\mathbb{P}_{>0}, \mathbb{P}_{=1}]$, ECTL$[\mathbb{P}_{>0}, \mathbb{P}_{=1}]$ and CTL$[\mathbb{P}_{>0}, \mathbb{P}_{=1}]$ (i.e. qualitative pCTL*, pECTL and pCTL respectively), the $F_\forall$ acceptance condition of the automaton is a Büchi condition, and we retrieve the optimal complexity bounds of [6, 5], i.e. 3-EXPTIME, 2-EXPTIME and EXPTIME, respectively.

A motivation for the study of alternating nonzero automata is the recent research on the logic TMSO+ZERO. The logic TMSO+ZERO is an extension of Monadic second-order logic on infinite binary trees with a new probabilistic operator [14, 15, 2]. The satisfiability of this logic is reducible to the emptiness problem for nonzero non-deterministic automata [2] which is decidable [3]. Since CTL$^*[\exists, \forall, \mathbb{P}_{>0}, \mathbb{P}_{=1}]$ is a fragment of TMSO+ZERO, this result implies that the satisfiability of CTL$^*[\exists, \forall, \mathbb{P}_{>0}, \mathbb{P}_{=1}]$ is decidable with non-elementary complexity. The reduction to the emptiness of alternating nonzero automata given in the present paper provides a better complexity bound.

2 Alternating nonzero automata

An alternating nonzero automaton on a finite alphabet Σ is a finite-state machine processing binary trees, equipped with a game semantics: every tree is either accepted or rejected by the machine depending on who wins the acceptance game on the tree.

Trees. A Σ-labelled binary tree (or Σ-tree for short) is a function $t : \{0,1\}^* \to \Sigma$. An element $n \in \{0,1\}^*$ is called a *node* of the tree and has exactly two sons $n0$ and $n1$. We use the usual notions of ancestors and descendants. A node n' is *(strictly) below* n if n is a (strict) prefix of n'. A *path* in the tree is a finite or infinite sequence of nodes $n_0, n_1, \ldots$ such that for every k the node n_{k+1} is a son of the node n_k.

A branch b is an element of $\{0,1\}^\omega$. If a node n is a prefix of b we say that n *belongs* to b or that b *visits* n. The set of branches is equipped with the uniform probability measure, denoted μ, corresponding to an infinite random walk taking at each step either direction 0 or 1 with equal probability $\frac{1}{2}$.

A set of nodes $T \subseteq \{0,1\}^*$ is a *subtree* if it contains a node r, called the root of T, such that every node $n \in T$ is a descendant of r, T contains all nodes on the path from r to n. A subtree is full if T contains all descendants of r.

Automata. An alternating nonzero automaton on alphabet Σ is presented as a tuple $\mathcal{A} = (Q, q_0, Q_E, Q_A, \to, F_\forall, F_1, F_{>0})$ where:

- Q is a finite set of states, equipped with a total order $\leq$, containing the initial state q_0.
- (Q_E, Q_A) is a partition of Q into Eve and Adam states.

- $\to$ is the set of transitions, there are two types of transitions: *local transitions* which are tuples (q, a, q') with $q, q' \in Q$ and $a \in \Sigma$, denoted $q \to_a q'$; and *split transitions* which are tuples $(q, a, q_0, q_1) \in Q \times \Sigma \times Q^2$, denoted $q \to_a (q_0, q_1)$.
- $F_\forall$, F_1 and $F_{>0}$ are subsets of Q defining the acceptance condition.

The input of such an automaton is an infinite binary tree $t : \{0,1\}^* \to \Sigma$. The source (resp. the target) of a local transition $q \to_a q'$ is q (resp q'). The source (resp. the targets) of a split transition $q \to_a (q_0, q_1)$ is q (resp q_0 and q_1). A state is said to be controlled by Eve or Adam whether it belongs to Q_E or Q_A. The controller of a transition is the controller of its source state. We always assume that

(HC) the automaton is **complete**: for every state q and letter a there is at least one transition with source q on a.

The (HC) condition makes it easier to define the game semantics of the automaton.

Game semantics. The acceptance of an input binary tree by the automaton is defined by mean of a stochastic game between Eve and Adam called the *acceptance game*.

The game of acceptance of a binary tree $t : \{0,1\}^* \to \Sigma$ by $\mathcal{A}$ is a two-player stochastic game with perfect information played by two strategic players Eve and Adam. The vertices of the game are all pairs (n, q) where $n \in \{0,1\}^*$ is a node of the infinite binary tree and q is a state of the automaton. The game starts in the initial vertex (ϵ, q_0).

Each vertex (n, q) is controlled by either Eve or Adam depending on whether $q \in Q_E$ or $q \in Q_A$. The controller of the current state chooses any transition with source q and letter $t(n)$. Intuitively, depending on whether the transition is a local or a split transition, the automaton stays on the current node n or move with equal probability $\frac{1}{2}$ to either node $n0$ or $n1$. If the transition is a local transition $q \to_{t(n)} q'$, the new vertex of the game is (n, q'). If the transition is a split transition $q \to_{t(n)} (r_0, r_1)$ then the new vertex is chosen randomly with equal probability $\frac{1}{2}$ between vertices $(n0, r_0)$ or $(n1, r_1)$.

A play is a finite or infinite sequence of vertices $\pi = (n_0, q_0)(n_1, q_1)\ldots$. We denote $\text{first}(\pi) = (n_0, q_0)$ and $\text{last}(\pi) = (n_k, q_n)$ (for finite plays).

A strategy for Eve associates with every finite play whose last vertex is controlled by Eve a transition with source q_n and letter $t(n_k)$ (such a transition always exists since the automaton is complete). Strategies for Adam are defined in a symmetric way. Strategies of Eve are usually denoted σ while strategies for Adam are denoted τ.

Measuring probabilities. Once both players Eve and Adam have chosen some strategies σ and τ, this defines naturally a non-homogenous Markov chain whose states are the vertices of the game. According to Tulcea theorem, if we equip the set of plays with the σ-field generated by cylinders, then there is a unique probability measure $\mathbb{P}^{\sigma,\tau}$ such that after a play $\pi = (n_0, q_0)\ldots(n_k, q_k)$, if $\delta(\pi)$ denotes the transition chosen by Eve or Adam after π (depending on whether $q_k \in Q_E$ or $q_k \in Q_A$), the probability to go to vertex (n_{k+1}, q_{k+1}) is:

$$\begin{cases} 1 & \text{if } \delta(\pi) \text{ is the local transition } q_k \to_{t(n_k)} q_{k+1} \ , \\ \frac{1}{2} & \text{if } \delta(\pi) \text{ is the split transition } q_k \to_{t(n_k)} (r_0, r_1) \text{ and } \begin{cases} n_{k+1} = n_k 0 \text{ and } q_{k+1} = r_0 \ ; \textit{or} \\ n_{k+1} = n_k 1 \text{ and } q_{k+1} = r_1 \ . \end{cases} \\ 0 & \text{otherwise .} \end{cases}$$

This way we obtain a probability measure $\mathbb{P}^{\sigma,\tau}$ on the set of infinite plays.

Consistency and reachability. If a finite play π is the prefix of another finite or infinite play π' we say that π' is a *continuation* of π. A finite π play is *consistent* with a strategy σ or, more simply, is a σ*-play* if there exists a strategy τ such that π may occur in the non-homogenous Markov chain induced by σ and τ. In this case, the number N of split transitions which occurred in π is exactly the depth of the node of $\mathrm{last}(\pi)$ and $\mathbb{P}^{\sigma,\tau}(\{ \text{ continuations of } \pi \}) = 2^{-N}$. A vertex w is σ*-reachable* if there exists a finite σ-play from the initial vertex to w. An infinite play is consistent with σ if all its prefixes are.

Bounded vs. unbounded plays. There are two kinds of infinite plays: *bounded plays* are plays whose sequence of nodes is ultimately constant, or equivalently which ultimately use only local transitions while *unbounded plays* use infinitely many split transitions.

Bounded plays consistent with σ and τ are the atoms of $\mathbb{P}^{\sigma,\tau}$: a play π is bounded and consistent with σ and τ iff $\mathbb{P}^{\sigma,\tau}(\{\pi\}) > 0$.

In this paper we will focus on subclasses of automata whose structural restrictions forbids the existence of bounded plays (see the **(NLL)** hypothesis below).

So in practice, every play $\pi = (n_0, q_0)(n_1, q_1)\dots$ we consider will visit a sequence of nodes $n_0, n_1, n_2, \dots$ which enumerates all finite prefixes of an infinite branch $b \in \{0,1\}^\omega$ of the binary tree, in a weakly increasing order: for every index i either $n_{i+1} = n_i$ (the player controlling (n_i, q_i) played a local transition) or $n_{i+1} = n_i d$ for some $d \in \{0,1\}$ (the player controlling (n_i, q_i) played a split transition and the play followed direction d).

Winning strategies and language. Whether Eve wins the game is defined as follows. The *limsup* of an infinite play $(n_0, q_0)(n_1, q_1)\dots$ is $\limsup_i q_i$ i.e. the largest automaton state visited infinitely often. An infinite play π' is a *positive continuation* of π if all states of π' visited after π belongs to $F_{>0}$.

Eve wins with σ against τ if the three following conditions are satisfied.

Sure winning. Every play consistent with σ and τ has limsup in $F_\forall$.

Almost-sure winning. Almost-every play consistent with σ and τ has limsup in F_1.

Positive winning. For every finite play π consistent with σ and τ whose last state belongs to $F_{>0}$, the set of positive continuations of π has nonzero probability.

A Büchi condition is a set of states $R \subseteq Q$ which is upper-closed with respect to $\leq$. Then a play has limsup in R iff it visits R infinitely often.

We say that *Eve wins* the acceptance game if she has a *winning strategy* i.e. a strategy which wins the acceptance game against any strategy of Adam.

▶ **Definition 1** (Acception and language)**.** A binary tree is *accepted* by the automaton if Eve has a winning strategy in the acceptance game. The language of the automaton is the set of its accepted trees.

We are interested in the following decision problem:

Emptiness problem: Given an automaton, decide whether its language is empty or not.

Alternation and the use of game semantics makes the following closure properties trivial.

▶ **Lemma 2** (Closure properties)**.** *The class of languages recognized by alternating nonzero automata is closed under union and intersection.*

Normalization. We assume all automata to be normalized in the sense where they satisfy:

(N1) every split transition whose source is in $F_{>0}$ has at least one successor in $F_{>0}$; and

(N2) every local transition whose source is in $F_{>0}$ has its target in $F_{>0}$ as well.

We can normalize an arbitrary automaton by removing all transitions violating **(N1)** and **(N2)**. This will not change the language because such transitions are never used by positively winning strategies of Eve. This normalization could lead to a violation of the completeness hypothesis, **(HC)**. In this case we can also delete the corresponding states without modifying the language of the automaton.

If one would drop **(HC)** then the game graph may have dead-ends and the rules of the game would have to be extended to handle this case, typically the player controlling the state in the dead-end loses the game. This extension does not bring any extra expressiveness to our model of automaton, we can always make an automaton complete by adding local transitions leading to losing absorbing states.

Moreover, we assume:

(N3) $F_1 \subseteq F_\forall$.

This is w.l.o.g. since replacing F_1 with $F_1 \cap F_\forall$ does not modify the language of the automaton.

Non-deterministic nonzero automata Non-deterministic *zero* automata were introduced in [2], followed by a variant of equivalent expressiveness, non-deterministic *nonzero* automata [4, Lemma 5]. In those automata, Adam is a dummy player, i.e. $Q_A = \emptyset$ and moreover all transitions are split-transitions.

▶ **Theorem 3.** *The emptiness problem for non-deterministic nonzero automata is in* NP∩coNP. *If $F_\forall$ is a Büchi condition then emptiness can be decided in* PTIME.

The first statement is established in [3, Theorem 3]. The second statement is proved in the appendix. The proof idea is as follows. Assume the alphabet to be a singleton, which is w.l.o.g. for non-deterministic automata. The existence of a winning strategy for Eve can be witnessed by a subset $W \subseteq Q$ which contains the initial state and two positional winning strategies $\sigma_1, \sigma_2 : W \to W \times W$. Strategy σ_1 should be almost-surely and positively winning while strategy σ_2 should be surely winning. These two strategies can be combined into a (non-positional) strategy for Eve which satisfies the three objectives, thus witnesses non-emptiness of the automaton.

3 An example: the language of PUCE trees

A $\{a, b\}$-tree is *positively ultimately constant everywhere* (*PUCE*) if for every node n,

- **i)** the set of branches visiting n and with finitely many a-nodes has > 0 probability; and
- **ii)** the set of branches visiting n and with finitely many b-nodes has > 0 probability.

No regular tree is PUCE. There are two cases. If the regular tree has a node n which is the root of a full subtree labelled with a single letter (either a or b) then clearly the tree is not PUCE. Otherwise, by a standard pumping argument, every node labelled a (resp. b) has a descendant labelled b (resp. a) at some depth $\leq |S|$, where S is the set of states of the regular tree. But in this second case from every node n there is probability at least $\frac{1}{2^{|S|}}$ to reach a descendant with a different label, thus almost-every branch of the regular tree has infinitely many a and b, and the tree is not PUCE either.

There exists a PUCE tree. However it is possible to build a non-regular tree t whose every node satisfies both i) and ii). For that, we combine together two partial non-regular trees. Let $H \subseteq \{0, 1\}^*$ be a subset of nodes such that a) the set of branches which visit no node in

H has probability $\frac{1}{2}$, b) every node in $\{0,1\}^*$ is either a descendant or an ancestor of a node in H, but not both (H is a cut).

To obtain t, we combine two partial trees t_a and t_b whose domain is $\{0,1\}^* \setminus (H\{0,1\}^+)$ and t_a is fully labeled with a while t_b is fully labelled with b. Since H is a cut, the nodes in H are exactly the leaves of t_a and t_b. To obtain t, we plug a copy of t_b on every leaf of t_a and a copy of t_a on every leaf of t_b. Then from every node, according to b) there is non-zero probability to enter either t_a or t_b and according to a) there is non-zero probability to stay in there forever.

An automaton recognizing PUCE trees. We can design one automaton for each of the two conditions and combine them together with an extra state controlled by Adam (cf proof of Lemma 2). We provide an alternating nonzero automaton checking condition ii), the automaton for condition i) is symmetric. The state space is: $Q = \{s < w < g < \sharp\}$.

Intuitively, Adam uses states s to search for a node n from which condition ii) does not hold. Once on node n, Adam switches to state w and challenges Eve to find a path to an a-node n' which is the root of an a-labelled subtree T_n of > 0 probability. For that Eve navigates the tree in state w to node n', switches to state g on node n', stays in g as long as the play stays in T_n and switches definitively to $\sharp$ whenever leaving T_n.

Formally, the only state controlled by Adam is s, i.e. $Q_A = \{s\}$, from which Adam can choose, independently of the current letter, between two split transitions $s \to (s, \sharp)$ and $s \to (\sharp, s)$ and a local transition $s \to w$. The state $\sharp$ is absorbing. From state w, Eve can guess the path to n' using the split transitions: $w \to (\sharp, w) \quad w \to (w, \sharp)$.

Once n' is reached Eve can switch to state g with a local transition $w \to g$ and, whenever the current node is an a-node, she can choose among three split transitions: $g \to_a (g, g) \quad g \to_a (g, \sharp) \quad g \to_a (\sharp, g)$ to identify T_n.

The acceptance conditions are: $F_\forall = F_1 = Q \setminus \{w\}$ and $F_{>0} = \{g\}$, so that from w Eve is forced to eventually switch to g (otherwise $\limsup = w \notin F_\forall$) and the a-subtree labelled by g must have positive probability for Eve to win. Adam may never exit the pathfinding state s, in which case Eve wins.

4 Deciding emptiness of automata with limited choice for Adam

In this section, we introduce the class of automata with *limited choice for Adam*, and show that emptiness of these automata is decidable.

For that we rely on a characterization of positional strategies of Eve which satisfy the surely and almost-surely winning conditions (Lemma 12, Lemma 13) and the positively winning condition (Lemma 17). Then we represent the positional strategies of Eve as labelled trees, called *strategic trees* (Definition 18). Finally, we show that the language of strategic trees whose corresponding positional strategy is winning can be recognized by a non-deterministic nonzero automaton (Theorem 19).

4.1 Automata with limited choice for Adam

In the rest of the paper, we focus on the class of automata with limited choice for Adam. Our motivation is that these automata capture the logic we are interested in and their acceptance games have good properties. In particular the existence of positional winning strategies for Eve is one of the key properties used to decide emptiness.

To define the class of automata with limited choice for Adam, we rely on the transition graph of the automaton.

▶ **Definition 4** (Equivalent and transient states)**.** The transitions of the automaton define a directed graph called the *transition graph* and denoted $G_{\to}$. The vertices of $G_{\to}$ are Q and the edges are labelled with Σ, those are all triplets (q, a, r) such that $q \to_a r$ is a local transition or such that $q \to_a (r, q')$ or $q \to_a (q', r)$ is a split transition for some state q'.

Two states q, r are *equivalent*, denoted $q \equiv r$, if they are in the same connected component of $G_{\to}$. A state is *transient* if it does not belong to any connected component of $G_{\to}$, or equivalently if there is no cycle on this state in $G_{\to}$.

▶ **Definition 5.** An automaton has limited choice for Adam if for every state q controlled by Adam, all transitions with source q are local transitions; and for every letter a, at most one of the (local) transitions $q \to_a q'$ satisfies $q \equiv q'$. Such a transition is called a *canonical* transition.

In a limited choice for Adam automaton, the only freedom of choice of Adam, apart from playing canonical transitions, is deciding to go to a lower connected component of the transition graph. This non-canonical decision can be done only finitely many times, hence the name *limited choice*.

In the classical (non-probabilistic) theory of alternating automata, similar notions of limited alternation have already been considered, for example *hesitant alternating automata* [12].

▶ **Definition 6** (Canonical plays and transient vertices)**.** A *canonical play* is a play in which Adam only plays canonical transitions. A vertex (n, q) of an acceptance game is *transient* if it has no immediate successor (n', q') (by a local or a split transition) such that $q \equiv q'$.

In the acceptance game of an automaton with limited choice for Adam, every infinite play visit finitely many transient vertices and has a canonical suffix.

The automaton recognizing PUCE trees described in Section 3 has not limited choice for Adam, since Adam can play the non-local transitions $s \to (s, \sharp)$ and $s \to (\sharp, s)$. However, it is an easy exercise to turn this automaton into an automaton with limited choice for Adam recognizing the same language: add a new state e controlled by Eve from which Eve has a single split transition $e \to (s, s)$. This new state e belongs to both $F_\forall$ and F_1. From s Adam can choose between two local transitions: the canonical transition $s \to e$ (keep navigating in the tree) or the transient transition $s \to w$ (start verification).

The no local loop assumption. We assume that every automata with limited choice for Adam also satisfies:

(NLL) the automaton has **no local loop**: there is no letter a and sequence of local transitions $q_0 \to_a q_1 \to_a \cdots \to_a q_i$ such that $q_0 = q_i$.

Under the hypothesis **(NLL)**, for every infinite play π there is a unique branch of the binary tree $b \in \{0, 1\}^\omega$ whose every prefix is visited by π. We say that π *projects* to b. It is direct that, under the hypothesis **(NLL)**, the measures on plays and on branches are linked.

▶ **Lemma 7.** *Under the hypothesis **(NLL)**, given a tree t, two strategies σ and τ in the acceptance game and X is a measurable set of plays we have that $\mathbb{P}^{\sigma,\tau}(X) = \mu(\tilde{X})$ where μ is the usual uniform measure on the set of branches of t and $\tilde{X}$ is the set of infinite branches that X projects to.*

Moreover, assuming **(NLL)** does not reduce expressiveness.

▶ **Lemma 8.** *Given an automaton $\mathcal{A}$ with limited choice for Adam and set of states Q one can effectively construct another automaton $\mathcal{A}'$ with limited choice for Adam satisfying **(NLL)** and recognizing the same language.*

The interest of the **(NLL)** assumption is to make the acceptance game acyclic, which in turn guarantees positional determinacy for Eve, as shown in the next section. The transformation performed in the proof of Lemma 8 creates an exponential blowup of the state space of the automaton, which is bad for complexity. We could do without this blowup by dropping the **(NLL)** assumption, in which case Eve might need one extra bit of memory in order to implement local loops with priority in $F_\forall \setminus F_1$.

However, we prefer sticking to the **(NLL)** assumption, which makes the alternating automata and their accepting games simpler and is anyway not restrictive when it comes to translating temporal logics into alternating automata: the natural translation produces automata with no local loop.

4.2 Positional determinacy of the acceptance game

A crucial property of automata with limited choice for Adam is that their acceptance games are positionally determined for Eve.

▶ **Definition 9** (Positional strategies)**.** A strategy σ of Eve in an acceptance game is *positional* if for every finite plays π, π' whose last vertices are controlled by *Eve* and coincide, i.e. $\text{last}(\pi) = \text{last}(\pi') \in \{0,1\}^* \times Q_E$, then $\sigma(\pi) = \sigma(\pi')$.

▶ **Lemma 10** (Positional determinacy for Eve)**.** *Every acceptance game of an automaton with limited choice for Adam is positionally determined for Eve: if Eve wins then she has a positional winning strategy.*

Sketch of proof. Since the **(NLL)** hypothesis is assumed, the underlying acceptance game is acyclic. The construction of a positional winning strategy σ' from a (non-positional) winning strategy σ relies on the selection of a canonical way of reaching a σ-reachable vertex w with a σ-play $\pi(w)$ and setting $\sigma'(w) = \sigma(\pi(w))$. ◀

4.3 On winning positional strategies of Eve

In the next section we show how to use use automata-based techniques to decide the existence of a (positional) winning strategy for Eve. These techniques rely on characterizing whether a positional strategy of Eve is surely, almost-surely and positively winning.

4.3.1 Surely and almost-surely winning conditions

We characterize (almost-)surely winning strategies.

▶ **Definition 11** (q-branches)**.** Let $q \in Q$ and σ a strategy. An infinite branch of the binary tree is a q-branch in σ if at least one σ-play which projects to this branch has limsup q.

▶ **Lemma 12.** *Assume the automaton has limited choice for Adam. Let σ be a positional strategy for Eve. Then σ is surely winning iff for every $q \in (Q \setminus F_\forall)$ there is no q-branch in σ. Moreover σ is almost-surely winning iff for every $q \in (Q \setminus F_1)$ the set of q-branches in σ has measure 0.*

Whether a branch is a q-branch can be checked by computing a system of σ-indexes. Intuitively, all σ-reachable vertices receives a finite index, such that along a σ-play the index does not change except when Adam performs a non-canonical move or when two plays merge on the same vertex, in which case the smallest index is kept. After a non-canonical move of Adam, a new play may start in which case it receives a fresh index not used yet in the current neither in the parent node. For this less than $2|Q|$ indices are required. The important properties of σ-indexes are:

▶ **Lemma 13** (Characterization of q-branches). *Every positional strategy σ of Eve can be associated with a function $\sigma : \{0,1\}^* \times Q \to \{0,1,\dots,2|Q|,\infty\}^Q$ with the following properties.*

First, σ can be computed on-the-fly along a branch. For every node n denote σ_n the restriction of σ on $\{n\} \times Q$. Then $\sigma(\epsilon)$ only depends on σ_ϵ. And for every node n and $d \in \{0,1\}$, $\sigma(nd)$ only depends on $\sigma(n)$ and σ_{nd}.

Second, a vertex (n,q) is reachable from the initial vertex by a σ-play iff $\sigma(n)(q)$ is finite.

Third, let $b \in \{0,1\}^\omega$ be an infinite branch of the binary tree, visiting successively the nodes $n_0, n_1, n_2, \dots$. Denote $R^\infty(b)$ the set of pairs $(k,q) \in \{0,\dots,2|Q|\} \times Q$ such that: $k \in \sigma(n_i)(Q)$ for every $i \in \mathbb{N}$ except finitely many; and $k = \sigma(n_i)(q)$ for infinitely many $i \in \mathbb{N}$.

Finally, for every state q, the branch b is a q-branch if and only if there exists $k \in \{0,1,\dots,2|Q|\}$ such that $q = \max\{r \in Q \mid (k,r) \in R^\infty(b)\}$.

4.3.2 Checking the positively winning condition

In order to check with a non-deterministic automaton whether a positional strategy is positively winning, we rely on the notion of *positive witnesses*. The point of positive witnesses is to turn the verification of up to $|Q|$ positively-winning conditions - depending on the decisions of Adam, there may be up to $|Q|$ different σ-reachable vertices on a given node - into a single one. This single condition can then be checked by a non-deterministic nonzero automaton equipped with a single positively-winning condition.

Everywhere thick subtrees. We need the notion of everywhere thick subtrees. We measure sets of infinite branches with the uniform probability measure μ on $\{0,1\}^\omega$.

▶ **Definition 14** (Everywhere thick sets of nodes). For every set $T \subseteq \{0,1\}^*$ of nodes denote $\vec{T}$ the set of branches in $\{0,1\}^\omega$ whose every prefix belongs to T. Then T is *everywhere thick* if starting from every node $n \in T$ there is nonzero probability to stay in T, i.e. if $\mu\left(\vec{T} \cap n\{0,1\}^\omega\right) > 0$.

Everywhere thick subtrees are almost everywhere.

▶ **Lemma 15.** *Let $P \subseteq \{0,1\}^\omega$ be a measurable set of infinite branches. Assume $\mu(P) > 0$. Then there exists an everywhere thick subtree T, with root ϵ such that $\vec{T} \subseteq P$.*

The proof relies on the inner-regularity of μ, so that P can be assumed to be a closed set, i.e. a subtree from which we can prune leaves whose subtree has probability 0.

Positive witnesses. Positive witnesses can be used to check whether a strategy is positively winning:

▶ **Definition 16** (Positive plays and witnesses). Let t be a Σ-labelled binary tree and σ a positional strategy of Eve in the acceptance game of t. Let Z be the set of σ-reachable vertices whose state is in $F_{>0}$.

A play is positive if all vertices it visits belong to $\{0,1\}^* \times F_{>0}$. A positive witness for σ is a pair (W,E) where: $W \subseteq Z$ are the *active* vertices, and $E \subseteq \{0,1\}^* \times \{0,1\}$ is the set of *positive edges*, and they have the following properties.

a) From every vertex $z \in Z$ there is a positive and canonical finite σ-play starting in z which reaches a vertex in W or a transient vertex.

b) Let $z = (n, q) \in W$. Then $(n, 0) \in E$ or $(n, 1) \in E$, or both. If $z \to z'$ is a local transition then $z' \in W$ as well whenever ($q \in Q_E$ and $z \to z'$ is consistent with σ) or ($q \in Q_A$ and $z \to z'$ is canonical). If z is controlled by Eve and $\sigma(z)$ is a split transition $q \to (q_0, q_1)$ then $((n, 0) \in E \implies (n0, q_0) \in W)$ and $((n, 1) \in E \implies (n1, q_1) \in W)$.

c) The set of nodes $\{nd \in \{0, 1\}^* \mid (n, d) \in E\}$ is everywhere thick.

▶ **Lemma 17** (Characterization of positively winning strategies)**.** *Assume the automaton has limited choice for Adam. A positional strategy σ for Eve is positively winning iff there exists a positive witness for σ.*

4.4 Deciding emptiness

A Σ-labelled binary tree t and a positional strategy σ in the corresponding acceptance game generate a tree $T_{t,\sigma} : \{0, 1\}^* \to (Q \cup Q \times Q)^{Q_E}$.

For every vertex (n, q) controlled by Eve, if $\sigma(n, q)$ is a local transition $q \to_{t(n)} q'$ then $T_{t,\sigma}(n)(q) = q'$ and if $\sigma(n, q)$ is a split transition $q \to_{t(n)} (q_0, q_1)$ then $T_{t,\sigma}(n)(q) = (q_0, q_1)$.

▶ **Definition 18** (Strategic tree)**.** A tree $T : \{0, 1\}^* \to (Q \cup Q \times Q)^{Q_E}$ is *strategic* if there exists a tree $t : \{0, 1\}^* \to \Sigma$ and a positional strategy σ for Eve such that $T = T_{t,\sigma}$.

We are interested in the strategic trees associated to winning strategies. The rest of the section is dedicated to the proof of the following theorem.

▶ **Theorem 19.** *Fix an alternating nonzero automata with limited choice for Adam. The language of strategic trees $T_{t,\sigma}$ such that σ wins the acceptance game of t can be recognized by a non-deterministic nonzero automaton of size exponential in $|Q|$. If $F_\forall = Q$ in the alternating automaton, then the sure condition of the non-deterministic automaton is Büchi.*

Proof. The characterizations of surely, almost-surely and positively winning strategies given in lemmas 12, 13 and 17 can be merged as follows.

▶ **Corollary 20.** *Let σ be a positional strategy σ for Eve. For every branch b denote $M(b) = \{\max\{q \mid (k, q) \in R^\infty(b)\} \mid k \in 0 \dots 2|Q|\}$.*

Then σ is winning if and only if for every branch b, $M(b) \subseteq F_\forall$; for almost-every branch b, $M(b) \subseteq F_1$; and there exists a positive witness for σ.

First of all, the non-deterministic automaton $\mathcal{B}$ checks whether the input tree is a strategic tree, for that it guesses on the fly the input tree $t : \{0, 1\}^* \to \Sigma$ by guessing on node n the value of $t(n)$ and checking that for every $q \in Q_E$, $q \to_{t(n)} T(n)(q)$ is a transition of the automaton.

On top of that $\mathcal{B}$ checks the three conditions of Corollary 20. For the first two conditions, it computes (asymptotically) along every branch b the value of $R^\infty(b)$ and thus of $M(b)$. For that the automaton relies on a Last Appearance Record memory (LAR) [10] whose essential properties are:

▶ **Lemma 21** (LAR memory [10])**.** *Let C be a finite set of symbols. There exists a deterministic automaton on C called the* LAR memory *on C with the following properties. First, the set of states, denoted Q, has size $\leq |C|^{|C|+1}$ and is totally ordered. Second, for every $u \in C^\omega$ denote $L^\infty(u)$ the set of letters seen infinitely often in u and $\mathrm{LAR}(u)$ the largest state seen infinitely often during the computation on u. Then $L^\infty(u)$ can be inferred from $\mathrm{LAR}(u)$, precisely there is a mapping $\phi : Q \to 2^C$ such that: $\forall u \in C^\omega, L^\infty(u) = \phi(\mathrm{LAR}(u))$.*

In order to compute $R^\infty(b)$ along a branch b, the non-deterministic automaton $\mathcal{B}$ computes deterministically on the fly the σ-index of the current node n, as defined in Lemma 13, and implements a LAR memory on the alphabet $C = \{0, \dots, 2|Q|\} \times (Q \cup \{\bot\})$.

When visiting node n, $\mathcal{B}$ injects into the LAR memory all pairs $(\sigma(q), q)$ such that $q \in Q$ and $\sigma(q) \neq \infty$ plus all pairs $(k, \bot)$ such that $k \notin \sigma(n)(Q)$. For every branch b, the set $R^\infty(b)$ is equal to all pairs (k, q) seen infinitely often such that $(k, \bot)$ is seen only finitely often. Thus, the LAR memory can be used to check the first two conditions of Corollary 20, more details are given at the end of the proof.

For now, we describe how the non-deterministic automaton $\mathcal{B}$ checks whether there exists a positive witness (W, E) (Definition 16). Denote by Z the set of σ-reachable vertices whose state is in $F_{>0}$. On node n the automaton guesses (resp. computes) the vertices of W (resp. Z) of the current node and guesses the elements of E by storing three sets of states: $W_n = \{q \in Q \mid (n, q) \in W\}$; $Z_n = \{q \in F_{>0} \mid \sigma(n, q) < \infty\}$; and $E_n = \{b \in \{0, 1\} \mid (n, b) \in E\}$.

Then $\mathcal{B}$ checks conditions a), b) and c) in the definition of a positive witness as follows.

$\mathcal{B}$ checks condition a) in the definition of a positive witness by guessing on the fly for every vertex in Z a canonical positive σ-play to a vertex which is either transient or in W, in which case we say the canonical positive play *terminates*.

For that $\mathcal{B}$ maintains an ordered list P_n of states. On the root node, P_ϵ is $Z_\epsilon \setminus W_\epsilon$. When the automaton performs a transition, it guesses for each state q in P_n and direction b_q a successor s_q, such that (nb_q, s_q) can be reached from (q, n) by a positive canonical σ-play. In direction b, every state q for which $b_q \neq b$ is removed from the list, while every state q for which $b_q = b$ is replaced by the corresponding s_q. Then all states in Z_{nb} are added at the end of the list. In case of duplicates copies of the same state in the list, only the first copy is kept. In case the head of the list is in W_{nb} or is transient, a Büchi condition is triggered and the head is moved at the back of the list. Finally, all entries of the list which are in W_{nb} are removed.

This way, condition a) holds iff the Büchi condition is triggered infinitely often on every branch. We discuss below how to integrate this Büchi condition in the sure accepting condition of the automaton.

$\mathcal{B}$ checks condition b) in the definition of a positive witness by entering an absorbing error state as soon as

1) there is some local transition $(n, q) \to_{t(n)} (n, q')$ such that $q \in W_n$ and ($q \in Q_E$ and $z \to z'$ is consistent with σ) or ($q \in Q_A$ and $z \to z'$ is canonical); or

2) there is some $q \in W_n$ controlled by Eve and $b \in E_n$ such that $\sigma(n, q)$ is a split transition $q \to_{t(n)} (q_0, q_1)$ but $q_b \notin W_{nb}$.

The guessed sets W_n are bound to satisfy condition 1) and condition 2) is checked by storing a subset of Q.

$\mathcal{B}$ checks condition c) in the definition of a positive witness by triggering the positive acceptance condition whenever it moves in direction b on a node n such that $b \in E_n$.

The sure and almost-sure acceptance condition are defined as follows. The Büchi condition necessary for checking condition a) in the definition of a positive witness is integrated in the LAR memory, for that we add to the alphabet C of the LAR memory a new symbol $\top$ which is injected in the LAR memory whenever the Büchi condition is triggered. The order between states of $\mathcal{B}$ is induced by the order of the LAR memory.

This way, according to Lemma 21, the largest state seen infinitely often along a branch b reveals whether $\top$ was seen infinitely often, and reveals the value of $R^\infty(b)$ (the set of pairs (k, q) seen infinitely often such that $(k, \bot)$ was seen finitely often) hence of $M(b)$ as well. The state is surely (resp. almost-surely) accepting iff $\top$ was seen infinitely often and $M(b) \subseteq F_\forall$

(resp. $M(b) \subseteq F_1$). In case $F_\forall = Q$ in the alternating automaton then the sure condition boils down to the Büchi condition.

According to Corollary 20, and by construction of $\mathcal{B}$, the computation of $\mathcal{B}$ is accepting iff the input is a strategic tree whose corresponding strategy of Eve is winning. ◀

▶ **Theorem 22.** *Emptiness of alternating nonzero automata with limited choice for Adam is decidable in* NEXPTIME∩*co*-NEXPTIME. *If $F_\forall = Q$, emptiness can be decided in* EXPTIME.

Proof. Emptiness of an alternating automaton reduces to the emptiness of a non-deterministic automaton of exponential size. This non-deterministic automaton guesses on-the-fly a tree $\{0,1\}^* \to (Q \cup Q \times Q)^{Q_E}$ and checks it is a winning strategic tree, using the automaton given by Theorem 19. In case the alternating automaton is $F_\forall$-trivial, the sure condition of the non-deterministic automaton is Büchi (Theorem 19). We conclude with Theorem 3. ◀

5 Satisfiability of CTL*$[\exists, \forall, \mathbb{P}_{>0}, \mathbb{P}_{=1}]$

Our result on alternating nonzero automata can be applied to decide the satisfiability of the logic CTL*$[\exists, \forall, \mathbb{P}_{>0}, \mathbb{P}_{=1}]$, a generalization of CTL* which integrates both deterministic and probabilistic state quantifiers.

Markov chains. The models of CTL*$[\exists, \forall, \mathbb{P}_{>0}, \mathbb{P}_{=1}]$ formulas are Markov chains. A Markov chain with alphabet Σ is a tuple $\mathcal{M} = (S, p, t)$ where S is the (countable) set of *states*, $p : S \to \Delta(S)$ are the *transition probabilities* and $t : S \to \Sigma$ is the *labelling function*.

For every state $s \in S$, there is a unique probability measure denoted $\mathbb{P}_{\mathcal{M},s}$ on S^ω such that $\mathbb{P}_{\mathcal{M},s}(sS^\omega) = 1$ and for every sequence $s_0 \cdots s_n s_{n+1} \in S^*$, $\mathbb{P}_{\mathcal{M},s}(s_0 \cdots s_n s_{n+1} S^\omega) = p(s_n, s_{n+1}) \cdot \mathbb{P}_{\mathcal{M},s}(s_0 s_1 \cdots s_n S^\omega)$. When $\mathcal{M}$ is clear from the context this probability measure is simply denoted $\mathbb{P}_s$. A *path* in $\mathcal{M}$ is a finite or infinite sequence of states $s_0 s_1 \cdots$ such that $\forall n \in \mathbb{N}, p(s_n, s_{n+1}) > 0$ .. We denote $\mathrm{Path}_\mathcal{M}(s_0)$ the set of such paths.

A binary tree $t : \{0,1\}^* \to \Sigma$ is seen as a specific type of Markov chain, where from every node $n \in \{0,1\}^*$ there is equal probability $\frac{1}{2}$ to perform transitions to $n0$ or $n1$.

Syntax. For a fixed alphabet Σ, there are two kinds of formula: state formula (typically denoted ψ) and path formula (denoted ϕ), generated by the following grammar:

$$\psi ::= \top \mid \bot \mid a \in \Sigma \mid \psi \wedge \psi \mid \psi \vee \psi \mid \neg\psi \mid \exists\phi \mid \forall\phi \mid \mathbb{P}_{>0}(\phi) \mid \mathbb{P}_{=1}(\phi)$$
$$\phi ::= \psi \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid X\phi \mid \phi U \phi \mid G\phi \ .$$

Semantics. Let $\mathcal{M} = (S, t, p)$ a Markov chain. We define simultaneously and inductively the satisfaction $\mathcal{M}, s \models \psi$ of a state formula ψ by a state $s \in S$ and the satisfaction $\mathcal{M}, w \models \phi$ of a path formula ϕ by a path $w \in \mathrm{Path}_\mathcal{M}$. When $\mathcal{M}$ is clear from the context, we simply write $s \models \psi$ and $w \models \phi$.

If a state formula is produced by one of the rules $\top \mid \bot \mid p \mid \psi \wedge \psi \mid \psi \vee \psi \mid \neg\psi$, its satisfaction is defined as usual. If ϕ is a path formula and $\psi \in \{\exists\phi, \forall\phi, \mathbb{P}_{>0}(\phi), \mathbb{P}_{=1}(\phi)\}$ then

$$\begin{aligned} s &\models \exists\phi && \text{if } \exists w \in \mathrm{Path}_\mathcal{M}(s), w \models \phi \\ s &\models \forall\phi && \text{if } \forall w \in \mathrm{Path}_\mathcal{M}(s), w \models \phi \\ s &\models \mathbb{P}_{\sim b}(\phi) && \text{if } \mathbb{P}_{\mathcal{M},s}(w \in \mathrm{Path}_\mathcal{M}(s) \mid w \models \phi) \sim b \ . \end{aligned}$$

The satisfaction of a path formula ϕ by an infinite path $w = s_0 s_1 \cdots \in \mathrm{Path}_\mathcal{M}(s_0)$ is defined as follows. If ϕ is produced by one of the rules $\neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi$ then its satisfaction

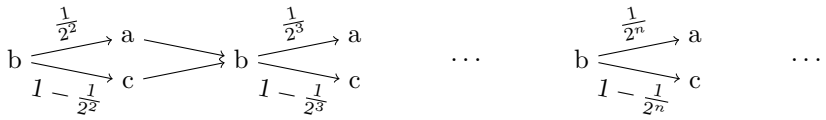

Figure 1 A model of $(\forall(G\exists(\top U a))) \wedge (\mathbb{P}_{>0}(G\neg a))$.

is defined as usual. If ϕ is a state formula (rule $\phi := \psi$) then $w \models \psi$ if $s_0 \models \psi$. Otherwise, $\phi \in \{X\phi', G\phi', \phi_1 U \phi_2\}$ where ϕ', ϕ_1 and ϕ_2 are path formulas. For every integer k, we denote $w[k]$ the path $s_k s_{k+1} \cdots \in \mathrm{Path}_{\mathcal{M}}(s_k)$. Then:

$$
\begin{array}{ll}
w \models X\phi' & \text{if } w[1] \models \phi' \\
w \models G\phi' & \text{if } \forall i \in \mathbb{N}, w[i] \models \phi' \\
w \models \phi_1 U \phi_2 & \text{if } \exists n \in \mathbb{N}, (\forall 0 \leq i < n, w[i] \models \phi_1 \wedge w[n] \models \phi_2).
\end{array}
$$

The Markov chain given in Figure 1 satisfies the formula $(\forall(G\exists(\top U a))) \wedge (\mathbb{P}_{>0}(G\neg a))$.

A formula for PUCE trees. The language of PUCE trees introduced in Section 3 can be described by the following CTL$^*[\exists, \forall, \mathbb{P}_{>0}, \mathbb{P}_{=1}]$ formula:

$$\forall G(\mathbb{P}_{>0}(\top U(G\neg a)) \wedge \mathbb{P}_{>0}(\top U(G\neg b))) \ .$$

Variants and fragments. A formula of CTL$^*[\exists, \forall, \mathbb{P}_{>0}, \mathbb{P}_{=1}]$ belongs to the fragment CTL if in each of its state subformula ψ of type $\exists\phi \mid \forall\phi \mid \mathbb{P}_{>0}(\phi) \mid \mathbb{P}_{=1}(\phi)$ the path formula ϕ has type $X\psi' \mid \psi' U \psi'' \mid G\psi'$ where ψ' and ψ'' are state subformulas.

In the variant ECTL, every path formula ϕ is described as the composition of a deterministic Büchi automata on some alphabet $\{0,1\}^k$ with k state subformulas. A path satisfies ϕ if the Büchi automaton accepts the sequence of letters obtained by evaluating the k state subformulas on every state along the path. This variant augments both the expressivity and the conciseness of the logic at the cost of a less intuitive syntax. For more details see [6].

We are also interested in the fragments where the operators $\exists$ and $\forall$ are not used, i.e. the qualitative fragments of the logics pCTL*, pECTL and pCTL.

Satisfiability problem. A Markov chain $\mathcal{M}$ *satisfies* a formula ξ at state s, or equivalently $(\mathcal{M}, s)$ *is a model* of ξ, if $\mathcal{M}, s \models \xi$. We are interested in the problem:
MC-SAT: given a formula, does it have a model?

This logic is an extension of monadic second-order logic on infinite binary trees with a new probabilistic operator [14, 15, 2]. The satisfiability of this logic is reducible to the emptiness problem for nonzero non-deterministic automata [2] which is decidable [3]. Since CTL$^*[\exists, \forall, \mathbb{P}_{>0}, \mathbb{P}_{=1}]$ is a fragment of TMSO+ZERO, this result implies that the satisfiability of CTL$^*[\exists, \forall, \mathbb{P}_{>0}, \mathbb{P}_{=1}]$ is decidable with non-elementary complexity. The reduction to the emptiness of alternating nonzero automata given in the present paper provides a better complexity bound.

▶ **Theorem 23.** *For CTL$^*[\exists, \forall, \mathbb{P}_{>0}, \mathbb{P}_{=1}]$ the satisfiability problem is in 3-*NEXPTIME ∩ *co-3-*NEXPTIME*. The following table summarizes complexities of the satisfiability problem for various fragments and variants of CTL$^*[\exists, \forall, \mathbb{P}_{>0}, \mathbb{P}_{=1}]$:*

	CTL*	*ECTL*	*CTL*
$[\exists, \forall, \mathbb{P}_{>0}, \mathbb{P}_{=1}]$	*3-*NEXPTIME ∩ *co-3-*NEXPTIME	*2-*NEXPTIME ∩ *co-2-*NEXPTIME	NEXPTIME ∩ *co-*NEXPTIME
$[\mathbb{P}_{>0}, \mathbb{P}_{=1}]$	*3-*EXPTIME *[6] (qualitative pCTL*)*	*2-*EXPTIME *[6] (qualitative pECTL)*	EXPTIME *[5] (qualitative pCTL)*

According to [5, 6], the complexities for ECTL$[\mathbb{P}_{>0}, \mathbb{P}_{=1}]$ and CTL$[\mathbb{P}_{>0}, \mathbb{P}_{=1}]$ are optimal.

The first step in the proof of Theorem 23 is a linear-time reduction from **MC-SAT** to: **BIN-SAT:** given a formula, does it have a model among binary trees?

▶ **Theorem 24.** *Any formula ξ of CTL$^*[\exists, \forall, \mathbb{P}_{>0}, \mathbb{P}_{=1}]$ on alphabet Σ can be effectively transformed into a formula ξ' of linear size on alphabet $\Sigma \cup \{\circ\}$ such that ξ is **MC-SAT** iff ξ' is **BIN-SAT**. As a consequence, **MC-SAT** linearly reduces to **BIN-SAT**. This transformation preserves the fragment CTL$^*[\mathbb{P}_{>0}, \mathbb{P}_{=1}]$.*

The second step is a standard translation from logic to alternating automata [12].

▶ **Lemma 25.** *For every formula ξ of CTL$^*[\exists, \forall, \mathbb{P}_{>0}, \mathbb{P}_{=1}]$ (resp. ECTL$[\exists, \forall, \mathbb{P}_{>0}, \mathbb{P}_{=1}]$), there is an alternating automaton $\mathcal{A}$ with limited choice for Adam whose language is the set of binary trees satisfying the formula at the root. The automaton is effectively computable, of size $O(2^{2^{|\xi|}})$ (resp. $\mathcal{O}(2^{|\xi|})$). If ξ is a CTL formula, the size of $\mathcal{A}$ is $\mathcal{O}(|\xi|)$. In case the formula does not use the $\exists$ and $\forall$ operators, the $F_\forall$ condition is trivial i.e. $F_\forall = Q$.*

Proof of Theorem 23. All the complexity results are obtained by reduction of **MC-SAT** to the emptiness problem for an alternating nonzero automaton with limited choice for Adam, which is decidable in NEXPTIME∩co-NEXPTIME (Theorem 22). The size of the automaton varies from doubly-exponential to linear size depending on whether the formula is in CTL*, ECTL or CTL (Lemma 25). In case the formula does not use the deterministic operators $\exists$ and $\forall$ (i.e. for qualitative pCTL*, pECTL and pCTL) the $F_\forall$ condition of the alternating automaton is trivial thus its emptiness is decidable in EXPTIME (Theorem 22). ◀

Conclusion

We have introduced the class of *alternating nonzero* automata, proved decidability of the emptiness problem for the subclass of automata with limited choice for Adam and obtained as a corollary algorithms for the satisfiability of a temporal logic extending both CTL* and the qualitative fragment of pCTL*.

A natural direction for future work is to find more general classes of alternating nonzero automata with a decidable emptiness problem, which requires some more insight on the properties of the acceptance games, in particular the existence of positional strategies in the acceptance game.

References

1. Christel Baier, Marcus Größer, and Nathalie Bertrand. Probabilistic ω-automata. *J. ACM*, 59(1):1, 2012.
2. Mikołaj Bojańczyk. Thin MSO with a probabilistic path quantifier. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, pages 96:1–96:13, 2016.
3. Mikołaj Bojańczyk, Hugo Gimbert, and Edon Kelmendi. Emptiness of zero automata is decidable. *CoRR*, abs/1702.06858, 2017. URL: `http://arxiv.org/abs/1702.06858`.
4. Mikołaj Bojańczyk, Hugo Gimbert, and Edon Kelmendi. Emptiness of zero automata is decidable. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 106:1–106:13, 2017.
5. Tomás; Brázdil, Vojtech Forejt, Jan Kretínský, and Antonín Kucera. The satisfiability problem for probabilistic CTL. In *Proc. of LICS*, pages 391–402, 2008.

6 Tomáš Brázdil, Vojtěch Forejt, and Antonín Kučera. Controller synthesis and verification for markov decision processes with qualitative branching time objectives. *Automata, Languages and Programming*, pages 148–159, 2008.

7 Arnaud Carayol, Axel Haddad, and Olivier Serre. Randomization in automata on infinite trees. *ACM Trans. Comput. Log.*, 15(3):24:1–24:33, 2014.

8 Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. Alternation. *J. ACM*, 28(1):114–133, January 1981.

9 E Allen Emerson and Joseph Y Halpern. Sometimes and not never revisited: on branching versus linear time temporal logic. *Journal of the ACM (JACM)*, 33(1):151–178, 1986.

10 Yuri Gurevich and Leo Harrington. Trees, automata, and games. In *Proceedings of STOC'82*, pages 60–65, New York, NY, USA, 1982. ACM.

11 Hans Hansson and Bengt Jonsson. A logic for reasoning about time and reliability. *Formal aspects of computing*, 6(5):512–535, 1994.

12 Orna Kupferman, Moshe Y Vardi, and Pierre Wolper. An automata-theoretic approach to branching-time model checking. *Journal of the ACM (JACM)*, 47(2):312–360, 2000.

13 Daniel Lehmann and Saharon Shelah. Reasoning with time and chance. *Information and Control*, 53(3):165–1983, 1982.

14 Henryk Michalewski and Matteo Mio. Measure quantifier in monadic second order logic. In *Proc. of LFCS 2016*, pages 267–282, 2016. doi:10.1007/978-3-319-27683-0_19.

15 Henryk Michalewski, Matteo Mio, and Mikołaj Bojańczyk. On the regular emptiness problem of subzero automata. In *Proc. of ICE 2016, Heraklion, Greece, 8-9 June 2016.*, pages 1–23, 2016.

16 David E. Muller and Paul E. Schupp. Alternating automata on infinite trees. *Theoretical Computer Science*, 54(2):267 – 276, 1987. doi:http://dx.doi.org/10.1016/0304-3975(87)90133-2.

17 A. Paz. *Introduction to probabilistic automata.* Academic Press, 1971.

18 M. O. Rabin. Probabilistic automata. *Information and Control*, 6(3):230–245, 1963.

19 Moshe Y Vardi and Larry Stockmeyer. Improved upper and lower bounds for modal logics of programs. In *Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pages 240–251. ACM, 1985.