Emptiness of Zero Automata Is Decidable*†

Mikołaj Bojańczyk[1], Hugo Gimbert[2], and Edon Kelmendi[2]

1 **Institute of Informatics, University of Warsaw, Warsaw, Poland**
`bojan@mimuw.edu.pl`
2 **LaBRI, Université de Bordeaux, CNRS, Bordeaux, France**
`hugo.gimbert@labri.fr`
3 **LaBRI, Université de Bordeaux, CNRS, Bordeaux, France**
`edon.kelmendi@labri.fr`

Abstract

Zero automata are a probabilistic extension of parity automata on infinite trees. The satisfiability of a certain probabilistic variant of MSO, called TMSO + ZERO, reduces to the emptiness problem for zero automata. We introduce a variant of zero automata called nonzero automata. We prove that for every zero automaton there is an equivalent nonzero automaton of quadratic size and the emptiness problem of nonzero automata is decidable, with complexity co-NP. These results imply that TMSO + ZERO has decidable satisfiability.

1998 ACM Subject Classification F.4.3 Formal Languages, F.4.1 Mathematical Logic

Keywords and phrases tree automata, probabilistic automata, monadic second-order logic

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.106

1 Introduction

In this paper, we prove that emptiness is decidable for two classes of automata, namely *zero* and *nonzero* automata. Zero automata were introduced as a tool for recognizing models of a probabilistic extension of MSO on infinite trees [1]. Nonzero automata, introduced in this paper, are equivalent to zero automata, but have simpler semantics.

Both zero and nonzero automata are probabilistic extensions of parity automata on infinite trees. Here we focus on the case of binary trees. The automaton performs a random walk on the infinite binary input tree: when the automaton is in a state q on a node labelled with a, it selects non-deterministically a transition (q, a, r_0, r_1) and moves with equal probability $\frac{1}{2}$ either to the left node in state r_0 or to the right node in state r_1.

The set of branches of the infinite binary tree is equipped with the uniform probability measure, which is used to define the acceptance condition. There are two variants of the acceptance condition, one for zero automata and one for nonzero automata

A *nonzero* automaton is equipped with a total order $\leq$ on its set of states Q and three accepting subsets of states $F_\forall, F_1$ and $F_{>0}$. A run is accepting if:

(a) on every branch the limsup state (i.e. the maximal state seen infinitely often) is in $F_\forall$;
(b) and with probability 1 the limsup state is in F_1;
(c) and every time the run visits a state in $F_{>0}$ there is a nonzero probability that all subsequent states are in $F_{>0}$.

* Full version with proof is [2], `http://arxiv.org/abs/1702.06858`.

† The research of M. Bojańczyk is supported by the ERC grant LIPA under the Horizon 2020 framework. H. Gimbert and E. Kelmendi are supported by the French ANR project "Stoch-MC" and "LaBEX CPU" of Université de Bordeaux.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 106; pp. 106:1–106:13

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Condition (a) is the classical parity condition for tree automata and condition (b) is equivalent to the qualitative condition from [5]. Condition (c) seems to be new. Conditions (a) and (b) are used to define the acceptance condition of zero automata as well, the difference between zero and nonzero automata lies in condition (c).

The paper [1] introduced a variant of MSO on infinite trees with a probabilistic quantifier, called TMSO+zero, inspired by probabilistic MSO from [9]. In the case where zero is the unary predicate which checks whether a set of branches has probability 0, the contribution of [1] was a proof that for every formula of this logic one can compute a zero automaton which accepts the same trees. The logic is powerful enough to formulate properties like "every node in the tree has a descendant node labelled with b and the set of branches with infinitely many b has probability 0". As argued in [1], the motivation for this logic is twofold. First, it extends various probabilistic logics known in the literature, e.g. qualitative probabilistic CTL* [8], or qualitative probabilistic CTL* extended with ω-regular path properties [3]. Second, the logic, although less general than MSO, represents a robust class of languages of infinite trees that goes beyond classical MSO, and thus falls under the scope of the programme of searching for decidable extensions of MSO.

The emptiness problem for zero automata was not solved in [1], thus leaving open the logic's decidability. A step toward an emptiness algorithm was made in [10], where it was shown that for subzero automata – the special case of zero automata where only conditions (a) and (b) are used – one can decide if the recognized language contains a regular tree. In this paper we prove that zero and nonzero automata have decidable emptiness, and therefore also the logic from [1] has decidable satisfiability.

The main results of this paper are:

(i) For every zero automaton there is an equivalent nonzero automaton of quadratic size.
(ii) A nonzero automaton with $F_\forall = Q$ is nonempty if and only if its language contains a regular tree of size $|Q|$. This is decidable in NP.
(iii) The emptiness problem of nonzero automata is in co-NP.

To prove (iii) we provide a reduction of the emptiness problem to the computation of the winner of a parity game called the *jumping game*. For that we rely on (ii): the states of the jumping game are regular runs of a nonzero automaton where $F_\forall = Q$. According to (i) the emptiness problem for zero automata is in co-NP as well.

These results were recently improved: the emptiness problem is actually in NP$\cap$co-NP, and even in PTIME if $F_\forall = Q$, see [2].

The plan of the paper is as follows. In Section 2 we introduce zero and nonzero automata and state our main result (iii) (Theorem 3). In Section 3 we show (i) (Lemma 5). In Section 4 we focus on the special case where $Q = F_\forall$ and show (ii) (Theorem 10). In Section 5 we introduce jumping games and combine the previous results to provide a proof of (iii).

2 Zero and nonzero automata

This section introduces trees and nonzero and zero automata.

Trees, branches and subtrees. The automata of this paper describe properties of infinite binary labelled trees. A node in a tree is a sequence in $\{0,1\}^*$. A *tree* over an alphabet Σ is a function $t : \{0,1\}^* \to \Sigma$. We use standard terminology for trees: node, root, left child, right child, leaf, ancestor and descendant. A *branch* is a sequence in $\{0,1\}^\omega$, viewed as an infinite sequence of left or right turns. A branch *visits* a node if the node is a prefix of the branch.

A *subtree* is a non-empty and ancestor-closed set of nodes. A subtree is *leaf-free* if each of its nodes has at least one child in the subtree. A branch of a subtree is a branch which visits only nodes of the subtree.

Probability measure over branches. We use the *coin-flipping* measure on $\{0,1\}^\omega$: each bit is chosen independently at random, with 0 and 1 having equal probability, and every Borel subset of $\{0,1\}^\omega$ is measurable. The probability of a subtree is the probability of the set of branches of the subtree. The inner regularity of the coin-flipping measure (see e.g. [7, Theorem 17.10]) implies:

▶ **Lemma 1.** *The probability of a measurable set E is the supremum of the probabilities of the subtrees whose every branch belongs to E.*

Nonzero automata. Intuitively, a nonzero automaton is a nondeterministic parity tree automaton which has the extra ability to check whether the set of branches satisfying the parity condition has zero or nonzero probability.

▶ **Definition 2.** The syntax of a nonzero automaton is a tuple

$$\underbrace{Q}_{\text{states}} \qquad \underbrace{\Sigma}_{\text{input alphabet}} \qquad \underbrace{\Delta \subseteq Q \times \Sigma \times Q^2}_{\text{transitions}},$$

with all components finite, together with a total order $\leq$ on Q and three subsets

$$F_\forall, F_1, F_{>0} \subseteq Q \ .$$

A *run* of the automaton on an input tree $t : \{0,1\}^* \to \Sigma$ is an infinite binary tree $r : \{0,1\}^* \to Q$ whose root is labelled by the maximal state of Q, also called the *initial* state and which is consistent with the transition relation in the usual sense, i.e. $\forall v \in \{0,1\}^*, (r(v), t(v), r(v0), r(v1)) \in \Delta$. Define the *limsup* of a branch of the run to be the maximal state that appears infinitely often on the branch.

The run is *accepting* if it is surely, almost-surely and nonzero accepting:

- **surely accepting:** every branch has limsup in $F_\forall$; and
- **almost-surely accepting:** the set of branches with limsup in F_1 has probability 1; and
- **nonzero accepting:** for every node v with state in $F_{>0}$, the set of branches which visit v and visit only $F_{>0}$-labelled nodes below v has nonzero probability.

The emptiness problem. The emptiness problem asks whether an automaton has an accepting run. Our main result:

▶ **Theorem 3.** *Emptiness of a nonzero automaton is decidable in co-*NP.

Proof. This is a corollary of a series of intermediary results. In section 4 we focus on the special case where $F_\forall = Q$ (Theorem 10). In section 5 we reduce the emptiness problem for nonzero automata to the computation of the winner in a parity game called the *jumping game* (Lemma 17) and give an NP algorithm to compute the winner of the jumping game (Lemma 18). ◀

Zero automata. Nonzero automata are a variant of *zero automata* introduced in [1]. A zero automaton differs slightly from a nonzero automaton in that it uses a notion of "seed state" for the nonzero acceptance condition. On top of $F_\forall, F_1$ and $F_{>0}$ there is a subset $Q_{\text{seed}} \subseteq Q$. A run is accepting if it is surely, almost-surely and *zero* accepting:

- **zero accepting:** for every node v with state $q \in Q_{\text{seed}}$, with nonzero probability the run visits node v, then below v visits only states $\leq q$ and moreover has limsup in $F_{>0}$.

In the next section, we show that every zero automaton can be transformed in an equivalent nonzero automaton of quadratic size (Lemma 5). Combined with Theorem 3,

▶ **Corollary 4.** *The emptiness problem of zero automata is decidable in co-*NP*.*

According to [1], this implies that TMSO + zero has decidable satisfiability when zero is the unary predicate checking that a set of branches has probability 0.

An example: the dense but not very dense language. A tree over alphabet $\{a, b\}$ is *dense but not very dense* if:

1. every node has a descendant with label a; and
2. there is zero probability that a branch visit infinitely many nodes with letter a.

This language is non-empty, contains no regular tree and is recognized by a nonzero automaton. This automaton has three states, totally ordered as follows:

$$\underbrace{s}_{\text{searching for } a} < \underbrace{n}_{\text{not searching for } a} < \underbrace{f}_{\text{just found } a} .$$

The automaton begins in state f in the root. When the automaton reads a node with label b, then it sends s to some child and n to the other child, regardless of its current state. Choosing which child gets s and which child gets n is the only source of nondeterminism in this automaton. When the automaton sees letter a, it sends f to both children regardless of its current state. The acceptance condition is:

$$F_\forall = \{n, f\} \qquad F_1 = \{n\} \qquad F_{>0} = \emptyset \ .$$

3 From zero to nonzero automata

In this section we show that nonzero automata are as expressive as zero automata.

▶ **Lemma 5.** *For every zero automaton one can compute a nonzero automaton of quadratic size which accepts the same trees.*

The rest of the section is dedicated to the proof of Lemma 5, which is a direct corollary of Lemma 7 and Lemma 8 below.

Without loss of generality, we assume that in every *zero* automaton $F_{>0} \subseteq F_1 \subseteq F_\forall$. Changing F_1 for $F_1 \cap F_\forall$ and $F_{>0}$ for $F_{>0} \cap F_1$ does not modify the set of accepting runs of a zero automaton, since all branches should have limsup in $F_\forall$ and if the limsup is equal with nonzero probability to some $q \in F_{>0}$ then necessarily $q \in F_1$. By contrast, for *nonzero* automata there is no obvious reason for the same remark to hold.

We make use of an intermediary acceptance condition. Let r be a run. We say that a path from a node v to a node w is *seed-consistent* if whenever the path visits a seed state s, subsequent states are $\leq s$.

- **Strong zero acceptance condition:** for every node v labelled by a seed state, there is a seed-consistent path from v to a strict descendant w of v such that the state $r(w)$ of w is in $F_{>0}$ and there is a nonzero probability that the run
  - visits node w and visits only states $\leq r(w)$ below w,
  - has limsup $r(w)$,
  - in case $r(w) \notin Q_{\text{seed}}$, visits no seed state below w,
  - in case $r(w) \in Q_{\text{seed}}$, visits no seed state other than $r(w)$ below w.

Actually, the strong zero and zero acceptance conditions coincide (proof in appendix):

▶ **Lemma 6.** *A run is zero accepting if and only if it is strongly zero accepting.*

Construction of the nonzero automaton. Intuitively, every *zero* automaton can be simulated by a *nonzero* automaton which guesses on the fly a run of the zero automaton and checks simultaneously that the guessed run is strongly zero accepting. Whenever the automaton visits a node v with a seed state then it enters in the next step a *path-finding state* and guesses a seed-consistent path to a node w which is a witness of the strong zero condition. Once on the node w the automaton enters a *subtree-guessing* state and starts guessing a leaf-free subtree of the run, whose nodes are labelled by states $\leq r(w)$, whose branches have limsup $r(w)$ and which has nonzero probability.

There are some verifications to do in order to certify that the guessed run is strongly zero accepting. The surely accepting condition is used to prevent the automaton from staying forever in the path-finding mode and also to check that every branch of the subtree has limsup $r(w)$. The nonzero condition is used to check that the subtree has nonzero probability. To perform these verifications, the nonzero automaton stores some data in its control state. In path-finding mode the automaton records the smallest seed state seen so far in order to check on-the-fly that the path from v to w is seed-consistent. In subtree-guessing mode the automaton keeps track of the state $r(w)$.

The set of states of this automaton is denoted R, every state in R has as a first component a control state Q of the zero automaton. Precisely, R is the union of three sets:

- **normal states:** Q,
- **path-finding states:** $\{(q, s) \mid q \in Q, s \in Q_{\text{seed}}, q \leq s\}$,
- **subtree-guessing states:** $\{(q, f, *) \mid q \in Q, f \in F_{>0}, q \leq f, (q \notin Q_{\text{seed}} \vee q = f)\}$.

We equip R with any order $\prec$ such that

- the projection on the first component $\Pi_1 : (R, \prec) \to (Q, <)$ is monotonic,
- $(q, s) \prec q$ for every $q \in Q$ and $s \in Q_{\text{seed}}$ with $q \leq s$.

The zero, almost-surely and surely accepting conditions are defined respectively as:

$$G_{>0} = \text{ the set of subtree-guessing states,}$$
$$G_1 = F_1 \cup \{(f, f, *) \mid f \in F_{>0}\},$$
$$G_\forall = F_\forall \cup \{(f, f, *) \mid f \in F_{>0}\} \ .$$

The transitions of the automaton can be informally described as follows. The nonzero automaton guesses on the fly a run $\rho : \{0, 1\}^* \to Q$ of the zero automaton by storing the value of $\rho(v)$ as the first component of its own control state on the node v. The nonzero automaton stays in the set of normal states as long as the run does not enter a seed state. On a node v labelled by $s \in Q_{\text{seed}}$, the nonzero automaton starts looking for a path to a descendant node w that satisfies the strong zero condition. For that in the next step the automaton enters either a path-finding or a subtree-guessing state. While in a path-finding state, the

automaton guesses on the fly a seed-consistent path. Whenever the run is in a nonzero state $f \in F_{>0}$ the nonzero automaton can enter the subtree-guessing state $(f, f, *)$, or not. While in subtree-guessing mode the second component is constant, and the automaton control state is of type $(q, f, *)$ with $q \leq f$ and $q \notin Q_{\text{seed}}$ unless $q = f \in Q_{\text{seed}}$. From a subtree-guessing state the automaton may switch back any time to a normal state.

Formally, for every transition $q \to r_0, r_1$ of the zero automaton, there is a transition

$$q' \to r'_0, r'_1$$

in the nonzero automaton if the first component of q' is q and

$$r'_0 = \begin{cases} r_0 & \text{whenever } q' \text{ is not path-finding} \\ (r_0, r_0, *) & \text{whenever } \begin{cases} q \in Q_{\text{seed}}, q' = q \text{ and } r_0 \in F_{>0} \text{ and } r_0 \leq q \\ \text{or } q' = (q, s) \text{ and } r_0 \in F_{>0} \text{ and } r_0 \leq s, \end{cases} \\ (r_0, f, *) & \text{whenever } q' = (q, f, *) \text{ and } r_0 \leq f \text{ and } (r_0 \notin Q_{\text{seed}} \vee r_0 = f). \end{cases}$$

The possible values of r'_1 are symmetric. There are also *left path-finding transitions*: for every seed states $s, s' \in Q_{\text{seed}}$ such that $q \leq s$ and $r_0 \leq s$ there are transitions

$$q' \to (r_0, s'), r_1 \text{ where } q' = \begin{cases} q \text{ or } (q, q) \text{ if } q = s \\ (q, s) \text{ otherwise} \end{cases} \quad \text{and } s' = \begin{cases} s & \text{if } r_0 \notin Q_{\text{seed}} \\ r_0 & \text{if } r_0 \in Q_{\text{seed}}. \end{cases}$$

There may also be a symmetric *right path-finding transition* $(q, s) \to r_0, (r_1, s')$ when the symmetric conditions hold.

The next two lemmas relate the accepting runs of the zero and the nonzero automata, their proofs can be found in the appendix.

▶ **Lemma 7.** *Let $d : \{0,1\}^* \to R$ be an accepting run of the nonzero automaton. Then its projection $r = \Pi_1(d)$ on the first component is an accepting run of the zero automaton.*

▶ **Lemma 8.** *If the zero automaton has an accepting run $r : \{0,1\}^* \to Q$ then the nonzero automaton has an accepting run $d : \{0,1\}^* \to R$ such that $r = \Pi_1(d)$.*

4 Emptiness of $F_\forall$-trivial automata is in NP

A run of a nonzero automaton needs to satisfy simultaneously three conditions, which correspond to the accepting sets $F_\forall, F_1, F_{>0}$. For a subset

$$I \subseteq \{F_\forall, F_1, F_{>0}\}$$

define I-automata to be the special case of nonzero automata where only the acceptance conditions corresponding to I need to be satisfied. These are indeed special cases: ignoring $F_{>0}$ can be achieved by making it empty, ignoring F_1 can be achieved by making it equal to $F_\forall$, and ignoring $F_\forall$ can be achieved by making it equal to all states Q.

Generalizing parity automata, with standard and qualitative semantics. A $\{F_\forall\}$-automaton is a parity automaton. Thus solving emptiness for nonzero automata is at least as hard as emptiness for parity automata on trees, which is polynomial time equivalent to solving parity games, in NP ∩co-NP [12] or in quasi-polynomial time [4].

A $\{F_1\}$-automaton is the same as a parity automaton with qualitative semantics as introduced in [5]. Emptiness for such automata can be solved in polynomial time using standard linear programming algorithms for Markov decision processes.

Subzero automata. A $\{F_1, F_\forall\}$-automaton is the same as a *subzero* automaton as considered in [10]. In [10], it was shown how to decide if a subzero automaton accepts some regular tree. Since some subzero automata are nonempty but accept no regular trees, see e.g. the example in [1], the result from [10] does not solve non-emptiness for subzero automata.

$F_\forall$-trivial automata. In a $\{F_1, F_{>0}\}$-automaton, the surely accepting condition is trivial, i.e. $F_\forall = Q$. We call such automata $F_\forall$-trivial. The acceptance of a run of a $F_\forall$-trivial automaton depends only on the probability measure on Q^ω induced by the run, individual branches do not matter.

▶ **Definition 9** (Positional run)**.** A run is *positional* if whenever the states of two nodes coincide then the states of their left children coincide and the states of their right children coincide.

▶ **Theorem 10.** *If a $F_\forall$-trivial automaton has an accepting run, then it has a positional accepting run. Emptiness of $F_\forall$-trivial automata is in co-*NP*.*

This result was recently improved in [2]: the complexity is actually PTIME. The proof of this theorem relies on the notion of acceptance witnesses.

▶ **Definition 11** (Transition graph and acceptance witness)**.** Let D be a set of transitions.

The transition graph of D, denoted G_D, is the directed graph whose vertices are all states appearing in one of the transitions in D, denoted Q_D, and whose edges are induced by the transitions in D: for every $(q, a, l, r) \in D$ both (q, l) and (q, r) are edges of G_D.

The set D is an *acceptance witness* if it satisfies the four following conditions:

(i) Q_D contains the initial state of the automaton and G_D has no dead-end,
(ii) the maximum of every bottom strongly connected component (BSCC) of G_D is in F_1,
(iii) every BSCC of G_D is either contained in $F_{>0}$ or does not intersect $F_{>0}$,
(iv) from every state in $F_{>0} \cap Q_D$ there is a path in $F_{>0} \cap Q_D$ to a BSCC contained in $F_{>0}$.

▶ **Lemma 12.** *If a $F_\forall$-trivial automaton has an acceptance witness, it has a positional accepting run.*

Proof. The proof is by induction on $N_D = |D| - |Q_D|$. Since G_D has no dead-end, every state in Q_D is the source of a transition in D thus $N_D \geq 0$.

If $N_D = 0$ then for every state $q \in Q_D$ there is a unique transition $\delta_q = (q, a_q, l_q, r_q)$. Let ρ be the positional run whose root has the initial state and every node with vertex $q \in Q_D$ has children l_q and r_q, which is well-defined according to property (i). We show that ρ is an accepting run. The graph G_D can be seen as a Markov chain, with probability either 1 or $\frac{1}{2}$ on every edge, depending on the out-degree. The probability measure on Q_D^ω produced by the random walk on ρ coincide with the probability measure on Q_D^ω produced by this finite Markov chain: indeed both measures coincide on finite cylinders $q_0 \cdots q_n Q_D^\omega$. Basic theory of finite homogenous Markov chain implies that almost-surely every branch of the run ends up in one of the BSCCs of G_D and visits all its states infinitely often. Thus property (ii) ensures that the run ρ is almost-surely accepting. Properties (iii) and (iv) guarantee that the run is moreover nonzero-accepting.

Assume now that $N_D > 0$. We show that there is a strictly smaller acceptance witness $D' \subsetneq D$. Let $q \in Q_D$ which is the source of several transitions in D, then D' is obtained by removing from D all these transitions except one. To choose which transition δ to keep, pick some shortest path $q = q_0 \dots q_n$ in G_D of length ≥ 1 which leads to a maximal state of one of the BSCCs of G_D. Moreover if $q \in F_{>0}$ we require the whole path to stay in $F_{>0}$. By

definition of G_D there is at least one transition in D whose origin is q and one of the two successors is q_1. To get D' we delete all other transitions with source q from D.

Clearly property (i) is preserved by this operation. To address properties (ii)–(iv), we show that every BSCC B' of $G_{D'}$ is either a BSCC of G_D or contained in the BSCC B of G_D whose maximum is q_n, in which case $\max B = \max B' = q_n$. There are two cases. If B' does not contain q_n then it does not contain q either (because $q = q_0 \dots q_n$ is still a path in $G_{D'}$). Since the only difference between G_D and $G_{D'}$ are the outgoing transitions from q then B' is actually a BSCC of G_D. If B' contains q_n then $B' \subseteq B$ (because there are less edges in $G_{D'}$ than in G_D) and since $q_n = \max B$ then $\max B = \max B'$.

As a consequence property (ii) and (iii) are preserved. And property (iv) is preserved as well: in case $q \notin F_{>0}$ then there is nothing to prove and in case $q \in F_{>0}$ then $q = q_0 \dots q_n$ is still a path in $G_{D'}$, with all vertices in $F_{>0}$. Moreover the set of vertices from which q_n is accessible is the same in G_D and $G_{D'}$ thus q_n is in a BSCC of $G_{D'}$. ◀

A strong version of the converse implication of Lemma 12 holds:

▶ **Lemma 13.** *If a $F_\forall$-trivial automaton has an accepting run, it has an acceptance witness.*

Proof. We fix an accepting run ρ on some input tree t. To extract an acceptance witness from ρ, we make use of the notion of end-component introduced in [6].

▶ **Definition 14** (End-component)**.** The *transition of a node v* is $d(v) = (\rho(v), t(v), \rho(v0), \rho(v1))$. For every branch b, we denote $\Delta^\infty(b)$ the set of transitions labelling infinitely many nodes of the branch. For every subset $D \subseteq \Delta$ we denote B_D the set of branches b such that $\Delta^\infty(b) = D$. A set of transitions $D \subseteq \Delta$ is an *end-component* of the run if B_D has nonzero probability.

Call a branch b *even* if for every transition $\delta = (q, a, l, r) \in \Delta^\infty(b)$, not only the state q but also the states l and ρ appear infinitely often on the branch in the run ρ. Almost-surely every branch is even, because each time a branch visits a node with transition δ it proceeds left or right with equal probability $\frac{1}{2}$. As a consequence,

▶ **Lemma 15.** *Let D be an end-component of the run. Then the transition graph of D has no dead-end, is strongly connected and its maximal state is in F_1.*

Proof. Denote G_D the transition graph of D, with states Q_D. Since D is an end-component then B_D has non-zero probability, and since almost every branch is even then B_D contains at least one even branch b. The set of states appearing infinitely often on b is exactly Q_D. By removing a prefix long enough of b so that only states in Q_D occur on the remaining suffix then one obtains a path in G_D which visits every state in Q_D infinitely often. Thus G_D has no dead-end and is strongly connected. Moreover every even branch in B_D has limsup $\max Q_D$ and since the run is almost-surely accepting then $\max Q_D \in F_1$. ◀

Let $\mathcal{D}$ be the collection of all end-components of the run ρ. We define two subsets of $\mathcal{D}$, denoted respectively $\mathcal{D}_0$ and $\mathcal{D}_1$, which collect the end-components whose states are respectively included in $F_{>0}$ and disjoint from $F_{>0}$. Let $D_0 \subseteq \Delta$ (resp. $D_1 \subseteq \Delta$) be the union of all end-components in $\mathcal{D}_0$ (resp. in $\mathcal{D}_1$). These transitions are easy to reach:

▶ **Lemma 16.** *Every node v has a descendant w whose transition $d(w)$ belongs to $D_0 \cup D_1$. Moreover if the state of v is in $F_{>0}$ then w can be chosen such that the path v to w is labelled by $F_{>0}$ and the transition $d(w)$ is in D_0.*

Proof. Let v be a node and S_v the set of branches which visit v and, in case v is labelled by $F_{>0}$, visit only $F_{>0}$-labelled nodes below v. Since the run is accepting then S_v has positive probability. By definition of end-components, almost-every branch is in $\bigcup_{D\in\mathcal{D}} B_D$. Thus there exists an end-component D such that $B_D \cap S_v$ has positive probability. As a consequence, v has a descendant w whose transition is in D. Since almost-every branch is even and $B_D \cap S_v$ has positive probability then there is at least one branch in $B_D \cap S_v$ which visits infinitely often all states appearing in Q_D. In case v is labelled by $F_{>0}$, this implies that $Q_D \subseteq F_{>0}$ thus $D \in \mathcal{D}_0$, and the proof of the second statement is complete. In case v has no descendant labelled by $F_{>0}$ this implies that $Q_D \cap F_{>0} = \emptyset$ thus $D \in \mathcal{D}_1$, and the first statement holds in this case. In the remaining case, v has a descendant v' labelled with $F_{>0}$, which itself has a descendant w whose transition belongs to some $D \in \mathcal{D}_0$, thus the first statement holds for v. ◀

We terminate the proof of Lemma 13. Let G_0 (resp. G_1) the transition graph of D_0 (resp. D_1) and denote Q_0 (resp. Q_1) the set of states of G_0 (resp. G_1).

Let D be the set of all transitions appearing in the run. According to Lemma 16, in the transition graph G_D, $Q_0 \cup Q_1$ is accessible from every state $q \in Q_D$ and moreover Q_0 is accessible from every state $q \in Q_D \cap F_{>0}$ following a path in $Q_D \cap F_{>0}$.

We say that an edge (q, r) of G_D is *progressive* if $q \notin Q_0 \cup Q_1$ and either ($q \in F_{>0}$ and $r \in F_{>0}$ and (q, r) decrements the distance to Q_0 in G_D) or ($q \notin F_{>0}$ and (q, r) decrements the distance to $Q_0 \cup Q_1$ in G_D). Every state in $Q_D \setminus (Q_0 \cup Q_1)$ is the source of at least one progressive edge.

We denote D_+ the union of D_0 and D_1 plus all the transitions $\delta = (q, a, r_0, r_1) \in D$ such that either (q, r_0) or (q, r_1) is progressive. Then D_+ has all four properties of Lemma 12. Denote G_+ the transition graph associated to D_+. Property (i) holds because every state in Q_D, including the initial state, is either in $Q_0 \cup Q_1$ or is the source of a progressive edge.

Remark that the BSCCs of G_+ are exactly the BSCCs of G_0 and G_1. Since both G_0 and G_1 are unions of strongly connected graphs, they are equal to the union of their BSCCs. The BSCCs of G_0 and G_1 are still BSCCs in G_+ because no edges are added inside them (progressive edges have their source outside G_0 and G_1). Following the progressive edges leads to G_0 or G_1 from every state in G_+, thus there are no other BSCCs in G_+.

This implies property (ii) because, according to Lemma 15, both graphs G_0 and G_1 are the union of strongly connected graphs whose maximal states are in F_1. This also implies property (iii) since $Q_0 \subseteq F_{>0}$ and $Q_1 \cap F_{>0} = \emptyset$. Property (iv) is obvious for states in Q_0 because Q_0 is a union of BSCCs included in $F_{>0}$. Property (iv) holds as well for states in $(Q_D \cap F_{>0}) \setminus Q_0$, the path to Q_0 is obtained following the progressive edges in $F_{>0} \times F_{>0}$. ◀

Proof of Theorem 10. According to Lemma 13, the existence of an accepting run implies the existence of an acceptance witness and according to Lemma 12 this implies the existence of a positional accepting run. Guessing a subset of transitions and checking it is an acceptance witness can be done in non-deterministic polynomial time. ◀

5 Emptiness of nonzero automata is in co-NP

In this section we show how to decide the emptiness of nonzero automata. The main ingredient are jumping games.

Call a run $\{F_1, F_{>0}\}$-*accepting* if it satisfies the almost-surely and the nonzero acceptance condition, but it does not necessarily satisfy the surely accepting condition, and the condition on the initial state is dropped as well.

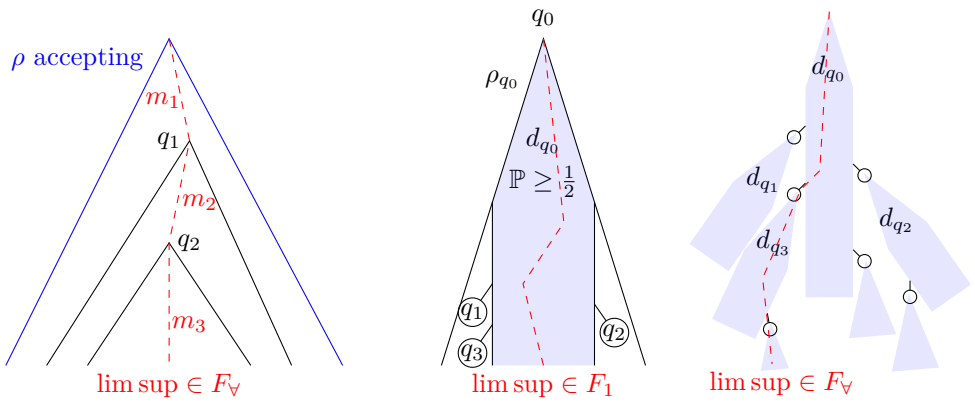

Figure 1 The left picture illustrates how an accepting run is turned into a winning strategy for Automaton in the jumping game, the two other pictures illustrate the converse transformation.

The jumping game. For a run ρ, define its *profile* to be the set of state pairs (q, m) such that some non-root node in ρ has state q and m is the maximal state of its strict ancestors.

The *jumping game* is a parity game played by two players, *Automaton* and *Pathfinder*. Positions of Automaton are states of the automaton and positions of Pathfinder are profiles of $\{F_1, F_{>0}\}$-accepting runs, not necessarily positional. The game is an edge-labelled parity game, i.e. the priorities are written on the edges. The edges originating in Automaton positions are of the form

$$q \xrightarrow{q} \Pi \qquad \text{such that } \Pi \text{ is the profile of some } \{F_1, F_{>0}\}\text{-accepting run with root state } q.$$

The edges originating in Pathfinder positions are of the form

$$\Pi \xrightarrow{m} q \qquad \text{such that } (q, m) \in \Pi.$$

We say that Automaton wins the jumping game if he has a winning strategy from the position which is the initial state of the automaton. If the play ever reaches a dead-end, i.e. a state which is not the root of any $\{F_1, F_{>0}\}$-accepting run, then the game is over and Automaton loses. Otherwise Automaton wins iff the limsup of the priorities is in $F_\forall$.

Lemmas 17 and 18 below establish that non-emptiness of a nonzero automaton is equivalent to Automaton winning the jumping game, and this can be decided in NP.

▶ **Lemma 17.** *The automaton is nonempty if and only if Automaton wins the jumping game.*

Sketch of Proof. The proof transforms an accepting run ρ of the nonzero automaton into a winning strategy σ of Automaton, and back, this is illustrated by Fig. 1.

When the nonzero automaton has an accepting run ρ, Automaton can win the jumping game as follows. For a start, Automaton plays the profile Π_0 of ρ. Then Pathfinder chooses some pair $(q_1, m_1) \in \Pi_0$, by definition of profiles this corresponds to some non-root node v_1 of ρ labelled by q_1 and m_1 is the maximal state of the ancestors of v_1. At each step $n > 0$, Pathfinder chooses some pair $(q_n, m_n) \in \Pi_n$ corresponding to some node v_{n+1} whose v_n is a strict ancestor, and Automaton plays the profile Π_n of the subtree ρ_n of ρ rooted in v_n. Since the run ρ is accepting then *a fortiori* the run ρ_n is $\{F_1, F_{>0}\}$-accepting. Quite clearly, this is a winning strategy for Automaton.

Conversely, we use a positional winning strategy of Automaton (whose existence is well-known [12]) to build an accepting run of the nonzero automaton. Denote W the set of states

winning for Automaton. With every state q in W we associate the profile Π_q chosen by the positional winning strategy of Automaton and a $\{F_1, F_{>0}\}$-accepting run ρ_q with profile Π_q.

We show the existence of a leaf-free subtree d_q of ρ_q such that:

(a) the set of branches of d_q has probability $\geq \frac{1}{2}$,

(b) *every* branch of d_q has limsup in F_1,

(c) for every node v of d_q with state in $F_{>0}$, the set of branches of d_q which visit v and visit only $F_{>0}$-labelled nodes below v has nonzero probability.

Since ρ_q is almost-surely accepting, then according to Lemma 1, there is a subtree d_q of ρ_q whose set of branches has probability $\geq \frac{1}{2}$ and *all of them* have limsup in F_1 (while in the run ρ_q there may be a non-empty set of branches with limsup in $F_\forall \setminus F_1$, with probability zero). Since we are only interested in branches of d_q, we can assume that d_q is leaf-free. This guarantees properties a) and b) but not c), however using Lemma 1 again, we can extend d_q to d'_q such that property c) holds as well.

These partial runs $(d'_q)_{q\in W}$ can be combined in order to get a graph whose unravelling, starting from the initial state, is an accepting run of the automaton. Each time a branch enters a subtree d'_q, there is probability $\geq \frac{1}{2}$ to stay in d_q forever. Thus almost every branch of the unravelling eventually stays in one of the subtrees $(d'_q)_{q\in W}$, thus has limsup in $F_1 \subseteq F_\forall$ according to property b). As a consequence the unravelling is almost-surely accepting. Still, with probability 0, some branches switch infinitely often from a subtree to another. These branches correspond to an infinite play consistent with σ and are $F_\forall$-accepting. ◀

▶ **Lemma 18.** *Given a nonzero automaton, whether Automaton wins the jumping game is decidable in* NP .

The game is not constructed explicitly, which would require exponential time, but strategies of Automaton can be represented in a compact way, which is enough to get the NP upper bound. This result was recently improved: the winner can be decided in NP∩co-NP, see [2].

Sketch of Proof. By positional determinacy of parity games, it suffices to find a positional strategy of player Automaton, which maps states to profiles of $\{F_1, F_{>0}\}$-accepting runs. It is equivalent and easier to find an *acceptance witness*. This is a pair (W, σ) where W is a subset of Q containing the initial state of the automaton, and $\sigma : W \to 2^{W\times W}$ satisfies:

α) For every sequence $(q_0, m_0)(q_1, m_1)\dots$ in $(W \times W)^\omega$, if q_0 is the initial state of the automaton and $\forall n, (q_{n+1}, m_{n+1}) \in \sigma(q_n)$ then $\limsup_n m_n \in F_\forall$.

β) $\forall q \in W$, $\sigma(q)$ *contains* the profile of a $\{F_1, F_{>0}\}$-accepting run with initial state q.

Finding a witness can be done in NP. Condition α) is checked in linear time. Given $\Pi \subseteq Q\times Q$, one can use Theorem 10 to check condition Condition β) in NP, by storing in the state space of the automaton the maximal state of the ancestors of the curent node. ◀

Example: the everywhere positive language. A tree t on the alphabet $\{a, b\}$ is *everywhere positive* if for every node v,

1. there is positive probability to see only the letter $t(v)$ below v,
2. there is positive probability to see finitely many times the letter $t(v)$ below v.

This language is non-empty and contains no regular tree. The language of everywhere positive trees with root state a is recognized by a nonzero automaton with six states

$$\{s_b < s_a < n_b < n_a < f_b < f_a\} \quad .$$

On a node labelled by letter a, the automaton can perform a transition from any of the three states $\{s_b, n_b, f_a\}$, meaning intuitively "searching for b", "not searching for b" and "just

found a". From these states the automaton can choose any pair of successor states which intersects $\{s_b, f_b\}$. Transitions on letter b are symmetrical. The acceptance condition is:

$$F_\forall = \{n_a, n_b, f_a, f_b\} \qquad F_1 = F_\forall \qquad F_{>0} = \{n_a, s_a, n_b, s_b\} \ .$$

Among the simplest moves of Automaton in the jumping game are the two moves $n_b \to \{(n_b, n_b)(s_b, n_b)\}$ and $s_b \to \{(n_b, n_b)(s_b, n_b)\}$, which correspond to the profiles of some $\{F_1, F_{>0}\}$-accepting runs on the tree whose all nodes have letter a, and everywhere in the tree the automaton applies the same two transitions $n_b \to_b (n_b, s_b)$ and $s_b \to_b (n_b, s_b)$. In those runs, the automaton always looks for a letter b in the right direction (state s_b), and does not look for b in the left direction (state n_b). Since the tree has no b at all then the quest for a letter b is hopeless, and on all branches of the run that ultimately always turn right (i.e. branches in $\{0,1\}^*1^\omega$), the automaton ultimately stays in state s_b and the branch has limsup s_b, which is neither in $F_\forall$ nor in F_1. But such branches happen with probability zero: almost-every branch performs infinitely many turns left and right, thus has limsup n_b. As a consequence such a run is almost-surely accepting: Such a run is nonzero-accepting as well because every node labelled by $F_{>0}$ has all its descendants labelled by $F_{>0}$.

Yet legal, these two moves are not good options for Automaton in the jumping game because then Pathfinder can generate the play

$$s_b \xrightarrow{s_b} \{(n_b, n_b)(s_b, n_b)\} \xrightarrow{n_b} s_b \xrightarrow{s_b} \{(n_b, n_b)(s_b, n_b)\} \xrightarrow{s_b} s_b \xrightarrow{s_b} \dots$$

which has limsup $n_b = \max\{s_b, n_b\}$ and is losing for Automaton since $n_b \notin F_\forall$.

Actually, Automaton can win with the moves

$$s_a/n_a \to \{(f_a, f_a), (n_b, f_a), (s_b, f_a), (n_a, n_a), (s_a, n_a)\}$$
$$f_a \to \{(n_b, f_a), (s_b, f_a)\}$$

and their symmetric counterparts from states $\{s_b, n_b, f_b\}$. In the jumping game, this forces Pathfinder to take only edges labelled by one of the states $\{f_a, n_a, f_b, n_b\}$. These states dominate the states $\{s_a, s_b\}$ thus the limsup of the corresponding plays is in $F_\forall$ and Automaton wins.

6 Conclusion

We have shown that the emptiness problem for zero and nonzero automata is decidable and belongs to co-NP. As a consequence, the satisfiability for the logic MSO+zero from [1] is decidable (in non-elementary time), when zero is the unary predicate that checks a set of branches has probability 0.

As shown by Stockmeyer, the satisfiability problem for first-order logic on finite words cannot be solved in elementary time [11]. Therefore any translation from a logic stronger than first-order logic on finite words (such as TMSO+zero on infinite trees) to an automaton model with elementary emptiness (such as nonzero automata) is necessarily non-elementary. This does not make the relatively low NP complexity of nonzero automata any less interesting. One can imagine other logics than TMSO+zero, either less expressive or maybe even equally expressive but less succinct, which will have a relatively low complexity by virtue of a translation into nonzero automata. One natural direction is the study of temporal logics.

Our results were recently improved [2]: the emptiness of nonzero automata actually belongs to NP∩co-NP, and is even in PTIME for $F_\forall$-trivial automata.

Acknowledgments. We thank Henryk Michalewski and Matteo Mio for helpful discussions.

References

1. Mikolaj Bojanczyk. Thin MSO with a Probabilistic Path Quantifier. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*, volume 55 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 96:1–96:13. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2016. doi:10.4230/LIPIcs.ICALP.2016.96.
2. Mikołaj Bojańczyk, Hugo Gimbert, and Edon Kelmendi. Emptiness of zero automata is decidable. Technical report, CNRS, 2017. URL: http://arxiv.org/abs/1702.06858.
3. Tomás Brázdil, Vojtech Forejt, and Antonín Kucera. Controller synthesis and verification for mdps with qualitative branching time objectives. In *ICALP 2008.*, pages 148–159, 2008.
4. C. S. Calude, S. Jain, B. Khoussainov, W. Li, and F. Stephan. Deciding parity games in quasipolynomial time. Technical report, CDMTCS, October 2016. URL: https://www.cs.auckland.ac.nz/research/groups/CDMTCS/researchreports/index.php?download&paper_file=631.
5. Arnaud Carayol, Axel Haddad, and Olivier Serre. Randomization in automata on infinite trees. *ACM Trans. Comput. Log.*, 15(3):24:1–24:33, 2014. doi:10.1145/2629336.
6. L. de Alfaro. *Formal Verification of Probabilistic Systems.* PhD thesis, Stanford University, december 1997.
7. A. S. Kechris. *Classical Descriptive Set Theory.* Graduate Texts in Mathematics. Springer-Verlag, 1995.
8. Daniel Lehmann and Saharon Shelah. Reasoning with time and chance. *Information and Control*, 53(3):165–1983, 1982.
9. Henryk Michalewski and Matteo Mio. Measure quantifier in monadic second order logic. In *LFCS 2016, Deerfield Beach, FL, USA, January 4-7, 2016. Proceedings*, pages 267–282, 2016. doi:10.1007/978-3-319-27683-0_19.
10. Henryk Michalewski, Matteo Mio, and Mikołaj Bojańczyk. On the regular emptiness problem of subzero automata. *CoRR*, abs/1608.03319, 2016. URL: http://arxiv.org/abs/1608.03319.
11. Larry J. Stockmeyer. *The Complexity of Decision Problems in Automata Theory and Logic.* PhD thesis, MIT, 1974.
12. Wieslaw Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theor. Comput. Sci.*, 200(1-2):135–183, 1998. doi:10.1016/S0304-3975(98)00009-7.