On the Problem of Computing the Probability of Regular Sets of Trees
Henryk Michalewski, Matteo Mio

To cite this version:

HAL Id: hal-02140487
https://hal.science/hal-02140487
Submitted on 28 May 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
On the Problem of Computing the Probability of Regular Sets of Trees

Henryk Michalewski¹ and Matteo Mio²

¹ University of Warsaw, Poland
² CNRS/ENS-Lyon, France

Abstract

We consider the problem of computing the probability of regular languages of infinite trees with respect to the natural coin-flipping measure. We propose an algorithm which computes the probability of languages recognizable by game automata. In particular this algorithm is applicable to all deterministic automata. We then use the algorithm to prove through examples three properties of measure: (1) there exist regular sets having irrational probability, (2) there exist comeager regular sets having probability 0 and (3) the probability of game languages \( W_{i,k} \), from automata theory, is 0 if \( k \) is odd and is 1 otherwise.

1998 ACM Subject Classification F.1.1 Models of Computation, F.1.2 Modes of Computation

Keywords and phrases regular languages of trees, probability, meta-parity games

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2015.489

1 Introduction

Regular languages of trees are sets of infinite binary trees, labeled by letters from a finite alphabet \( \Sigma \), definable by a formula of Monadic Second Order (MSO) logic interpreted over the full binary tree [25] or, equivalently, specified by an alternating tree automaton [19].

In this paper we consider the following problem. Suppose a \( \Sigma \)-labeled tree \( t \) is generated by labeling each vertex by a randomly and uniformly chosen letter \( a \in \Sigma \). For a given regular language \( L \), what is the probability that \( t \) belongs to \( L \)? By probability we mean the standard coin–flipping probability measure \( \mu \) (see Section 2 for definitions) on the space of \( \Sigma \)-labeled trees. Hence a precise formulation of our problem is as follows.

Probability Problem: Does there exist an algorithm which for a given regular language of trees \( L \) computes the probability \( \mu(L) \)?

A qualitative variant of the problem only asks for a decision procedure for the question “is \( \mu(L) = 1 \)?”. The problem is well posed since it was recently shown in [12, Theorem 1] that regular sets of trees are measurable with respect to any Borel measure and thus, in particular, with respect to the coin-flipping measure. An extended version of this paper is available at [15].
1.1 Main Results

We give a positive solution to the Probability Problem for a subclass of regular languages.

▶ Theorem 1. Let $L$ be a regular set of infinite trees recognizable by a game automaton. Then the probability of $L$ is computable and is an algebraic number.

Game automata (see [10, 11, 21]) are special types of alternating parity tree automata. The class of languages recognizable by game automata includes, beside all deterministic languages, other important examples of regular sets. The most notable examples are the game languages $W_{i,k}$ which play a fundamental role in the study of tree languages with topological methods [3, 12]. Game automata definable languages are, at the present moment, the largest known subclass of regular languages for which the long-standing Mostowski–Rabin index problem\(^1\) is known to be decidable (see [10, 11]). Theorem 1 confirms the good algorithmic properties of game automata. At the same time, however, we suspect that generalizing the result of Theorem 1 to arbitrary regular languages might be hard. Some ideas for further research in this direction are discussed in Section 6.

From Theorem 1 we derive the following propositions (for proofs see Section 5).

▶ Proposition 2. There exists a regular language of trees $L$ definable by a deterministic automaton such that $L$ has an irrational probability.

▶ Proposition 3. There exists a regular language of trees $L$ definable by a deterministic automaton such that $L$ is comeager and $L$ has probability 0.

These two propositions should be contrasted with known properties of regular languages of infinite words. First, a result of Staiger in [22] states that a regular language $L$ of infinite words has coin-flipping measure 0 if and only if it is of Baire first category (or meager). 3 shows that this correspondence fails in the context of infinite trees. Second, the coin-flipping measure of a regular language of infinite words is always rational (see, e.g., Theorem 2 of [7]). Hence, the probabilistic properties of regular languages of trees seem to be significantly more refined than in the case of languages of $\omega$-words.

Lastly, we calculate the probability of all game languages $W_{i,k}$ (see [3, 12] and Subsection 5.4), a result that might eventually be useful given the importance of game languages in the topological study of regular sets of trees.

▶ Proposition 4. For $0 \leq i < k$, the game language $W_{i,k}$ has probability 0 if $k$ is odd and 1 if $k$ is even.

1.2 The Algorithm

In Section 4 we propose Algorithm 2 which computes the probability of regular languages recognized by game automata. Algorithm 2 is based on a reduction to Markov Branching plays (MBP’s): to each game automaton $A$ we associate a MBP $M$. The value of $M$ can be computed and corresponds to the probability of the language recognized by $A$. This reduction is described in Sections 3 and 4.

The notion of MBP, as a special kind of two-player stochastic meta-parity game has been introduced by the second author in [16, 17] in order to interpret a probabilistic version

---

\(^1\) The Mostowski–Rabin Problem: for a given regular language $L$, compute the minimal number of priorities required to define $L$ using an alternating parity tree automaton.
of the modal $\mu$-calculus. For a given MBP $\mathcal{M}$ having $n$ states, the vector $val \in [0, 1]^n$ of values of $\mathcal{M}$ can be expressed as the solution of a system $\mathcal{S}$ of (nested) least and greatest fixed-point equations over the space $[0, 1]^n$. From $\mathcal{S}$ one can then construct a first order formula $\phi_{\mathcal{S}}(val)$ in the language of real-closed fields having the property that $val$ is the unique tuple of real numbers satisfying $\phi_{\mathcal{S}}$. The tuple $val$ can be computed by Tarski’s quantifier elimination algorithm [24] and consists of algebraic numbers. See Algorithm 1 in Section 3 for a description of the procedure for computing the value of MBP’s.

One can find interesting the connections between the machinery of MBP’s (and thus, as mentioned, the probabilistic $\mu$-calculus), the class of languages definable by game automata, the algorithmic problem of computing the probability of regular languages of trees and the usage of Tarski’s quantifier elimination procedure.

1.3 Related Work

In [23] L. Staiger presented an algorithm for computing the Hausdorff measure of regular sets of $\omega$-words. The method, based on the decomposition of the input language into simpler components, can be adapted to compute the coin–flipping measure of regular sets of $\omega$-words. Our research on the coin-flipping measure of regular languages of trees can be seen as a continuation of Staiger’s work.

Natural variants of the qualitative version of the Probability Problem, obtained by replacing “has probability 1” by other notions of largeness, are known to have positive solutions: in [20] D. Niwiński described an algorithm which takes as input a regular language of trees $L$ (presented as a Rabin tree automaton) and decides if $L$ is uncountable and, similarly, an algorithm for establishing if a regular language of trees $L$ is comeager can be extracted from the result of [14].

Addendum. After the submission of this article we have been informed that the Probability Problem has already been implicitly considered in [8], although differently phrased as the verification problem for a class of stochastic branching processes. Following our terminology, in [8] the authors provide an algorithm for computing the probability of regular languages definable by deterministic tree automata. Hence our results can be seen as extending the work of [8] from deterministic to game-automata definable languages.

2 Background in Topology and Automata Theory

2.1 Topology and measure

In this section we present elementary topological and measure–theoretical notions required in this work. We refer to [13] as a standard reference on the subject.

The set of natural numbers is denoted by $\omega$. A topological space $X$ is Polish if it is separable and completely metrizable. An important example of a Polish space is the Cantor space $\{0,1\}^\omega$ of infinite sequences of bits endowed with the product topology. In this paper we are interested in the probability Lebesgue measure $\mu$ on the product space $\Sigma^I$ for $I$, a countable set of indices. The measure $\mu$ is uniquely defined by the assignment $\mu(\{t \in \Sigma^I \mid t(i_1) = a_1, \ldots, t(i_k) = a_k\}) = \left(\frac{1}{|\Sigma|}\right)^k$ for $i_1, \ldots, i_k \in I$, $a_1, \ldots, a_k \in \Sigma$ ($i_j \neq i_{j'}$ whenever $j \neq j'$, see [13, Chapter 17] for additional details). In particular, for the alphabet $\Sigma = \{0, 1\}$ and $I = \omega$ this is known as the coin–flipping probability measure on the Cantor space.

The countable set $V = \{L, R\}^*$ of finite words over the alphabet $\{L, R\}$ is called the full binary tree and each $v \in \{L, R\}^*$ is referred to as a vertex. The product space $\Sigma^V$ is denoted...
On the Problem of Computing the Probability of Regular Sets of Trees

by \( T_\Sigma \) and an element \( t \in T_\Sigma \) is called a \( \Sigma \)-labeled tree, or just a \( \Sigma \)-tree. Intuitively, the stochastic processes associated with the coin-flipping measure \( \mu \) on \( T_\Sigma \) generates an infinite \( \Sigma \)-tree by labeling each vertex with a randomly (uniformly) chosen label in \( \Sigma \).

Given a topological space \( X \), a set \( A \subseteq X \) is nowhere dense if the interior of its closure is the empty set, that is \( \text{int}(c(A)) = \emptyset \). A set \( A \subseteq X \) is of (Baire) first category (or meager) if \( A \) can be expressed as a countable union of nowhere dense sets. The complement of a meager set is called comeager.

### 2.2 Alternating Parity Tree Automata and Game Automata

We include a brief exposition of alternating automata which follows the presentation in [19, Appendix C]. In this paper we are mostly interested in a subclass of alternating parity tree automata called game automata, which is introduced later in the Section.

> **Definition 5** (Alternating Parity Tree Automaton). Given a finite set \( X \), we denote with \( \mathcal{DL}(X) \) the set of expressions \( e \) generated by the grammar \( e ::= x \in X \mid e \cdot e \mid e \lor e \). An alternating parity tree automaton over a finite alphabet \( \Sigma \) is a tuple \( A = (\Sigma, Q, q_0, \delta, \pi) \) where \( Q \) is a finite set of states, \( q_0 \in Q \) is the initial state, \( \delta : Q \times \Sigma \rightarrow \mathcal{DL}(\{L, R\} \times Q) \) is the alternating transition function, and \( \pi : Q \rightarrow \omega \) is the parity condition.

An alternating parity tree automaton \( A \) over the alphabet \( \Sigma \) defines, or “accepts”, a set of \( \Sigma \)-trees. The acceptance of a tree \( t \in T_\Sigma \) is defined via a two-player (\( \exists \) and \( \forall \)) game of infinite duration denoted by \( A(t) \). Game states of \( A(t) \) are of the form \( \langle t, q \rangle \) or \( \langle t, e \rangle \) with \( t \in \{L, R\}^* \), \( q \in Q \) and \( e \in \mathcal{DL}(\{L, R\} \times Q) \).

The game \( A(t) \) starts at state \( \langle e, q_0 \rangle \). Game states of the form \( \langle t, e \rangle \), including the initial state, have only one successor state, to which the game progresses automatically. The successor state is \( \langle t, e \rangle \) with \( e = \delta(q, a) \), where \( a = t(t) \) is the labeling of the vertex \( t \) given by \( t \). The dynamics of the game at states \( \langle t, e \rangle \) depends on the possible shapes of \( e \). If \( e = e_1 \lor e_2 \), then Player \( \exists \) moves either to \( \langle t, e_1 \rangle \) or \( \langle t, e_2 \rangle \). If \( e = e_1 \land e_2 \), then Player \( \forall \) moves either to \( \langle t, e_1 \rangle \) or \( \langle t, e_2 \rangle \). If \( e = (L, q) \) then the game progresses automatically to the state \( \langle t.L, q \rangle \). Lastly, if \( e = (R, q) \) the game progresses automatically to the state \( \langle t.R, q \rangle \). Thus a play in the game \( A(t) \) is a sequence \( \Pi \) of game–states, that looks like: \( \Pi = ((e_0, q_0), \ldots, (L, q_1), \ldots, (L.R, q_2), \ldots, (LRL, q_3), \ldots, (LRLL, q_4), \ldots) \), where the dots represent part of the play in game–states of the form \( \langle t, e \rangle \). Let \( \omega(\Pi) \) be the set of automata states \( q \in Q \) occurring infinitely often in configurations \( \langle t, e \rangle \) of \( \Pi \). We then say that the play \( \Pi \) of \( A(t) \) is winning for \( \exists \), if \( \max \{ \pi(q) \mid q \in \omega(\Pi) \} \) is an even number. The play \( \Pi \) is winning for \( \forall \) otherwise. The set (or “language”) of \( \Sigma \)-trees defined by \( A \) is the collection \( \{ t \in T_\Sigma \mid \exists \text{ has a winning strategy in the game } A(t) \} \).

We reserve the symbols \( T \) and \( \perp \) for two special sink states having even and odd priority, respectively. The transition function is defined, for all \( a \in \Sigma \), as \( \delta(T, a) = (L, T) \lor (R, T) \) and \( \delta(\perp, a) = (L, \perp) \lor (R, \perp) \). Clearly every tree is accepted at the state \( T \) and rejected at \( \perp \). Game automata are a subfamily of alternating parity tree automata satisfying the constraint that, for each \( q \in Q \) and \( a \in \Sigma \), the transition \( \delta(q, a) = e \) has either the form \( e = (L, q_L) \lor (R, q_R) \) or \( e = (L, q_L) \lor (R, q_R) \). Transitions of a game automaton \( A \) can be schematically depicted as in the figure above with the left–hand and right–hand diagrams representing the transitions \( (q, a) \rightarrow (L, q_L) \lor (R, q_R) \) and \( (q, a) \rightarrow (L, q_L) \lor (R, q_R) \), respectively. Deterministic automata are a subfamily of game automata satisfying the stronger constraint that, for each
3 Introduction to meta-parity games

In this section we describe a class of stochastic processes called Markov branching plays (MBP’s) [16, 17] which, as we will observe, is closely related to game automata and will provide a method for computing the probability of regular languages defined by such automata. For a quick overview, a procedure for computing the value associated with a MBP is presented as Algorithm 2 in the next section. The procedure for computing the probability of regular languages defined by game automata is presented as Algorithm 1, at the end of this section. The construction starts from the root of 

MBP’s as Stochastic Processes: given a MBP \( \mathcal{M} = \langle \langle S, E \rangle, (S_p, B_2, B_\text{d}), p, \text{Par} \rangle \) and an initial vertex \( s_0 \in S \), the stochastic process of construction of \( T \) is described as follows.

- The construction starts from the root of \( T \) which is labeled by \( s_0 \).
- A leaf \( x \) in the so far constructed tree \( T \) is extended, independently from all other leaves, depending on the type of its labeling state \( s \), as follows:
  - If \( s \in S_p \) then \( x \) is extended with a unique child which is labeled by a successor state \( s' \) of \( s \) randomly chosen in accordance with \( p(s) \).
  - If \( s \in B_2 \) or \( s \in B_\text{d} \) and \( \{ s_1, \ldots, s_n \} \) are the successors of \( s \) in \( \mathcal{M} \), then \( x \) is extended with \( n \) children \( y_1, \ldots, y_n \) and \( y_i \) is labeled by \( s_i \), for \( 1 \leq i \leq n \).
We give in Figure 1 an example of a MBP. Probabilistic states, ∃-branching and ∀-branching states are marked as circles, diamond and boxes, respectively. The first six initial steps of the stochastic process associated with $M$ at state $q_1$ are depicted in Figure 2. In the first step, the construction of $T$ starts by labeling the root by $q_1$. Since $q_1$ is a probabilistic state, the tree is extended (second step) with only one child labeled by either $q_2$ (with probability $\frac{1}{3}$) or $q_3$ (prob. $\frac{2}{3}$). The picture shows the case when $q_2$ is chosen. Since the new leaf is labeled by $q_2$, and this is a $∃$-branching state, the tree is extended by adding one new vertex for each successor of $q_2$ in $M$, i.e., for both $q_1$ and $q_4$. The construction continues as described above. For example, the probability that the generated infinite tree will have the prefix as at the bottom right of Figure 2 is $\frac{1}{3} \cdot \frac{2}{3} \cdot \frac{1}{2} = \frac{2}{18}$.

The kind of infinite trees produced by the stochastic process just described are called branching plays. Branching plays are characterized by the property that each vertex labeled with a probabilistic state has only one child, and each vertex labeled with a ($∃$ or $∀$) branching state $s$ has as many children as there are successors of $s$ in the MBP.

The collection of branching plays in a MBP $M$ starting from a state $s$ is denoted by $BP(M, s)$. The set $BP(M, s)$ naturally carries a Polish topology making $BP(M, s)$ homeomorphic to the Cantor space (see, e.g., Definition 4.4 in [17]). The stochastic process associated to a MBP $M$, specified on the previous page, can be naturally formalized by a probability measure $\mu_M$ over the space $BP(M, s)$ of branching plays. See also Definition 4.7 in [17] for a formal definition.

Each branching play $T$ can itself be viewed as an ordinary (infinite) two-player parity game $G(T)$, played on the tree structure of $T$, by interpreting the vertices of $T$ labeled by $∃$-branching and $∀$-branching states as under the control of Player $∃$ and Player $∀$, respectively. All other states (i.e., those labeled by a probabilistic state) have a unique successor in $T$ to which the game $G(T)$ progresses automatically. Lastly, the parity condition associated to each vertex corresponds to the parity assigned in $M$ to the state labeling it. We denote with $W_s$ the set of branching plays starting at $s$ and winning for Player $∃$, i.e., the set defined as: $W_s = \{ T \in BP(M, s) | \text{ Player } ∃ \text{ has a winning strategy in } G(T) \}$.
Theorem 10 below refines Proposition 8 by identifying maximal and minimal priorities used in the MBP, respectively, and let parity games. To adhere to such notation, we will define a function notation adopted in the textbook \[1, \S 4.3\] for presenting a similar result valid for ordinary system of nested (least and greatest) fixed-point equations. Its formulation closely follows the equality \(\mu = f(\mu)\) holds, where \(f:\{0,1\}^n \rightarrow \{0,1\}^n\) is:

\[
(f \left( \begin{array}{c} x_1 \\ \vdots \\ x_n \end{array} \right) )_i = \begin{cases} \sum_{\{j \mid (s_i,s_j) \in E\}} p(s_i)(s_j) \cdot x_j & \text{if } s_i \in S_P \\ \prod_{\{j \mid (s_i,s_j) \in E\}} x_j & \text{if } s_i \in B_P \\ \prod_{\{j \mid (s_i,s_j) \in E\}} x_j & \text{if } s_i \in B_E \\ \end{cases}
\]

Proof. Here we sketch the main idea of the argument, for a formal proof, see Theorem 4.22 of \[17\]. If \(s_i\) is a probabilistic state, then \(val_i\) is the weighted average of the value of its successors, since the stochastic process associated with the MBP chooses a unique successor \(s_j\) of \(s_i\) with probability \(p(s_i)(s_j)\). If \(s_i\) is a \(\forall\)-branching state, then \(val_i\) is the probability that all \(\forall\)-generated subtrees are winning for Player \(\exists\) and this is captured by the \(\prod\) expression. Similarly, if \(s_i\) is a \(\exists\)-branching state then \(val_i\) is the probability that at least one generated subtree is winning for Player \(\exists\), as formalized by the \(\prod\) expression. Hence the vector \(val\) is one of the fixed-points of the function \(f:\{0,1\}^n \rightarrow \{0,1\}^n\).

Theorem 10 below refines Proposition 8 by identifying \(val\) as the unique vector satisfying a system of nested (least and greatest) fixed-point equations. Its formulation closely follows the notation adopted in the textbook \[1, \S 4.3\] for presenting a similar result valid for ordinary parity games. To adhere to such notation, we will define a function \(g\), a variant of the function \(f\) presented above. Let \(k = \max\{Par(s) \mid s \in S\}\) and \(l = \min\{Par(s) \mid s \in S\}\) be the maximal and minimal priorities used in the MBP, respectively, and let \(c = k - l + 1\).

Definition 9. The function \(g:\{0,1\}^n \rightarrow \{0,1\}^n\) is defined as follows:

\[
(g \left( \begin{array}{c} x_1^1 \\ \vdots \\ x_n^1 \\ \vdots \\ x_1^c \\ \vdots \\ x_n^c \end{array} \right) )_i = \begin{cases} \sum_{\{j \mid (s_i,s_j) \in E\}} p(s_i)(s_j) \cdot x_j^{Par(s_j)} & \text{if } s_i \in S_P \\ \prod_{\{j \mid (s_i,s_j) \in E\}} x_j^{Par(s_j)} & \text{if } s_i \in B_P \\ \prod_{\{j \mid (s_i,s_j) \in E\}} x_j^{Par(s_j)} & \text{if } s_i \in B_E \\ \end{cases}
\]
The function $g$ depends, like the function $f$, only on $n$ variables \{$x_1^{\text{Par}(s_1)}, \ldots, x_n^{\text{Par}(s_n)}$\} appearing in the body of its definition. The input of $g$ can indeed be regarded as the input of $f$ divided in $c$ baskets, where each variable $x_i$ is put in the basket corresponding to the priority of $s_i$, for $1 \leq i \leq n$.

The set $[0,1]^n$, equipped with the pointwise order defined as $(x_1, \ldots, x_n) \leq (y_1, \ldots, y_n) \iff \forall i. (x_i \leq y_i)$, is a complete lattice and the function $g$ is monotone with respect to this order in each of its arguments. Hence the Knaster–Tarski theorem ensures the existence of least and greatest points. We are now ready to state the main result regarding the values of a given MBP. We adopt standard $\mu$-calculus notation (see, e.g., [1] and [16, 17]) to express systems of least and greatest fixed-points equations.

**Theorem 10 ([16, Theorem 6.4.2]).** The following equality holds:  
\[
\begin{bmatrix}
\text{val}_1 \\
\vdots \\
\text{val}_n
\end{bmatrix} = \theta_k \begin{bmatrix}
x_1^k \\
\vdots \\
x_n^k
\end{bmatrix} \cdots \theta_l \begin{bmatrix}
x_1^l \\
\vdots \\
x_n^l
\end{bmatrix} \cdot \text{g} \begin{bmatrix}
x_1^l \\
\vdots \\
x_n^l
\end{bmatrix}, \ldots, \begin{bmatrix}
x_1^k \\
\vdots \\
x_n^k
\end{bmatrix}\]

where $\theta_i$, for $l \leq i \leq k$ is a least-fixed point operator ($\mu$) if $i$ is an odd number and a greatest-fixed point operator ($\nu$) if $i$ is even.

**Proof.** The proof goes by induction on the number of priorities in the MBP $\mathcal{M}$ and by transfinite induction on a rank-function defined on the space of branching plays. See [16] for a detailed proof.

The next theorem states that the value of a MBP is computable and is always a vector of algebraic numbers. The examples discussed in Section 5 will illustrate the applicability of this result.

**Theorem 11.** Let $\mathcal{M}$ be a MBP. Then for each state $s_i$ of $\mathcal{M}$ the value $\text{val}_i$ is computable and is an algebraic number.

**Proof.** (sketch) Using known ideas (see, e.g., Lemma 9 in [9] and Proposition 4.1 in [18]) the unique vector $\text{val}=(\text{val}_1, \ldots, \text{val}_n)$ satisfying the system of fixed-point expressions $\mathcal{S}$ given by Theorem 10 can be computed by a reduction to the first-order theory of real closed fields. A first order formula $F(x_1, \ldots, x_n)$, inductively defined from $\mathcal{S}$, is constructed with the property that $(\text{val}_1, \ldots, \text{val}_n) \in \mathbb{R}^n$ is the unique vector of reals satisfying the formula $F(x_1, \ldots, x_n)$. By Tarski’s quantifier elimination procedure [24], the formula $F(x_1, \ldots, x_n)$ can be effectively reduced to an equivalent formula $G(x_1, \ldots, x_n)$ without quantifiers, that is, to a Boolean combination of equations and inequalities between polynomials over $(x_1, \ldots, x_n)$. It then follows that the $(\text{val}_1, \ldots, \text{val}_n)$, which can be extracted from $G$ with standard methods, is a vector of algebraic numbers. In Section 5 we apply the above procedure to a number of examples.

---

2 Theorem 6.4.2 of [16] actually proves a stronger result valid for arbitrary $2\frac{1}{2}$-player meta-parity games whereas, as mentioned in the beginning of this section, Markov branching plays are $2\frac{1}{2}$-player meta-parity games without Player 1 and Player 2 states. Also, Theorem 6.4.2 of [16] is stated assuming the validity of the set-theoretic axiom MA$_{\aleph_1}$, but as shown in [12] such assumption is not necessary and can thus be dropped.
We are implicitly assuming, for the sake of simplicity, that each transition $\pi_A$ the parity condition $p$, is exactly two successors. The assignment $\delta(q,a) = (L, q_L) \land (R, q_r)$. The transition relation $E$ is defined as follows:

- a probabilistic state $s_q$ has as successors the states $\{s_{q,a} \mid a \in \Sigma\}$,
- a $\exists$-branching (resp. $\forall$-branching) state $s_{q,a}$ have two successors $s_{q_1}$ and $s_{q_2}$ where $\delta(q,a) = (L, q_1) \lor (R, q_2)$ (resp. $\delta(q,a) = (L, q_1) \land (R, q_2)$).

Note that each state $s_q$, for $q \in Q$ has exactly $|\Sigma|$ successors and that each state $s_{q,a}$ has exactly two successors. The assignment $p : S \rightarrow [0,1]$ is defined as assigning to each probabilistic state (i.e., state of the form $s_q$) a uniform distribution over its successors, that is, $p(s_q)(s_{q,a}) = \frac{1}{|\Sigma|}$. Lastly, the parity assignment $\text{Par} : S \rightarrow \omega$ of the MBP $M$ is defined as in the parity condition $\pi$ of the game automaton $A$ by the mapping $\text{Par}(s_q) = \text{Par}(s_{q,a}) = \pi(q)$.

As an illustrative example of this translation, consider the deterministic automaton $A = \langle \{q_1, q_2\}, q_1, \delta, \pi \rangle$ over the alphabet $\Sigma = \{a, b, c\}$, with parity assignment $\pi(q_2) = 2$, $\pi(q_1) = 1$ and transition $\delta$ defined by $\delta(q_1,a) = (L, q_2)$, $\delta(q_1,\bot) = (L, q_2) \land (R, q_1)$, and $\delta(q_2, l) = (L, q_1) \land (R, q_1)$, for $l \in \{a, b\}$.

---

3 We are implicitly assuming, for the sake of simplicity, that each transition $(L, q_1) \land (R, q_2)$ and $(L, q_1) \lor (R, q_2)$ of $\delta$ in $A$ is such that $q_1 \neq q_2$, and thus that $s_{q,a}$ has exactly two successors. If necessary, the game-automaton $A$ can be made satisfy this assumption by introducing additional copies of the states.

4 From Game Automata to Markov Branching Plays

In this section we present a reduction of the problem of computing the probability of regular languages definable by game automata to the problem of computing the value of a given MBP, which is algorithmically solvable using Algorithm 1.

We now describe how to construct from a game automaton $A = (Q, q_0, \delta, \pi)$ over the alphabet $\Sigma$ a corresponding MBP $M = \langle (S, E), (S_P, F_3, F_4), p, \text{Par} \rangle$. The set $S$ of states of $M$ contains a probabilistic state $s_q$, for each $q \in Q$, a $\exists$-branching state $s_{q,a}$ for each pair $(q,a)$, with $q \in Q$ and $a \in \Sigma$, such that $\delta(q,a) = (L, q_L) \lor (R, q_r)$, and a $\forall$-branching state $s_{q,a}$ for each pair $(q,a)$ such that $\delta(q,a) = (L, q_L) \land (R, q_r)$. The transition relation $E$ is defined as follows:

- a probabilistic state $s_q$ has as successors the states $\{s_{q,a} \mid a \in \Sigma\}$,
- a $\exists$-branching (resp. $\forall$-branching) state $s_{q,a}$ have two successors $s_{q_1}$ and $s_{q_2}$ where $\delta(q,a) = (L, q_1) \lor (R, q_2)$ (resp. $\delta(q,a) = (L, q_1) \land (R, q_2)$).

Note that each state $s_q$, for $q \in Q$ has exactly $|\Sigma|$ successors and that each state $s_{q,a}$ has exactly two successors. The assignment $p : S \rightarrow [0,1]$ is defined as assigning to each probabilistic state (i.e., state of the form $s_q$) a uniform distribution over its successors, that is, $p(s_q)(s_{q,a}) = \frac{1}{|\Sigma|}$. Lastly, the parity assignment $\text{Par} : S \rightarrow \omega$ of the MBP $M$ is defined as in the parity condition $\pi$ of the game automaton $A$ by the mapping $\text{Par}(s_q) = \text{Par}(s_{q,a}) = \pi(q)$.

As an illustrative example of this translation, consider the deterministic automaton $A = \langle \{q_1, q_2\}, q_1, \delta, \pi \rangle$ over the alphabet $\Sigma = \{a, b, c\}$, with parity assignment $\pi(q_2) = 2$, $\pi(q_1) = 1$ and transition $\delta$ defined by $\delta(q_1,a) = (L, q_2)$, $\delta(q_1,\bot) = (L, q_2) \land (R, q_1)$, and $\delta(q_2, l) = (L, q_1) \land (R, q_1)$, for $l \in \{a, b\}$.

---

3 We are implicitly assuming, for the sake of simplicity, that each transition $(L, q_1) \land (R, q_2)$ and $(L, q_1) \lor (R, q_2)$ of $\delta$ in $A$ is such that $q_1 \neq q_2$, and thus that $s_{q,a}$ has exactly two successors. If necessary, the game-automaton $A$ can be made satisfy this assumption by introducing additional copies of the states.

Algorithm 1: computing the vector of values of a MBP.
The corresponding MBP $M$ is schematically\footnote{\text{Due to the chosen succinct definition, the automaton $\mathcal{A}$ does not satisfy the assumption of Footnote 3. Rather than formally introducing copies $q_1$ and $q_2$ in $\mathcal{A}$, we have simply depicted all $\forall$-branching states of $\mathcal{M}$ as having two successors.}} depicted in Figure 3 (right), by representing probabilistic states with circles, $\forall$-branching states with boxes and the probabilistic assignment $p$ by the probabilities labeling the outgoing edges of probabilistic states. The soundness of our reduction is stated as follows.

\begin{proof}(sketch)\end{proof}

Since each probabilistic state has exactly one successor for every letter $a \in \Sigma$ and each branching state have precisely two successors, there exists a one-to-one correspondence between $\Sigma$-trees $t \in T_\Sigma$ and branching plays $T \in \mathcal{BP}(M, s_{q_0})$. Furthermore, it follows directly from the definition of acceptance by $\mathcal{A}$ (see Section 2.2) and the definition of the set $W_\Sigma$ (see Section 3) that $t$ is accepted by $\mathcal{A}$ if and only if the corresponding branching play $T$ is in $W_\Sigma$. Lastly, due to the uniform assignment $p$ of probabilities in $\mathcal{M}$, the coin-flipping measure $\mu$ on $T_\Sigma$ and the probability measure $\mu_M$ on $\mathcal{BP}(M, s_{q_1})$ are identical.

The result of Theorem 1 in the Introduction then follows as a corollary of Theorem 12 above and the fact that the vector of values of a MBP can be computed using Algorithm 1.

The final algorithm for computing the probability of regular languages definable by game automata is then as follows.

\begin{algorithm}
\begin{enumerate}
\item \textbf{input}: a game automaton $\mathcal{A} = (Q, q_0, \delta, \pi)$ recognizing a language $L$.
\item \textbf{output}: a real number corresponding to $\mu(L)$.
\item \textbf{begin}
\item $M \leftarrow$ Construct the MBP $M$ corresponding to $\mathcal{A}$
\item $(val_1, \ldots, val_n) \leftarrow$ Apply Algorithm 1 to compute the vector of values of the states of $M$
\item \textbf{return} the value $val_i$, where $i$ is the index of the probabilistic state $s_{q_0}$ of $M$
\end{enumerate}
\end{algorithm}

Algorithm 2: computing the probability of regular languages $L$ recognized by game automata.

\section{Examples}

In this section we will apply Algorithm 2 to analyze examples which will prove Propositions 2, 3 and 4 stated in the Introduction. In some instances, in order to perform the quantifier elimination procedure required by Algorithm 1, we use the tool \texttt{qepcad} [5].

We fix the alphabet $\Sigma = \{a, b, c\}$ and, for each $n \in \omega$, we define the regular language $L_n \subseteq T_\Sigma$ as $L_n = \{t \in T_\Sigma \mid a \text{ appears } \geq n \text{ times on every branch of } t\}$ and the language $L_\infty$ as $L_\infty = \cap_{n \in \omega} L_n$, i.e., as the set of $\Sigma$-trees having, on every branch, infinitely many occurrences of the letter $a$.

\subsection{An introductory example}

The language $L_1$ is recognized by the deterministic automaton in Figure 4 (left) defined as $\mathcal{A}_1 = (\{q_1, \top\}, q_1, \delta_1, \pi)$ where $\top$ is an accepting sink state (see Section 2.2 for automata-related definitions). The priority assignment is $\pi(q_1) = 1$ and the transition function $\delta_1$ is defined on $q_1$ as $\delta_1(q_1, a) = (L, \top) \land (R, \top)$ and $\delta_1(q_1, l) = (L, q_1) \land (R, q_1)$ for $l \in \{b, c\}$. 

Due to the chosen succinct definition, the automaton $\mathcal{A}$ does not satisfy the assumption of Footnote 3. Rather than formally introducing copies $q_1$ and $q_2$ in $\mathcal{A}$, we have simply depicted all $\forall$-branching states of $\mathcal{M}$ as having two successors.
We will compute the probability $\mu(L_1)$ using the procedure of Algorithm 2. As a first step we construct the MBP $M_1$ corresponding to $A_1$, as specified in Section 4. In order to improve readability, we have represented in Figure 4 (center) a simplified version of $M_1$ where the states $s_\top$, $s_{\top,a}$, $s_{\top,b}$ and $s_{\top,c}$ have been identified with the single state $s_{q_1,a}$. This is convenient since, clearly, all of these states have value 1. Accordingly, the MBP $M_1$ has four states, all of priority 1. Following the procedure of Algorithm 2 we need to compute the values of the states of $M_1$ using Algorithm 1. In accordance with Theorem 10, the fixed-point equation characterizing the vector $\text{val}=(\text{val}_{s_0}, \text{val}_{s_1}, \text{val}_{s_2}, \text{val}_{s_3})$ of values of the states of $M_1$ is $\text{val} = \mu \cdot g(\text{val})$, where $g$ is defined as in Figure 4 (right). Then $\text{val}_{s_0}$ is the least solution in $[0,1]$ of the equation $x = \frac{1}{3} + \frac{2}{3}x^2$. As it is simple to verify, even without running the solver based on Tarski’s quantifier elimination procedure, the solution is $\text{val}_{s_0} = \frac{1}{12}$, and this is the output returned by Algorithm 2. Hence the probability of $L_1$ is $\mu(L_1) = \frac{1}{12}$.

5.2 Examples of regular languages having irrational probabilities

This subsection constitutes a proof of Proposition 2. The automaton $A_2$ recognizing the language $L_2$ is defined as $A_2 = \langle (\{q_1, q_2, \top\}, q_2, \delta_2, \pi) \rangle$ where $q_2$ is the initial state, the priority function is defined as $\pi(q_1) = \pi(q_2) = 1$ and the transition function $\delta_2$ is defined on $q_1$ as the function $\delta_1$ of the previous example, and on the state $q_2$ as $\delta_2(q_2, a) = (L, q_1) \land (R, q_1)$ and $\delta_2(q_2, b) = (L, q_2) \land (R, q_2)$, for $a \in \{b, c\}$. The transition $\delta_2$ is shown in Figure 5 (left).

The MBP $M_2$ corresponding to $A_2$ extends the MBP $M_1$ of the previous example with the probabilistic state $s_{q_2}$ and the three $\forall$-branching states $s_{q_2,a}$, $s_{q_2,b}$ and $s_{q_2,c}$. The new part of the automaton $A_2$ is depicted in Figure 5 (center). Noting the four new states are not reachable by the other states already present in $M_1$, we already know that $\text{val}_{s_{q_1}} = \frac{1}{12}$. Hence we can consider the simplified system of fixed-point equations $\mu \cdot g(\text{val})$ for calculating the values $\text{val} = (\text{val}_{s_0}, \text{val}_{s_1}, \text{val}_{s_2}, \text{val}_{s_3})$ where $g$ is defined in Figure 5 (right). Hence the value $\text{val}_{s_0}$ is the least solution in $[0,1]$ of the equation $x = \frac{1}{12} + \frac{2}{3}x^2$ and this is $\text{val}_{s_0} = \frac{1}{4}(3 - \sqrt{7})$ which is irrational and approximately equal to 0.088.
One can verify\(^5\) that the probability of \(L_3\) is \(\mu(L_3) = \frac{1}{4}(3 - \sqrt[4]{1 + 3\sqrt{2}})\) and thus not of the form \(a + b\sqrt{c}\) for integers \(a, b, c, d\). This means that \(\mu(L_3)\) is not a quadratic irrational. By a characterization proved by Euler and Lagrange this in turn means that the continued fraction representation of \(\mu(L_3)\) is not eventually periodic.

### 5.3 Example of a comeager language of probability \(0\)

This subsection constitutes a proof of Proposition 3. The regular language \(L_\infty\) is recognized by the (deterministic) game automaton already defined in Section 4 and depicted in Figure 3 (left), where the states \(q_1\) and \(q_2\) have priority 1 and 2, respectively. The MBP associated with this automaton, depicted in Figure 3 (right), has eight states. The vector of values \(val\) is equal to \(\nu \vec{y}^2 - \mu \vec{y}^1, g(\vec{y}^1, \vec{y}^2)\) where

\[
val = \begin{pmatrix}
val_{s_{q_1}} \\
val_{s_{q_1} \rightarrow s_{q_2}} \\
val_{s_{q_1} \rightarrow s_{q_3}} \\
val_{s_{q_2}} \\
val_{s_{q_2} \rightarrow s_{q_3}} \\
val_{s_{q_3}} \\
val_{s_{q_3} \rightarrow s_{q_4}}
\end{pmatrix}
\quad \text{and} \quad
g(\begin{pmatrix}
y_1 \\
y_2 \\
y_3 \\
y_4 \\
y_5 \\
y_6 \\
y_7 \\
y_8
\end{pmatrix}, \begin{pmatrix}
y_1 \\
y_2 \\
y_3 \\
y_4 \\
y_5 \\
y_6 \\
y_7 \\
y_8
\end{pmatrix}) = \begin{pmatrix}
y_5 \cdot y_5 \\
y_1 \cdot y_1 \\
y_1 \cdot y_5 \\
y_5 \cdot y_5 \\
y_1 \cdot y_1 \\
y_1 \cdot y_5 \\
y_5 \cdot y_5 \\
y_1 \cdot y_1
\end{pmatrix}
\]

By straightforward simplifications we obtain the system of fixed-point equations

\[
\begin{align*}
x_1 &= \frac{1}{4}x_2^2 + \frac{3}{4}x_1^2 \\
x_2 &= \frac{1}{4}x_2^3 + \frac{3}{4}x_1^3
\end{align*}
\]

in the two variables \(x_1\) and \(x_2\) (corresponding to the variables \(y_1\), representing \(s_{q_1}\), and \(y_5\), representing \(s_{q_2}\)). The execution\(^6\) of Algorithm 1 reveals that the solution of the system of equations is \((0, 0)\). Hence \(val_{s_{q_2}} = 0\) and this shows that the probability \(\mu(L_\infty)\) of the language \(L_\infty\) is 0.

### 5.4 Computing the measure of \(W_{i,k}\)

The family of regular languages \(W_{i,k}\), indexed by pairs \(i < k\) of natural numbers, constitutes a tool for investigating properties of regular languages using topological methods ([2, p. 329], see also [3, 4, 12]). The standard game automaton \(A_{i,k}\) over the language \(\Sigma_{i,k} = \{\forall, \exists\} \times \{i, i+1, \ldots, k-1\}\) accepting \(W_{i,k} \subseteq \mathcal{T}_{\Sigma_{i,k}}\) is defined as \(A_{i,k} = (Q, q_i, \delta, \pi)\) where \(Q = \{q_i, q_{i+1}, \ldots, q_k\}\), the initial state is \(q_i\) and, for each \(i \leq j \leq k\), the state \(q_j\) has priority \(\pi(q_j) = j\) and the transition function \(\delta\) is defined on \(q_j\) as in Figure 6. Our proof of Proposition 4, stated in the Introduction, goes by analyzing the system of fixed-point values.

---

\(^5\) See Section 5 of [15].

\(^6\) Details are presented in Section 5 of [15] along with a proof that the set \(L_\infty \subseteq \mathcal{T}_{a,b,c}\) is comeager.
equations associated with the game automaton $A_{i,k}$. Importantly, such a system consists of linear equations and not, as in the general case, of higher order polynomials. This system can be solved using standard techniques of linear algebra. A detailed proof of Proposition 4 can be found in Subsection 5.6 of [15].

6 Conclusion

In this work we presented an algorithm for computing the probability of regular languages defined by game automata. The Probability Problem in its full generality remains open. A possible direction for future research is to investigate approximations of regular languages by simpler regular languages. For example, given a regular language $L$ of trees, is it possible to find a regular language $G$ defined by a game automaton such that $L \triangle G = (L \setminus G) \cup (G \setminus L)$ is of probability 0, i.e. $L$ differs from $G$ by a set of probability 0? An effective answer to this question, that is an algorithm constructing a language $G$ from $L$, combined with the algorithm described in this paper would lead to a full solution to the Probability Problem.

References

On the Problem of Computing the Probability of Regular Sets of Trees


