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

Henryk Michalewski[1] and Matteo Mio[2]

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

October 7, 2015

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.

1 Introduction

Regular languages of trees are sets of infinite binary trees, labeled by letters from a finite alphabet Σ, definable by a formula of Monadic Second Order (MSO) logic interpreted over the full binary tree [23] or, equivalently, specified by an *alternating tree automaton* [18]. In this paper we consider the following problem. Suppose a Σ-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* μ (see Section 2 for definitions) on the space of Σ-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.

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 [10, 11] 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 *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 [20] 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*). Proposition 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 ω-words.

Lastly, we calculate the probability of all game languages $W_{i,k}$ (see [3, 12] and Subsection 5.6), 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 $\mathcal{A}$ we associate a MBP $\mathcal{M}$. The *value* of $\mathcal{M}$ can be computed and corresponds to the probability of the language recognized by $\mathcal{A}$. This reduction to MBP's is described in Sections 3 and 4.

[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.

The notion of MBP, as a special kind of *two-player stochastic meta-parity game* has been introduced by the second author in [15, 16] in order to interpret a probabilistic version of the modal μ-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 [22] 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 μ-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 [21] L. Staiger presented an algorithm for computing the *Hausdorff measure* of regular sets of ω-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 ω-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 [19] 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 ω. 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 μ on the product space Σ^I for I, a countable set of indices. The measure μ is uniquely defined by the assignment $\mu(\{t \in \Sigma^I \mid t(i_1) = a_1, \ldots, t(i_k) = a_k\}) = (\frac{1}{|\Sigma|})^k$ 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 Σ^V is denoted by $\mathcal{T}_\Sigma$ and an element $t \in \mathcal{T}_\Sigma$ is called a Σ-labeled tree, or just a Σ-tree. Intuitively, the stochastic processes associated with the coin–flipping measure μ on $\mathcal{T}_\Sigma$ generates an infinite Σ-tree by labeling each vertex with a randomly (uniformly) chosen label in Σ.

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 $\mathtt{int}(\mathtt{cl}(A)) = \emptyset$. 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 [18, 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.

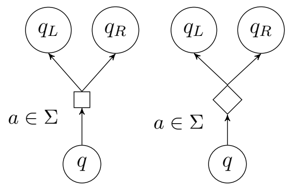
Definition 5 (Alternating Parity Tree Automaton)**.** *Given a finite set X, we denote with $\mathcal{DL}(X)$ the set of expressions e generated by the grammar $e ::= x \in X \mid e \wedge e \mid e \vee e$. An* alternating parity tree automaton *over a finite alphabet Σ is a tuple $\mathcal{A} = \langle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

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

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

We reserve the symbols $\top$ and $\bot$ for two special *sink* states having even and odd priority, respectively. The transition function is defined, for all $a \in \Sigma$, as $\delta(\top, a) = (L, \top) \wedge (R, \top)$ and $\delta(\bot, a) = (L, \bot) \wedge (R, \bot)$. Clearly every tree is accepted at the state $\top$ and rejected at $\bot$.

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 $e = (L, q_L) \vee (R, q_R)$ or $e = (L, q_L) \wedge (R, q_R)$ (see [10, 11] for more information about this class of automata). Transitions of a game automaton $\mathcal{A}$ can be schematically depicted as in the figure above with the left–hand and right–hand diagrams representing the transitions $(q,a) \to (L, q_L) \wedge (R, q_R)$ and $(q,a) \to (L, q_L) \vee (R, q_R)$, respectively.



Deterministic automata are a subfamily of game automata satisfying the stronger constraint that, for each $q \in Q$ and $a \in \Sigma$, the transition $\delta(q,a) = e$ has the form $e = (L, q_L) \wedge (R, q_R)$. Note that the sink states $\top$ and $\bot$ defined above have transitions satisfying this requirement.

3 Introduction to meta-parity games

In this Section we describe a class of stochastic processes called *Markov branching plays* (MBP's) [15, 16] which, as we will observe, is closely related to game automata and will provide a method for calculating 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 1, at the end of this section. The procedure for computing the probability of regular languages defined by game automata in presented as Algorithm 2 in the next section.

We assume familiarity with the standard concepts of Markov chain and two-player stochastic ($2\frac{1}{2}$-player) parity game (see, e.g., [6]). Ordinary $2\frac{1}{2}$-player parity games are played on directed graphs whose set of states is partitioned into Player 1, Player 2 and probabilistic states. A $2\frac{1}{2}$-player parity game with neither Player 1 nor Player 2 states can be identified with a *Markov chain*.

Two-player stochastic meta-parity games [15, 16] generalize $2\frac{1}{2}$-player parity games by allowing the directed graph to have two additional kinds of states called $\exists$–branching states and $\forall$–branching states. In this paper we will only consider $2\frac{1}{2}$-player meta-parity games with neither Player 1 nor Player 2 states. Such structures, which thus constitute a generalization of Markov chains, are called *Markov branching plays* (MBP's). In what follows we provide a quick description of MBP and refer to [15] for a detailed account.

Definition 6 (Markov Branching Play)**.** *A Markov branching play (MBP) is a structure $\mathcal{M} = \langle (S,E), (S_P, B_\exists, B_\forall), p, Par \rangle$ where:*

- *(S,E) is a directed graph with finite set of vertices S and transition relation E. We say that s' is a successor of s if $(s,s') \in E$. We assume that each vertex has at least one successor state in the graph (S,E).*
- *The triple $(S_P, B_\exists, B_\forall)$ is a partition of S into probabilistic, $\exists$-branching and $\forall$-branching states.*

- *The function $p : S_P \to (S \to [0,1])$ associates to each probabilistic state s a discrete probability distribution $p(s) : S \to [0,1]$ supported over the (nonempty) set of successors of s in the graph (S,E).*

- *Lastly, the function $Par : S \to \omega$ is the* parity (or priority) assignment*.*

Recall that a Markov chain represents the stochastic process associated with a *random infinite walk* on its set of states. A MBP represents the more involved stochastic process, described below, of generation of a random unranked and unordered *tree* T whose vertices are labeled by states of the MDP.

MBP's as Stochastic Processes: given a MBP $\mathcal{M} = \langle (S,E), (S_P, B_\exists, B_\forall), p, Par \rangle$ 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 $s \in B_\exists$ or $s \in B_\forall$ 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, $\exists$-branching and $\forall$-branching states are marked as circles, diamond and boxes, respectively. The first six initial steps of the stochastic process associated with $\mathcal{M}$ 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 $\exists$-branching state, the tree is extended by adding one new vertex for each successor of q_2 in $\mathcal{M}$, 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 $\frac{1}{3} \cdot \frac{2}{3} \cdot \frac{1}{2} = \frac{2}{18}$.

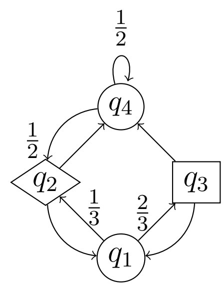


Figure 1: An example of a MBP.

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 ($\exists$ or $\forall$) branching state s has as many children as there are successors of s in the MBP.

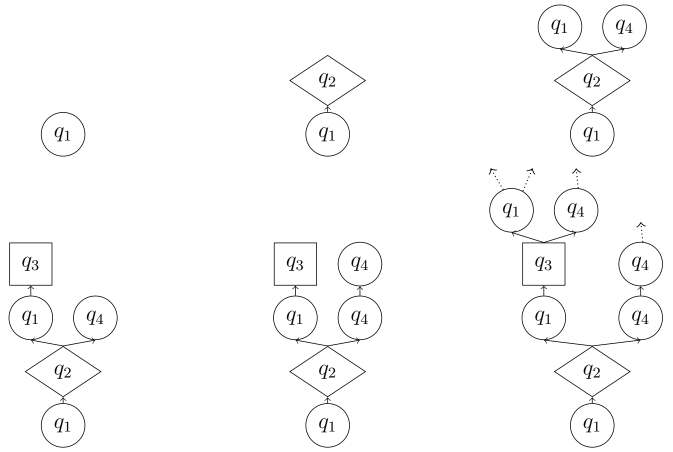


Figure 2: The stochastic process associated with the MBP in Figure 1.

The collection of branching plays in a MBP $\mathcal{M}$ starting from a state s is denoted by $\mathcal{BP}(\mathcal{M}, s)$. The set $\mathcal{BP}(\mathcal{M}, s)$ naturally carries a Polish topology making $\mathcal{BP}(\mathcal{M}, s)$ homeomorphic to the Cantor space (see, e.g., Definition 4.4 in [16]). The stochastic process associated to a MBP $\mathcal{M}$, specified on the previous page, can be naturally formalized by a probability measure $\mu_{\mathcal{M}}$ over the space $\mathcal{BP}(\mathcal{M}, s)$ of branching plays. See also Definition 4.7 in [16] for a formal definition.

Each branching play T can itself be viewed as an ordinary (infinite) two-player parity game $\mathcal{G}(T)$, played on the tree structure of T, by interpreting the vertices of T labeled by $\exists$-branching and $\forall$-branching states as under the control of Player $\exists$ and Player $\forall$, respectively. All other states (i.e., those labeled by a probabilistic state) have a unique successor in T to which the game $\mathcal{G}(T)$ progresses automatically. Lastly, the parity condition associated to each vertex corresponds to the parity assigned in $\mathcal{M}$ to the state labeling it. We denote with $\mathcal{W}_s$ the set of branching plays starting at s and winning for Player $\exists$, i.e., the set defined as: $\mathcal{W}_s = \big\{T \in \mathcal{BP}(\mathcal{M}, s) \mid \text{ Player } \exists \text{ has a winning strategy in } \mathcal{G}(T)\big\}$.

Definition 7 (Value of a MBP)**.** *The* value *of a MBP $\mathcal{M}$ at a state s, denoted by $val(\mathcal{M}, s)$, is the probability of generating a branching play winning for $\exists$ starting the stochastic process from the state s. Formally, $val(\mathcal{M}, s) = \mu_{\mathcal{M}}(\mathcal{W}_s)$.*

We remark that the above definition is valid because the set $\mathcal{W}_s$ is μ-measurable for every Borel measure μ on the space $\mathcal{BP}(\mathcal{M}, s)$ ([12]) and thus also for $\mu_{\mathcal{M}}$.

3.1 How to compute the value of a MBP

In this subsection we show how the values $val(\mathcal{M}, s)$ can be computed. The algorithm is based on a result of [15, 16], formulated as Theorem 10 below, characterizing such values as the solution of an appropriate system of (least and greatest) fixed-point equations. We first formulate Proposition 8 exposing a fixed-point property of the value of MBP's. Let us fix a MBP $\mathcal{M} = \langle (S, E), (S_P, B_\exists, B_\forall), p, Par\rangle$ with $S = \{s_1 \dots s_n\}$. To improve readability we just write val_i for $val(\mathcal{M}, s_i)$ and we denote by val the vector $val = (val_i)_{1\leq i \leq n}$ of length n. The symbols $\sum$ and $\prod$ denote the usual operations of sum and product on reals. We also use a "coproduct" operation defined as $\coprod_{i\in I} x_i = 1 - \prod_{i\in I} 1 - x_i$.

Proposition 8. *The equality $val = f(val)$ holds, where $f:[0,1]^n \to [0,1]^n$ is:*

$$\Big(f\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}\Big)_i = \begin{cases} \sum\limits_{\{j \mid (s_i,s_j)\in E\}} p(s_i)(s_j)\cdot x_j & \text{if } s_i \in S_P \\ \prod\limits_{\{j \mid (s_i,s_j)\in E\}} x_j & \text{if } s_i \in B_\forall \\ \coprod\limits_{\{j \mid (s_i,s_j)\in E\}} x_j & \text{if } s_i \in B_\exists \end{cases}$$

Proof. Here we sketch the main idea of the argument, for a formal proof, see Theorem 4.22 of [16]. 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 *independently* 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 $\coprod$ expression. Hence the vector val is one of the fixed-points of the function $f:[0,1]^n \to [0,1]^n$. $\square$

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, §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 $g:([0,1]^n)^c \to [0,1]^n$ is defined as follows:*

$$\Big(g\begin{pmatrix} x^l_1 \\ \vdots \\ x^l_n \end{pmatrix}, \dots, \begin{pmatrix} x^k_1 \\ \vdots \\ x^k_n \end{pmatrix}\Big)_i = \begin{cases} \sum\limits_{\{j \mid (s_i,s_j)\in E\}} p(s_i)(s_j)\cdot x_j^{Par(s_j)} & \text{if } s_i \in S_P \\ \prod\limits_{\{j \mid (s_i,s_j)\in E\}} x_j^{Par(s_j)} & \text{if } s_i \in B_\forall \\ \coprod\limits_{\{j \mid (s_i,s_j)\in E\}} x_j^{Par(s_j)} & \text{if } s_i \in B_\exists \end{cases}$$

The function g depends, like the function f, only on n variables $\{x_1^{Par(s_1)}, \dots, x_n^{Par(s_n)}\}$ appearing in the body of its definition. The input of g can indeed be regarded as the input of f divided into c baskets, where each variable x_i is put in the basket corresponding to the priority of s_i, for $1 \le i \le n$.

The set $[0,1]^n$, equipped with the pointwise order defined as

$$(x_1,\dots,x_n) \le (y_1,\dots,y_n) \Leftrightarrow \forall i.(x_i \le y_i),$$

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 μ-calculus notation (see, e.g., [1] and [15, 16]) to express systems of least and greatest fixed-points equations.

Theorem 10 ([15, Theorem 6.4.2])**.** *The following equality holds:*[2]

$$\begin{pmatrix} val_1 \\ \vdots \\ val_n \end{pmatrix} = \theta_k \begin{pmatrix} x_1^k \\ \vdots \\ x_n^k \end{pmatrix} . \cdots . \theta_l \begin{pmatrix} x_1^l \\ \vdots \\ x_n^l \end{pmatrix} . g(\begin{pmatrix} x_1^l \\ \vdots \\ x_n^l \end{pmatrix} , \ldots , \begin{pmatrix} x_1^k \\ \vdots \\ x_n^k \end{pmatrix})$$

where θ_i, for $l \leq i \leq k$ is a least-fixed point operator (μ) if i is an odd number and a greatest-fixed point operator if (ν) 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 [15] 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 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 [17]) the unique vector $val = (val_1, \ldots, val_n)$ 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 $(val_1, \ldots, val_n) \in \mathbb{R}^n$ is the unique vector of reals satisfying the formula $F(x_1, \ldots, x_n)$. By Tarski's quantifier elimination procedure [22], 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 $(val_1, \ldots, val_n)$, which can be extracted from G with standard methods, is a vector of algebraic numbers. In Section 5 we apply the procedure described above to a number of examples. □

input: a Markov Branching Play $\mathcal{M}$.
output: algebraic numbers $r_1, \ldots, r_n \in \mathbb{R}$ equal to $(val_1, \ldots, val_n)$.
begin
  $\mathcal{S} \leftarrow$ Generate system of fixed–point equations associated to $\mathcal{M}$
  $F(x_1, \ldots, x_n) \leftarrow$ Rewrite $\mathcal{S}$ to the corresponding first-order formula over $FO(\mathbb{R}, <, 0, 1, +, \times)$
  $G(x_1, \ldots, x_n) \leftarrow$ Apply quantifier elimination procedure to $F(x_1, \ldots, x_n)$
return the unique vector $(r_1, \ldots, r_n)$ satisfying $G(x_1, \ldots, x_n)$

Algorithm 1: computing the vector of values of a MBP.

[2] Theorem 6.4.2 of [15] 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 [15] is stated assuming the validity of the set-theoretic axiom $\mathrm{MA}_{\aleph_1}$, but as shown in [12], such assumption is not necessary and can thus be dropped.

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 $\mathcal{A}=(Q,q_0,\delta,\pi)$ over the alphabet Σ a corresponding MBP $\mathcal{M}=\langle (S,E),(S_P,B_\exists,B_\forall),p,Par\rangle$. The set S of states of $\mathcal{M}$ 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 $\delta(q,a)=(L,q_L)\vee(R,q_r)$, and a $\forall$–branching state $s_{q,a}$ for each pair (q,a) such that $\delta(q,a)=(L,q_L)\wedge(R,q_r)$. 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 $\delta(q,a)=(L,q_1)\vee(R,q_2)$ (resp. $\delta(q,a)=(L,q_1)\wedge(R,q_2)$).

Note that each state s_q, for $q\in Q$ has exactly $|\Sigma|$ successors and that each state $s_{q,a}$ has exactly[3] two successors. The assignment $p:S_P\to(S\to[0,1])$ 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 $Par:S\to\omega$ of the MBP $\mathcal{M}$ is defined as in the parity condition π of the game automaton $\mathcal{A}$ by the mapping $Par(s_q)=Par(s_{q,a})=\pi(q)$.

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

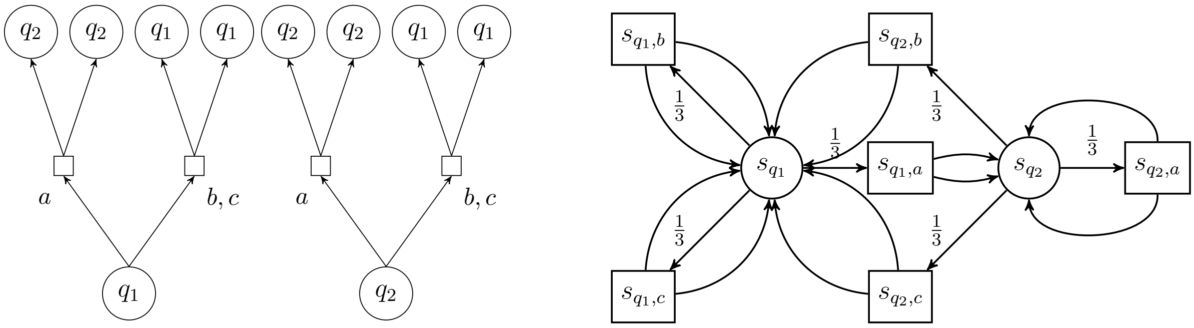


Figure 3: Transitions of the game automaton $\mathcal{A}$ and corresponding MBP $\mathcal{M}$.

The corresponding MBP $\mathcal{M}$ is schematically[4] 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.

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

[4] 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.

Theorem 12 (Correctness of Reduction)**.** *Let L be a regular language recognized by a game automaton $\mathcal{A}$ and let $\mathcal{M}$ be the MBP corresponding to $\mathcal{A}$. Then $\mu(L)=Val(\mathcal{M}, s_{q_0})$, where q_0 is the initial state of $\mathcal{A}$.*

Proof. (sketch) 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 Σ-trees $t\in\mathcal{T}_\Sigma$ and branching plays $T\in\mathcal{BP}(\mathcal{M}, s_{q_0})$. Furthermore, it follows directly from the definition of acceptance by $\mathcal{A}$ (see Section 2.2) and the definition of the set $\mathcal{W}_s$ (see Section 3) that t is accepted by $\mathcal{A}$ if and only if the corresponding branching play T is in $\mathcal{W}_s$. Lastly, due to the uniform assignment p of probabilities in $\mathcal{M}$, the coin-flipping measure μ on $\mathcal{T}_\Sigma$ and the probability measure $\mu_\mathcal{M}$ on $\mathcal{BP}(\mathcal{M}, s_{q_1})$ 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.

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

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

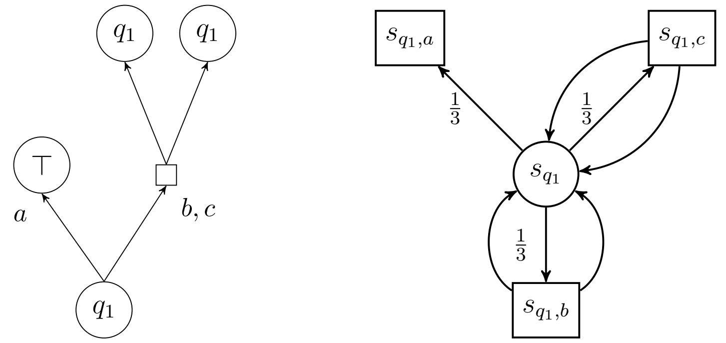
5 Examples

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 `qepcad` [5].

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

5.1 An introductory example

The language L_1 is recognized by the deterministic automaton in Figure 4 (left) defined as $\mathcal{A}_1=\langle\{q_1, \top\}, q_1, \delta_1, \pi\rangle$ 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 δ_1 is defined on q_1 as $\delta_1(q_1, a)=(L, \top)\wedge(R, \top)$ and $\delta_1(q_1, l)=(L, q_1)\wedge(R, q_1)$ for $l\in\{b, c\}$. We will compute the probability $\mu(L_1)$ using the procedure of Algorithm 2. As a first step we construct the MBP $\mathcal{M}_1$ corresponding to $\mathcal{A}_1$, as specified in Section 4. In order to improve readability, we have represented in Figure 4 (center) a simplified version of $\mathcal{M}_1$ 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 $\mathcal{M}_1$ has four states, all of priority 1. Following the procedure of Algorithm 2 we need to



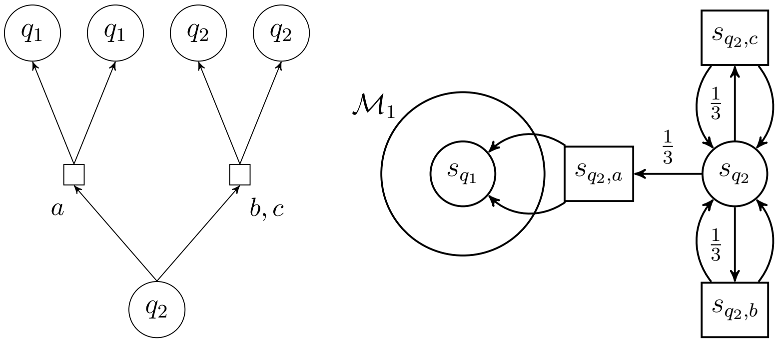
$$g\left(\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}\right) = \begin{pmatrix} \frac{1}{3}x_2 + \frac{1}{3}x_3 + \frac{1}{3}x_4 \\ 1 \\ x_1 \cdot x_1 \\ x_1 \cdot x_1 \end{pmatrix}$$

Figure 4: Automaton $\mathcal{A}_1$, MBP $\mathcal{M}_1$ and corresponding system of equations.

compute the values of the states of $\mathcal{M}_1$ using Algorithm 1. In accordance with Theorem 10, the fixed-point equation characterizing the vector $val = (val_{s_{q_1}}, val_{s_{q_1},a}, val_{s_{q_1},b}, val_{s_{q_1},c})$ of values of the states of $\mathcal{M}_1$ is $val = \mu\vec{x}.g(\vec{x})$, where g is defined as in Figure 4 (right). Then $val_{s_{q_1}}$ 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 $val_{s_{q_1}} = \frac{1}{2}$, and this is the output returned by Algorithm 2. Hence the probability of L_1 is $\mu(L_1) = \frac{1}{2}$.

5.2 Examples of regular languages having irrational probabilities

This subsection constitutes a proof of Proposition 2. The automaton $\mathcal{A}_2$ recognizing the language L_2 is defined as $\mathcal{A}_2 = \langle(\{q_1, q_2, \top\}, q_2, \delta_2, \pi)$ where q_2 is the initial state, the priority function is defined as $\pi(q_1) = \pi(q_2) = 1$ and the transition function δ_2 is defined on q_1 as the function δ_1 of the previous example, and on the state q_2 as $\delta(q_2, a) = (L, q_1) \wedge (R, q_1)$ and $\delta_2(q_2, l) = (L, q_2) \wedge (R, q_2)$, for $l \in \{b, c\}$. The transition δ_2 is shown in Figure 5 (left).



$$g\left(\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}\right) = \begin{pmatrix} \frac{1}{3}x_2 + \frac{1}{3}x_3 + \frac{1}{3}x_4 \\ \frac{1}{2} \cdot \frac{1}{2} \\ x_1 \cdot x_1 \\ x_1 \cdot x_1 \end{pmatrix}$$

Figure 5: Automaton $\mathcal{A}_2$, MBP $\mathcal{M}_2$ and corresponding system of equations.

The MBP $\mathcal{M}_2$ corresponding to $\mathcal{A}_2$ extends the MBP $\mathcal{M}_1$ 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 $\mathcal{A}_2$ is depicted in Figure 5 (center). Noting the four new states are not reachable by the other states already present in $\mathcal{M}_1$, we already know that $val_{s_{q_1}} = \frac{1}{2}$. Hence we can consider the simplified system of fixed-point equations $\mu\vec{x}.g(\vec{x})$ for calculating the values $val = (val_{q_2}, val_{q_2,a}, val_{q_2,b}, val_{q_2,c})$ where g is defined in Figure 5 (right). Hence the value val_{q_2} is the least solution in $[0,1]$ of the equation $x = \frac{1}{12} + \frac{2}{3}x^2$ and this is $val_{q_2} = \frac{1}{4}(3 - \sqrt{7})$ which is irrational and approximately equal to 0.088.

5.3 Automated computations for L_2

Above we computed the probability of L_2 using elementary ad hoc considerations. Here we will compute again this probability using Algorithm 2 and the tool `qepcad` [5]. As shown above, the measure of language L_2 is described by the following system of fixed–point equations

$$\begin{cases} x_1 & \stackrel{\mu}{=} & \frac{1}{3} + \frac{2}{3}x_1^2 \\ x_2 & \stackrel{\mu}{=} & \frac{1}{3}x_1^2 + \frac{2}{3}x_2^2 \end{cases}$$

This translates to the following `qepcad` code:

```
(A x1prime)
[
    [x1 = 1/3 + 2/3 x1^2] /\
    [x1prime = 1/3 + 2/3 x1prime^2 ==> x1prime >= x1 ]
].
```

The above formula $\phi_1(x_1)$, written in a formalized language of `qepcad`, expresses that x_1 is the smallest solution to the first equation. The solution generated by the tool is $2x_1 - 1 = 0$. Using this information we get the next formula $\phi_2(x_2)$, saying that x_2 is the least solution of the system of fixed point equations satisfying the inequality $x_2 \leq 1$.

```
(E x1)(A x2prime)(A x1prime)
[
    [2 x1 - 1 = 0] /\
    [x2 = 1/3 x1^2 + 2/3 x2^2] /\
    [x2 <= 1] /\
    [
        [
            [2 x1prime - 1 = 0] /\
            [x2prime = 1/3 x1prime^2 + 2/3 x2prime^2] /\
            [x2prime <= 1]
        ] ==>
        [x2prime >= x2]
     ]
].
```

The tool `qepcad` reduces the above formula $\phi_2(x_2)$ to a quantifier free expression

```
x2 - 1 < 0 /\ 8 x2^2 - 12 x2 + 1 = 0
```

which in turn can be easily solved analytically. The formula is satisfied by exactly one $x = \frac{1}{4}(3 - \sqrt{7})$, which is irrational and amounts approximately to 0.088.

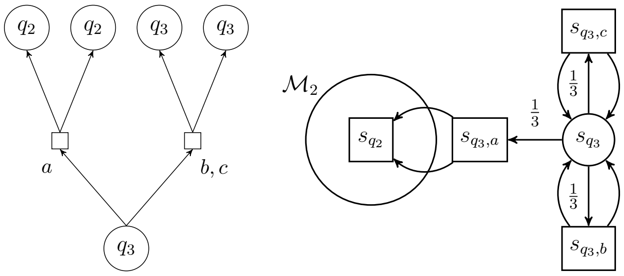
5.4 Automated computations for L_3

We show below that the probability of L_3 is $\mu(L_3) = \frac{1}{4}(3 - \sqrt{1 + 3\sqrt{7}})$ and thus not of the form $\frac{a+b\sqrt{c}}{d}$ 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.

The automaton $\mathcal{A}_3$ recognizing the language L_3 is defined as: $\mathcal{A}_3 = \langle \{q_1, q_2, q_3, \top\}, q_3, \delta_3, \pi \rangle$ where q_3 is the initial state, the priority function is defined as $\pi(q_1) = \pi(q_2) = \pi(q_3) = 1$

and the transition function δ_3 is defined on q_1 and q_2 as the function δ_2 of the previous example for the language L_2, and on the state q_3 as $\delta_3(q_3, a) = (L, q_2) \wedge (R, q_2)$ and $\delta_3(q_3, l) = (L, q_3) \wedge (R, q_3)$, for $l \in \{b, c\}$.

One can visualize the new state and transition as shown in Figure 6.



$$g\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} \frac{1}{3}x_2 + \frac{1}{3}x_3 + \frac{1}{3}x_4 \\ \frac{1}{4}(3-\sqrt{7}) \cdot \frac{1}{4}(3-\sqrt{7}) \\ x_1 \cdot x_1 \\ x_1 \cdot x_1 \end{pmatrix}$$

Figure 6: Language L_3 — the third approximation of the language L_∞. Transitions from q_1, q_2 are depicted in Figures 4,5. The initial state is q_3. The middle figure represents the MBP $\mathcal{M}_3$ corresponding to the automaton.

According to Theorem 8, the measure of language L_3 is equal to the value of the MBP $\mathcal{M}_3$, which is still simple enough, so that it can be analyzed directly by ad hoc methods. The MBP $\mathcal{M}_3$ corresponding to $\mathcal{A}_3$ extends the MBP $\mathcal{M}_2$ of the previous example with the probabilistic state s_{q_3} and the three $\forall$-branching states $s_{q_3,a}$, $s_{q_3,b}$ and $s_{q_3,c}$. The new part of the automaton $\mathcal{A}_3$ is depicted in Figure 5 (left). Noting the four new states are not reachable by the other eight states already present in $\mathcal{M}_2$, we already know that $val_{q_2} = \frac{1}{4}(3 - \sqrt{7})$. Hence we can consider the simplified system of fixed-point equations $\mu\vec{x}.g(\vec{x})$ for calculating the values $val = (val_{q_3}, val_{q_3,a}, val_{q_3,b}, val_{q_3,c})$ where g is defined in Figure 5 (right). Hence the value val_{q_3} is the least solution in $[0, 1]$ of the equation

$$x = \frac{1}{3}(\frac{1}{4}(3 - \sqrt{7}))^2 + \frac{2}{3}x^2,$$

which is equal to $\frac{1}{4}(3 - \sqrt{1 + 3\sqrt{7}})$.

We will obtain the above conclusion using our computational scheme and the tool `qepcad`. For this sake we will analyze the MBP $\mathcal{M}_3$ from scratch, first translating it into a system of 12 fixed-point equations.

$$val = \begin{pmatrix} val_{s_{q_1}} \\ val_{s_{q_1,a}} \\ val_{s_{q_1,b}} \\ val_{s_{q_1,c}} \\ val_{s_{q_2}} \\ val_{s_{q_2,a}} \\ val_{s_{q_2,b}} \\ val_{s_{q_2,c}} \\ val_{s_{q_3}} \\ val_{s_{q_3,a}} \\ val_{s_{q_3,b}} \\ val_{s_{q_3,c}} \end{pmatrix} \quad \text{and} \quad g\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \\ y_9 \\ y_{10} \\ y_{11} \\ y_{12} \end{pmatrix} = \begin{pmatrix} \frac{1}{3}y_2 + \frac{1}{3}y_3 + \frac{1}{3}y_4 \\ 1 \cdot 1 \\ y_1 \cdot y_1 \\ y_1 \cdot y_1 \\ \frac{1}{3}y_6 + \frac{1}{3}y_7 + \frac{1}{3}y_8 \\ y_1 \cdot y_1 \\ y_5 \cdot y_5 \\ y_5 \cdot y_5 \\ \frac{1}{3}y_{10} + \frac{1}{3}y_{11} + \frac{1}{3}y_{12} \\ y_5 \cdot y_5 \\ y_9 \cdot y_9 \\ y_9 \cdot y_9 \end{pmatrix}$$

After elementary reductions and reassigning of the variables $x_1 \leftarrow y_1, x_2 \leftarrow y_5, x_3 \leftarrow y_9$, the above equation is equivalent to the following system of equations

$$\begin{cases} x_1 & \stackrel{\mu}{=} & \frac{1}{3} + \frac{2}{3}x_1^2 \\ x_2 & \stackrel{\mu}{=} & \frac{1}{3}x_1^2 + \frac{2}{3}x_2^2 \\ x_3 & \stackrel{\mu}{=} & \frac{1}{3}x_2^2 + \frac{2}{3}x_3^2 \end{cases}$$

Like in the case of language L_2, we start the formal analysis in `qepcad` from the case of $\phi_1(x_1)$ characterizing the least solution to the first equation.

```
(A x1prime)
[
    [x1 = 1/3 + 2/3 x1^2] /\
    [x1prime = 1/3 + 2/3 x1prime^2 ==> x1prime >= x1 ]
].
```

The solver `qepcad` gives us solution $2x_1 - 1 = 0$, which we use to write down the formula $\phi_2(x_2)$, describing x_2 as the least solution to the second equation.

```
(E x1)(A x2prime)(A x1prime)
[
    [2 x1 - 1 = 0] /\
    [x2 = 1/3 x1^2 + 2/3 x2^2] /\
    [x2 <= 1] /\
    [
        [
            [2 x1prime - 1 = 0] /\
            [x2prime <= 1] /\
            [x2prime = 1/3 x1prime^2 + 2/3 x2prime^2]
        ] ==>
        [x2prime >= x2]
     ]
].
```

As we already computed before, the formula $\phi_2(x_2)$ translates to a quantifier free expression

```
x2 - 1 < 0 /\ 8 x2^2 - 12 x2 + 1 = 0
```

Below in formula $\phi_3(x_3)$ we characterize x_3 as the least solution to the third equation. In the formula we use the above quantifier–free translation of the formula $\phi_2(x_2)$. This leads to the following formulation of $\phi_3(x_3)$.

```
(E x2)(A x3prime)(A x2prime)
[
    [x2 - 1 < 0 /\ 8 x2^2 - 12 x2 + 1 = 0] /\
    [x3 = 1/3 x2^2 + 2/3 x3^2] /\
    [x3 <= 1] /\
    [
        [
            [x2prime - 1 < 0 /\ 8 x2prime^2 - 12 x2prime + 1 = 0] /\
            [x3prime <= 1] /\
            [x3prime = 1/3 x2prime^2 + 2/3 x3prime^2]
        ] ==>
        [x3prime >= x3]
     ]
].
```

The formula $\phi_3(x_3)$ is translated by `qepcad` to the following quantifier free expression

```
x3 - 1 < 0 /\ 256 x3^4 - 768 x3^3 + 832 x3^2 - 384 x3 + 1 = 0
```

The only real solution is $x_3 = \frac{1}{4}(3 - \sqrt{1 + 3\sqrt{7}})$, which is approximately 0.0026. This example shows in particular, that x_3 is not a quadratic irrational. In the case of the language L_2, the measure was expressible just with square roots of rational numbers.

5.5 Example of a comeager language of probability 0

This subsection constitutes a proof of Proposition 3. The regular language L_∞ 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

$$val = \begin{pmatrix} val_{s_{q_1}} \\ val_{s_{q_1,a}} \\ val_{s_{q_1,b}} \\ val_{s_{q_1,c}} \\ val_{s_{q_2}} \\ val_{s_{q_2,a}} \\ val_{s_{q_2,b}} \\ val_{s_{q_2,c}} \end{pmatrix} \quad \text{and} \quad g(\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ \_ \\ \_ \\ \_ \\ \_ \end{pmatrix}, \begin{pmatrix} \_ \\ \_ \\ \_ \\ \_ \\ y_5 \\ y_6 \\ y_7 \\ y_8 \end{pmatrix}) = \begin{pmatrix} \frac{1}{3}y_2 + \frac{1}{3}y_2 + \frac{1}{3}y_4 \\ y_5 \cdot y_5 \\ y_1 \cdot y_1 \\ y_1 \cdot y_1 \\ \frac{1}{3}y_5 + \frac{1}{3}y_1 + \frac{1}{3}y_1 \\ y_5 \cdot y_5 \\ y_1 \cdot y_1 \\ y_1 \cdot y_1 \end{pmatrix}$$

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

$$\begin{cases} x_1 & \overset{\mu}{=} & \frac{1}{3}x_2^2 + \frac{2}{3}x_1^2 \\ x_2 & \overset{\nu}{=} & \frac{1}{3}x_2^2 + \frac{2}{3}x_1^2 \end{cases}$$

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}) which has solution $(0, 0)$, as the automated computation below will reveal. Hence $val_{s_{q_2}} = 0$ and thus we will show that the probability $\mu(L_\infty)$ of the language L_∞ is 0.

The system of fixed-point equations in the variables x_1 and x_2 can be reduced to a formula in the language of real closed fields. First, we define a formula $\phi(x_1, x_2)$ which characterizes x_1 as the least fixed-point of the equation $x_1 = \frac{1}{3}x_2^2 + \frac{2}{3}x_1^2$ where x_2 is a parameter.

$$\psi_1(x_1, x_2) \overset{\text{def}}{=} (x_1 = \tfrac{1}{3}x_2^2 + \tfrac{2}{3}x_1^2) \wedge \forall x_1'.\big((x_1' = \tfrac{1}{3}x_2^2 + \tfrac{2}{3}(x_1')^2) \Rightarrow (x_1' \geq x_1)\big)$$

In the syntax of the tool `qepcad` (see [5] for more details) we have

```
(A x1prime)
[
    [x1 = 1/3 x2^2 + 2/3 x1^2] /\
    [x1prime = 1/3 x2^2 + 2/3 x1prime^2 ==> x1prime >= x1 ]
].
```

The solver `qepcad` produces an equivalent quantifier free formula

```
4 x1 - 3 <= 0 /\ x2^2 + 2 x1^2 - 3 x1 = 0
```

We now define a formula $\psi_2(x_2)$ characterizing x_2 as the greatest solution to the equation $x_2 = \frac{1}{3}x_2^2 + \frac{2}{3}x_1^2$ where x_1 is the unique number satisfying $\psi_1(x_1, x_2)$:

$$\begin{aligned}\psi_2(x_2) \stackrel{\text{def}}{=} \;& \exists x_1. \big(x_2 = \tfrac{1}{3}x_2^2 + \tfrac{2}{3}x_1^2\big) \;\wedge\; \psi_1(x_1, x_2)\big) \wedge \\ & \forall x_2' \big(\exists_{x_1'} (x_2' = \tfrac{1}{3}(x_2')^2 + \tfrac{2}{3}(x_1')^2) \;\wedge\; \psi_1(x_1', x_2')\big) \Rightarrow x_2' \leq x_2.\end{aligned}$$

In the `qepcad` syntax this can be expressed using the following code:

```
(E x1)(A x2prime)(A x1prime)
[
    [4 x1 - 3 <= 0 /\ x2^2 + 2 x1^2 - 3 x1 = 0] /\
    [x2 = 1/3 x2^2 + 2/3 x1^2] /\
    [
        [
            [4 x1prime - 3 <= 0 /\ x2prime^2 + 2 x1prime^2 - 3 x1prime = 0] /\
            [x2prime = 1/3 x2prime^2 + 2/3 x1prime^2]
        ] ==>
        [x2prime <= x2]
     ]
].
```

The solver `qepcad` translates $\psi_2(x_2)$ to the following quantifier free expression

```
x2 = 0
```

This means that the measure of the language L_∞ is 0. Our example is concluded by the following

Proposition 13. *The set $L_\infty \subseteq \mathcal{T}_{a,b,c}$ is comeager.*

Proof. The proof consists of unfolding of the definition of a comeager set. From the definition it is enough to verify that the complement of L is meager. Notice, that $t \notin L$ if and only if $t \notin L_n$ for certain $n \in \omega$. Hence, it is enough to prove that for every $n \in \omega$ the set $\mathcal{T}_{\{a,b,c\}} \setminus L_n$ is nowhere dense. That is, we have to show that for every non–empty open set $U \subset \mathcal{T}_{\{a,b,c\}}$ there exists $V \subset U$, an open subset, such that $V \subset L_n$. We can assume that U is a set of trees extending certain prefix p of the binary tree. As V we take a set of trees which extends a prefix q, where q is an extension of p by n frontiers consisting of letters a. $\square$

5.6 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 $\mathcal{A}_{i,k}$ over the language $\Sigma_{i,k} = \{\forall, \exists\} \times \{i, i+1, \ldots, k-1, k\}$ accepting $W_{i,k} \subseteq \mathcal{T}_{\Sigma_{i,k}}$ is defined as $\mathcal{A}_{i,k} = \langle Q, q_i, \delta, \pi\rangle$ 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 δ is defined on q_j as in Figure 7. Our proof of Proposition 4, stated in the Introduction, goes by analyzing the system of fixed-point equations associated with the game automaton $\mathcal{A}_{i,k}$. 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.

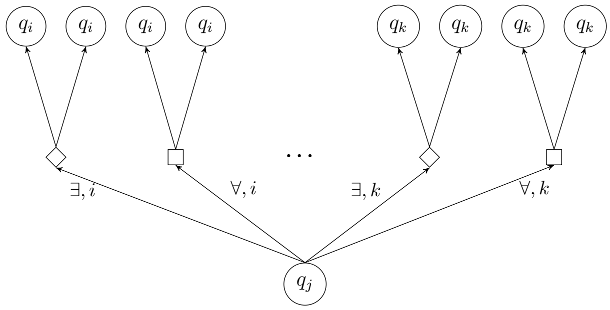


Figure 7: Transition of the automaton $\mathcal{A}_{i,k}$ recognizing $W_{i,k}$.

Proof. (of Proposition 4) In what follows we use notation $x \odot y = x + y - xy$ for denoting binary coproducts. Let $\mathcal{A}$ be the automaton depicted in Figure 7 accepting the language $W_{i,k}$ ($k > i$, $i = 0, 1$). The automaton has $k - i$ states $q_i, q_{i+1}, \dots, q_k$. We just consider the case $i = 1$ and k odd, since the other cases are analogous. We will show that the measure of $W_{1,k}$ is 0. From the definition of the game automaton $\mathcal{A}$ we infer the corresponding system of k fixed–point equations:

$$\begin{cases} x_1 & \stackrel{\mu}{=} & \frac{1}{2k}(x_1 \odot x_1 + x_1 \cdot x_1 + \ldots + x_k \odot x_k + x_k \cdot x_k) \\ x_2 & \stackrel{\nu}{=} & \frac{1}{2k}(x_1 \odot x_1 + x_1 \cdot x_1 + \ldots + x_k \odot x_k + x_k \cdot x_k) \\ & \vdots & \\ x_{k-1} & \stackrel{\nu}{=} & \frac{1}{2k}(x_1 \odot x_1 + x_1 \cdot x_1 + \ldots + x_k \odot x_k + x_k \cdot x_k) \\ x_k & \stackrel{\mu}{=} & \frac{1}{2k}(x_1 \odot x_1 + x_1 \cdot x_1 + \ldots + x_k \odot x_k + x_k \cdot x_k) \end{cases}$$

Note that the first and last equations are least-fixed point equations since 1 and k are odd by assumption. Unfolding the definition of $\odot$ we get

$$\begin{cases} x_1 & \stackrel{\mu}{=} & \frac{1}{2k}((2x_1 - x_1^2 + x_1^2) + \ldots + (2x_k - x_k^2 + x_k^2)) \\ x_2 & \stackrel{\nu}{=} & \frac{1}{2k}((2x_1 - x_1^2 + x_1^2) + \ldots + (2x_k - x_k^2 + x_k^2)) \\ & \vdots & \\ x_k & \stackrel{\mu}{=} & \frac{1}{2k}((2x_1 - x_1^2 + x_1^2) + \ldots + (2x_k - x_k^2 + x_k^2)) \end{cases}$$

This simplifies to the following system of equations

$$\begin{cases} x_1 & \stackrel{\mu}{=} & \frac{1}{2k}(2x_1 + \ldots + 2x_k) \\ x_2 & \stackrel{\nu}{=} & \frac{1}{2k}(2x_1 + \ldots + 2x_k) \\ & \vdots & \\ x_k & \stackrel{\mu}{=} & \frac{1}{2k}(2x_1 + \ldots + 2x_k) \end{cases}$$

Since k is odd, we are looking for the least $x_k \in [0, 1]$ satisfying the fixed–point formula. We will show that 0 is a fixed-point and, therefore, the desired least fixed-point.

Substituting 0 for x_k we get:

$$\begin{cases} x_1 & \overset{\mu}{=} & \frac{1}{k}(x_1 + \ldots + x_{k-1}) \\ x_2 & \overset{\nu}{=} & \frac{1}{k}(x_1 + \ldots + x_{k-1}) \\ & \vdots & \\ x_{k-1} & \overset{\nu}{=} & \frac{1}{k}(x_1 + \ldots + x_{k-1}) \\ 0 & = & \frac{1}{k}(x_1 + \ldots + x_{k-1}) \end{cases}$$

We are going to show that the solution of this system of fixed-point equations is the vector $x_1 = 0, x_2 = 0 \ldots, x_{k-1} = 0$. This will conclude the proof of Proposition 4.

To prove this, note that the solution is necessarily a solution of the linear system of equations

$$\begin{cases} x_1 & = & \frac{1}{k}(x_1 + \ldots + x_{k-1}) \\ x_2 & = & \frac{1}{k}(x_1 + \ldots + x_{k-1}) \\ & \vdots & \\ x_{k-1} & = & \frac{1}{k}(x_1 + \ldots + x_{k-1}) \\ 0 & = & \frac{1}{k}(x_1 + \ldots + x_{k-1}) \end{cases}$$

which, as we now prove, has only one solution. To prove this we now focus on the first $k-1$ equations and show that even without the last equation the only solution is $x_1 = 0, \ldots, x_{k-1} = 0$. This vector of numbers satisfies also the last equation $0 = \frac{1}{k}(x_1 + \ldots + x_{k-1})$. In the first step we move the left–hand side variables to the right–hand side.

$$\begin{cases} 0 & = & -\frac{k-1}{k}x_1 + \frac{1}{k}x_2 + \ldots + \frac{1}{k}x_{k-1} \\ 0 & = & \frac{1}{k}x_1 - \frac{k-1}{k}x_2 \ldots + \frac{1}{k}x_{k-1} \\ \vdots & & \\ 0 & = & \frac{1}{k}x_1 + \ldots + \frac{1}{k}x_{k-1} - \frac{k-1}{k}x_k \end{cases}$$

Hence we are getting a $(k-1) \times (k-1)$ matrix A_k with the following coefficients.

$$A_k = \begin{pmatrix} -\frac{k-1}{k} & \frac{1}{k} & \ldots & \frac{1}{k} \\ \frac{1}{k} & -\frac{k-1}{k} & \ldots & \frac{1}{k} \\ & \vdots & & \\ \frac{1}{k} & \frac{1}{k} & \ldots & -\frac{k-1}{k} \end{pmatrix}$$

This is a linear system of equations and therefore it is enough to verify that the determinant of the matrix is different than 0. Hence the Lemma below finishes the proof of Proposition 4. □

Lemma 14. *For every $k \geq 3$ holds*

$$\det(A_k) = (-1)^{k-1}\frac{1}{k}.$$

Before the proof let us consider a special of $k = 3$. Then $A_3 = \begin{pmatrix} -\frac{2}{3} & \frac{1}{3} \\ \frac{1}{3} & -\frac{2}{3} \end{pmatrix}$, hence $\det(A_3) = \frac{4}{9} - \frac{1}{9} = \frac{3}{9} = \frac{1}{3}$. Moreover, one can notice that if we do not limit attention to the case $x_k = 0$, then the system of equations has infinitely many solutions, because the matrix $\begin{pmatrix} -\frac{2}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & -\frac{2}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & -\frac{2}{3} \end{pmatrix}$ has determinant equal to 0.

Proof. (of Lemma 14)

We simplify the determinant of the $(k-1)\times(k-1)$ matrix A_k in order to have 0's in the first row on all positions, with except of the first one.

$$\det(A_k) = \det\begin{pmatrix} -\frac{k-1}{k} & \frac{1}{k} & \frac{1}{k} & \dots & \frac{1}{k} \\ \frac{1}{k} & -\frac{k-1}{k} & \frac{1}{k} & \dots & \frac{1}{k} \\ & \vdots & & & \\ \frac{1}{k} & \frac{1}{k} & \frac{1}{k} & \dots & -\frac{k-1}{k} \end{pmatrix}$$

$$= \det\begin{pmatrix} (-\frac{k-1}{k}-\frac{1}{k}) & (\frac{1}{k}+\frac{k-1}{k}) & 0 & \dots & 0 \\ \frac{1}{k} & -\frac{k-1}{k} & \frac{1}{k} & \dots & \frac{1}{k} \\ & \vdots & & & \\ \frac{1}{k} & \frac{1}{k} & \frac{1}{k} & \dots & -\frac{k-1}{k} \end{pmatrix}$$

$$= \det\begin{pmatrix} -1 & 1 & 0 & \dots & 0 \\ \frac{1}{k} & -\frac{k-1}{k} & \frac{1}{k} & \dots & \frac{1}{k} \\ & \vdots & & & \\ \frac{1}{k} & \frac{1}{k} & \frac{1}{k} & \dots & -\frac{k-1}{k} \end{pmatrix}$$

$$= \det\begin{pmatrix} -1 & 1-1 & 0 & \dots & 0 \\ \frac{1}{k} & (-\frac{k-1}{k}+\frac{1}{k}) & \frac{1}{k} & \dots & \frac{1}{k} \\ & \vdots & & & \\ \frac{1}{k} & (\frac{1}{k}+\frac{1}{k}) & \frac{1}{k} & \dots & -\frac{k-1}{k} \end{pmatrix}$$

$$= \det\begin{pmatrix} -1 & 0 & 0 & \dots & 0 \\ \frac{1}{k} & -\frac{k-2}{k} & \frac{1}{k} & \dots & \frac{1}{k} \\ & \vdots & & & \\ \frac{1}{k} & \frac{2}{k} & \frac{1}{k} & \dots & -\frac{k-1}{k} \end{pmatrix}.$$

Having 0's in the first row we unfold the determinant with respect to this row. The next determinant is of the size $(k-2)\times(k-2)$. Again, we aim to have 0's in the first row on all positions, with except of the first one.

$$(-1)\det\begin{pmatrix} -\frac{k-2}{k} & \frac{1}{k} & \frac{1}{k} & \dots & \frac{1}{k} \\ \frac{2}{k} & -\frac{k-1}{k} & \frac{1}{k} & \dots & \frac{1}{k} \\ & \vdots & & & \\ \frac{2}{k} & \frac{1}{k} & \frac{1}{k} & \dots & -\frac{k-1}{k} \end{pmatrix}$$

$$= (-1)\det\begin{pmatrix} -1 & 1 & 0 & \dots & 0 \\ \frac{2}{k} & -\frac{k-1}{k} & \frac{1}{k} & \dots & \frac{1}{k} \\ & \vdots & & & \\ \frac{2}{k} & \frac{1}{k} & \frac{1}{k} & \dots & -\frac{k-1}{k} \end{pmatrix}$$

$$= (-1)\det\begin{pmatrix} -1 & 1-1 & 0 & \dots & 0 \\ \frac{2}{k} & (-\frac{k-1}{k}+\frac{2}{k}) & \frac{1}{k} & \dots & \frac{1}{k} \\ & \vdots & & & \\ \frac{2}{k} & (\frac{1}{k}+\frac{2}{k}) & \frac{1}{k} & \dots & -\frac{k-1}{k} \end{pmatrix}$$

$$= (-1)\det\begin{pmatrix} -1 & 0 & 0 & \dots & 0 \\ \frac{2}{k} & -\frac{k-3}{k} & \frac{1}{k} & \dots & \frac{1}{k} \\ & \vdots & & & \\ \frac{2}{k} & \frac{3}{k} & \frac{1}{k} & \dots & -\frac{k-1}{k} \end{pmatrix}.$$

Now we are ready to again unfold the determinant with respect to the first row. In the following computations we deal with the matrix of the size $(k-3)\times(k-3)$.

$$(-1)^2\det\begin{pmatrix} -\frac{k-3}{k} & \frac{1}{k} & \frac{1}{k} & \dots & \frac{1}{k} \\ \frac{3}{k} & -\frac{k-1}{k} & \frac{1}{k} & \dots & \frac{1}{k} \\ & \vdots & & & \\ \frac{3}{k} & \frac{1}{k} & \frac{1}{k} & \dots & -\frac{k-1}{k} \end{pmatrix}$$

$$= (-1)^2\det\begin{pmatrix} -1 & 1 & 0 & \dots & 0 \\ \frac{3}{k} & -\frac{k-1}{k} & \frac{1}{k} & \dots & \frac{1}{k} \\ & \vdots & & & \\ \frac{3}{k} & \frac{1}{k} & \frac{1}{k} & \dots & -\frac{k-1}{k} \end{pmatrix}$$

$$= (-1)^2\det\begin{pmatrix} -1 & 1-1 & 0 & \dots & 0 \\ \frac{3}{k} & (-\frac{k-1}{k}+\frac{3}{k}) & \frac{1}{k} & \dots & \frac{1}{k} \\ & \vdots & & & \\ \frac{3}{k} & (\frac{1}{k}+\frac{3}{k}) & \frac{1}{k} & \dots & -\frac{k-1}{k} \end{pmatrix}$$

$$= (-1)^2\det\begin{pmatrix} -1 & 0 & 0 & \dots & 0 \\ \frac{3}{k} & -\frac{k-4}{k} & \frac{1}{k} & \dots & \frac{1}{k} \\ & \vdots & & & \\ \frac{3}{k} & \frac{4}{k} & \frac{1}{k} & \dots & -\frac{k-1}{k} \end{pmatrix} = \cdots$$

Finally, after $k-3$ steps, we are ending up with a determinant of the size 2×2:

$$(-1)^{k-3}\det\begin{pmatrix} -1 & 1 \\ \frac{k-2}{k} & -\frac{k-1}{k} \end{pmatrix} = (-1)^{k-3}(\frac{k-1}{k} - \frac{k-2}{k}) = (-1)^{k-3}\frac{1}{k}.$$

□

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

[1] A. Arnold and D. Niwiński. *Rudiments of μ-calculus.* Studies in Logic. North-Holland, 2001.

[2] André Arnold. The μ-calculus alternation-depth hierarchy is strict on binary trees. *ITA*, 33(4/5):329–340, 1999.

[3] André Arnold and Damian Niwiński. Continuous separation of game languages. *Fundamenta Informaticae*, 81:19–28, 2008.

[4] Julian C. Bradfield. The modal μ-calculus alternation hierarchy is strict. *Theor. Comput. Sci.*, 195(2):133–153, 1998.

[5] Christopher W. Brown. QEPCAD B: A program for computing with semi-algebraic sets using cads. *SIGSAM Bull.*, 37(4):97–108, 2003.

[6] Krishnendu Chatterjee. *Stochastic ω-Regular Games.* PhD thesis, University of California, Berkeley, 2007.

[7] Krishnendu Chatterjee, Marcin Jurdziński, and Thomas A. Henzinger. Quantitative stochastic parity games. In *Proc. of SODA*, pages 121–130, 2004.

[8] Taolue Chen, Klaus Dräger, and Stefan Kiefer. Model checking stochastic branching processes. In *Proceedings of MFCS*, volume 7468 of *Lecture Notes in Computer Science*, pages 271–282. Springer, 2012.

[9] Luca de Alfaro and Rupak Majumdar. Quantitative solution of omega-regular games. *Journal of Computer and System Sciences*, 68:374 – 397, 2004.

[10] Jacques Duparc, Alessandro Facchini, and Filip Murlak. Definable operations on weakly recognizable sets of trees. In *Proc. of FSTTCS*, pages 363–374, 2011.

[11] Alessandro Facchini, Filip Murlak, and Michal Skrzypczak. Rabin-Mostowski index problem: A step beyond deterministic automata. In *Proc. of LICS*, pages 499–508, 2013.

[12] Tomasz Gogacz, Henryk Michalewski, Matteo Mio, and Michal Skrzypczak. Measure properties of game tree languages. In *Proc. MFCS*, pages 303–314, 2014.

[13] A. S. Kechris. *Classical Descriptive Set Theory.* Springer Verlag, 1994.

[14] Henryk Michalewski and Matteo Mio. Baire Category Quantifier in Monadic Second Order Logic. In *Proc. of ICALP*, 2015.

[15] Matteo Mio. *Game Semantics for Probabilistic μ-Calculi.* PhD thesis, School of Informatics, University of Edinburgh, 2012.

[16] Matteo Mio. Probabilistic Modal μ-Calculus with Independent product. *Logical Methods in Computer Science*, 8(4), 2012.

[17] Matteo Mio and Alex Simpson. Łukasiewicz mu-calculus. In *Proc. of Workshop on Fixed Points in Computer Science*, volume 126 of *EPTCS*, 2013.

[18] David E. Muller and Paul E. Schupp. Simulating alternating tree automata by nondeterministic automata: New results and new proofs of the theorems of Rabin, McNaughton and Safra. *Theor. Comput. Sci.*, 141(1&2):69–107, 1995.

[19] Damian Niwiński. On the cardinality of sets of infinite trees recognizable by finite automata. In *Proc. MFCS*, 1991.

[20] Ludwig Staiger. Rich omega-words and monadic second-order arithmetic. In *Proc. of CSL*, pages 478–490, 1997.

[21] Ludwig Staiger. The Hausdorff measure of regular omega-languages is computable. *Bulletin of the EATCS*, 66:178–182, 1998.

[22] Alfred Tarski. *A Decision Method for Elementary Algebra and Geometry*. University of California Press, 1951.

[23] Wolfgang Thomas. Languages, automata, and logic. In *Handbook of Formal Languages*, pages 389–455. Springer, 1996.