Probabilistic Regular Expressions and MSO Logic on Finite Trees

Thomas Weidner*

Leipzig University, Institute of Computer Science, Leipzig, Germany
weidner@informatik.uni-leipzig.de

Abstract

We introduce probabilistic regular tree expressions and give a Kleene-like theorem for probabilistic tree automata (PTA). Furthermore, we define probabilistic MSO logic. This logic is more expressive than PTA. We define bottom-up PTA, which are strictly more expressive than PTA. Using bottom-up PTA, we prove a Büchi-like theorem for probabilistic MSO logic. We obtain a Nivat-style theorem as an additional result.

1998 ACM Subject Classification F.1.1 Models of Computation, G.3 Probability and Statictics

Keywords and phrases Probabilistic Regular Expressions, MSO Logic, Tree Automata

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2015.503

1 Introduction

Probabilistic tree automata (PTA) were introduced by Magidor [13] and Ellis [11] in the 1970s. These automata enjoy plentiful applications in the field of natural language processing, including parsing, deep language models, and machine translation. We consider the behaviour of a PTA as a function mapping a tree to a probability value. Recent research has already transferred the Kleene- and Büchi theorems on words to the probabilistic setting [4, 14, 17, 18] as well as to the weighted setting [7, 9]. The classical Nivat theorem [16] characterises regular tree transductions by decompositions in a regular tree language and homomorphisms. Nivat characterisations have attracted recent interest [1, 8]. In this work, we present probabilistic variants of these classical results for finite trees.

We introduce probabilistic regular tree expressions (PRTE). Compared to the existing regular tree expressions we use a different iteration operator $S^{\infty z}$, which we call infinity iteration. The usual Kleene-iteration involves a choice after every iteration step either to stop or to continue the iteration. Our iteration removes this ambiguity and forces the iteration to continue until there are no more variables to substitute.

In order to obtain a probabilistic extension of MSO logic, we add a second order expected value operator $\mathbb{E}_p X.\varphi$ to MSO logic. In the scope of this operator, formulas $x \in X$ are considered to be true with probability p. The semantics of the expected value operator is then defined as the expected value over all sets. It turns out that standard (top-down) PTA are not expressive enough to capture the semantics of this probabilistic MSO logic. Therefore, we introduce bottom-up PTA. These automata assign a probability to a state given all the states at the child nodes. Thus, bottom-up PTA are a generalisation of deterministic bottom-up tree automata, and are strictly more expressive than (top-down) PTA.

The main results of this paper are the following:

1. We prove that PRTE and top-down PTA are expressively equivalent.

* Partially supported by DFG Graduiertenkolleg 1763 (QuantLA).

35th IARCS Annual Conf. Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2015).
Editors: Prahladh Harsha and G. Ramalingam; pp. 503–516

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

2. We prove a Nivat-theorem, which states that the behaviours of PTA are exactly the functions which can be constructed from regular tree languages using operations like relabellings, intersections, and probability measures.
3. Using this Nivat-theorem, we show the expressive equivalence of probabilistic MSO logic and bottom-up PTA.

For the first result, we employ so-called substitution summable PTA, where variables can only be accepted in sink states. The use of our Nivat-theorem for the third result allows us to reuse the classical Büchi-theorem.

2 Preliminaries

Let $\mathbb{N}$ be the set $\{1, 2, 3, \ldots\}$ and $\mathbb{N}_0$ be $\mathbb{N} \cup \{0\}$. Any finite, non-empty set Σ is called an alphabet. By Σ^* we denote all words over Σ and by Σ^+ all words over Σ excluding the empty word ε.

A *rank alphabet* is a finite, non-empty set Σ with a function $ar\colon \Sigma \to \mathbb{N}_0$ which assigns to every symbol $f \in \Sigma$ its arity. For convenience we let $\Sigma_n = \{ f \in \Sigma \mid ar(f) = n \}$ for every $n \in \mathbb{N}_0$. We write Σ for (Σ, ar) if ar is understood.

A *tree over* Σ is a function $t\colon D \to \Sigma$ where $D \subseteq \mathbb{N}^*$ is a non-empty, finite, prefix-closed set such that $\{ i \in \mathbb{N} \mid xi \in D \} = \{1, \ldots, ar(t(x))\}$ for every $x \in D$. We write $\mathrm{dom}(t)$ for D and $\mathrm{dom}_A(t)$ for all $x \in \mathrm{dom}(t)$ with $t(x) \in A$. We denote by $\mathrm{leaf}(t)$ the set of all maximal positions with respect to the prefix order, and by $\mathrm{inner}(t)$ the set $\mathrm{dom}(t) \setminus \mathrm{leaf}(t)$. The set of all trees over Σ is written T_Σ.

Let $t \in \mathrm{T}_\Sigma$ and $x \in \mathrm{dom}(t)$. We write $t|_x$ for the *subtree of t rooted at x* given by $\mathrm{dom}(t|_x) = \{ y \mid xy \in \mathrm{dom}(t) \}$ and $t|_x(y) = t(xy)$. For $f \in \Sigma_n$ and $t_1, \ldots, t_n \in \mathrm{T}_\Sigma$ let $f(t_1, \ldots, t_n)$ be the tree $\bar{t}$ given by $\mathrm{dom}(\bar{t}) = \{\varepsilon\} \cup \bigcup_{i=1}^{n} i\,\mathrm{dom}(t_i)$, $\bar{t}(\varepsilon) = f$ and $\bar{t}(ix) = t_i(x)$ for $i \in \{1, \ldots, n\}$ and $x \in \mathrm{dom}(t_i)$. For an introduction into tree automata, regular tree expressions, and logic on finite trees, see [6].

By $\mathbb{1}_Y\colon X \to \{0, 1\}$ we denote the characteristic function of $Y \subseteq X$. For a countable, non-empty set X let $\Delta(X)$ denote the set of all *distributions on X*, i.e. all functions $d\colon X \to [0, 1]$ such that $\sum_{x \in X} d(x) = 1$. Let $\Delta_0(X) = \Delta(X) \cup \{\mathbb{0}_X\}$, where $\mathbb{0}_X$ is the functions which assigns 0 to every $x \in X$. We call any function $S\colon \mathrm{T}_\Sigma \to [0, 1]$ a probabilistic tree series or just a tree series.

▶ **Definition 1.** Let Σ be a rank alphabet. A *(top-down) probabilistic tree automaton (PTA)* is a quadruple $A = (Q, \delta, \mu, F)$ where

1. Q is a finite, non-empty set – the set of states,
2. $\delta = \bigcup_{n \geq 1} \delta_n$ where $\delta_n\colon Q \times \Sigma_n \to \Delta_0(Q^n)$ – the transition probability function
3. $\mu \in \Delta(Q)$ – the initial distribution,
4. $F \subseteq Q \times \Sigma_0$ – the acceptance condition.

The *behaviour of A* is a function $\|A\|\colon \mathrm{T}_\Sigma \to [0, 1]$ defined by

$$\|A\|(t) = \sum_{\substack{\rho\colon \mathrm{dom}(t) \to Q \\ (\rho(x), t(x)) \in F \text{ for all } x \in \mathrm{leaf}(t)}} \mu(\rho(\varepsilon)) \prod_{x \in \mathrm{inner}(t)} \delta(\rho(x), t(x))\big(\rho(x1), \ldots, \rho(x\,ar(t(x)))\big).$$

A state $q \in Q$ is called a *sink* if $\delta(q, f) = \mathbb{0}$ for all $f \in \Sigma$.

3 Probabilistic Regular Tree Expressions

Like with classical regular tree expressions, we introduce an additional set of variables which will be used to mark the positions where tree substitution can occur. Formally for a finite set V let $\mathrm{T}_\Sigma(V) = \mathrm{T}_{\Sigma'}$ where $\Sigma'_n = \Sigma_n$ for $n \geq 1$ and $\Sigma'_0 = \Sigma_0 \cup V$. The special notation for trees containing variables is used to emphasize the special role of variables.

3.1 Operations on Tree Series

Before we define the syntax and semantics of probabilistic regular tree expressions, we introduce the operations used in these expressions.

For two trees $s, t \in \mathrm{T}_\Sigma(V)$ and a set of variables $W \subseteq V$, we define $s \trianglelefteq_W t$ to hold if and only if t can be obtained by substituting all variables from W in s, i.e, $\mathrm{dom}(s) \subseteq \mathrm{dom}(t)$ and $s(x) = t(x)$ for all $x \in \mathrm{dom}_{(\Sigma \cup V) \setminus W}(s)$. Then, $\trianglelefteq_W$ is a quasi-order, which we call the *substitution order*. For $|W| = 1$ we even have that $\trianglelefteq_W$ is a partial order, i.e, it is also anti-symmetric.

▶ **Definition 2.** Let $S, T \colon \mathrm{T}_\Sigma(V) \to [0,1]$ and $z \in V$. We define the *concatenation* $S \cdot_z T$ of S and T by

$$\big(S \cdot_z T\big)(t) = \sum_{s \trianglelefteq_z t} S(s) \cdot \prod_{x \in \mathrm{dom}_z(s)} T(t|_x) \qquad \text{for all } t \in \mathrm{T}_\Sigma(V).$$

Note that this definition is the same as for weighted tree series as given in [7]. They also showed that this product is associative for a fixed variable z, i.e., $(R \cdot_z S) \cdot_z T = R \cdot_z (S \cdot_z T)$. This does not hold if two different variables are used.

It is easy to see, that $S \cdot_z T$ can assume values outside of $[0,1]$ if we allow arbitrary tree series S and T. Hence, we make the assumption that for any given tree t the tree series S is a distribution on the trees s which can be extended using tree substitution to obtain t. More formally:

▶ **Definition 3.** A tree series $S \colon \mathrm{T}_\Sigma(V) \to [0,1]$ is called *substitution summable* if

$$\sum_{s \trianglelefteq_V t} S(s) \leq 1$$

holds for all $t \in \mathrm{T}_\Sigma(V)$.

Restricting S to be substitution summable in Definition 2 assures that $S \cdot_z T$ is bounded by 1. In addition, substitution summability is preserved by $\cdot_z$.

▶ **Lemma 4.** *Let $S, T \colon \mathrm{T}_\Sigma(V) \to [0,1]$ and $z \in V$. If S is substitution summable, then $(S \cdot_z T)(t) \leq 1$ for all $t \in \mathrm{T}_\Sigma(V)$. Moreover, if T is also substitution summable, so is $S \cdot_z T$.*

Proof. The first claim is easy to see as the set of all s with $s \trianglelefteq_V t$ contains all trees s with $s \trianglelefteq_z t$. For the second statement let $t \in \mathrm{T}_\Sigma(V)$. We compute

$$\sum_{s \trianglelefteq_V t} \sum_{r \trianglelefteq_z s} S(r) \prod_{x \in \mathrm{dom}_z(r)} T(s|_x) = \sum_{r \trianglelefteq_V t} S(r) \prod_{x \in \mathrm{dom}_z(r)} \sum_{s_x \trianglelefteq_V t|_x} T(s_x) \leq 1.$$

Here, we applied the index transformation $(s, r) \mapsto (r, (s|_x)_{x \in \mathrm{dom}_z(r)})$, which is bijective map from $\{(s,r) \mid r \trianglelefteq_z s \trianglelefteq_V t\}$ to $\{(r, (s_x)_{x \in \mathrm{dom}_z(r)}) \mid r \trianglelefteq_V t \text{ and } s_x \trianglelefteq_V t|_x \text{ for all } x \in \mathrm{dom}_z(r)\}$. ◀

Next, we give a probabilistic iteration operation. The usual Kleene-iteration adds a choice after every step to either stop the iteration process or substitute the variable again. This non-deterministic choice cannot be easily modelled probabilistically. Therefore, we propose a slightly different notion of iteration, where this choice is not present.

▶ **Definition 5.** Let $S\colon \mathrm{T}_\Sigma(V) \to [0,1]$ and $z \in V$. We define the *infinity iteration* $S^{\infty z}$ by

$$S^{\infty z}(t) = \lim_{n\to\infty} S^{\cdot z n}(t)$$

for all $t \in \mathrm{T}_\Sigma$, where $S^{\cdot z 0} = \mathbb{1}_{\{z\}}$ and $S^{\cdot z n+1} = S \cdot_z S^{\cdot z n}$ for all $n \geq 0$.

▶ **Lemma 6.** *Let $S\colon \mathrm{T}_\Sigma(V) \to [0,1]$ substitution summable and $z \in V$. Then $S^{\infty z}(t)$ is well-defined for all $t \in \mathrm{T}_\Sigma(V)$, i.e., the limit always converges and attains values in $[0,1]$, and is again substitution summable.*

Proof. First consider the case that $z \notin t(\mathrm{dom}(t))$. We then have $S^{\cdot z (n+1)}(t) \geq S^{\cdot z n}(t)$ by the definition of tree series concatenation. Thus, the sequence $S^{\cdot z n}(t)$ is monotonically increasing and bounded by Lemma 4. Hence, the sequence converges.

Now let $z \in t(\mathrm{dom}(t))$. We may assume that $S(z) < 1$ as otherwise S would be equal to $\mathbb{1}_{\{z\}}$. Note that for any trees s', s with $s' \trianglelefteq_z s$ it holds that whenever $z \in s(\mathrm{dom}(s))$ also $z \in s'(\mathrm{dom}(s'))$. Hence, we obtain

$$S^{\cdot z n}(t) = \sum_{t_0 \trianglelefteq_z \cdots \trianglelefteq_z t_n = t} S(t_0) \prod_{i=1}^{n} \prod_{x \in \mathrm{dom}_z(t_{i-1})} S(t_i|_x) \leq \sum_{k=0}^{n} \sum_{\substack{t_0 \trianglelefteq_z \cdots \trianglelefteq_z t_n = t \\ |\{\, i \mid t_{i-1} \neq t_i \,\}| = k}} S(z)^{n-k}.$$

Let N be the finite number of trees s with $s \trianglelefteq_z t$. Thus, we can bound k in the above equation by N. Any chain $t_0 \trianglelefteq_z \cdots \trianglelefteq_z t_n = t$ can be uniquely identified by the choice of k positions where inequality occurs and the trees occurring at these positions. Hence, there are at most $\sum_{k=0}^{N} \binom{n}{k} N^k$ chains of length n. Therefore, $S^{\cdot z n}(t) \leq P(n) S(z)^{n-N}$ for some polynomial P of degree independent of n. Thus, $S^{\cdot z n}(t) \to 0$ as $n \to \infty$ and so $S^{\infty z}(t) = 0$.

By Lemma 4 we know that $S^{\cdot z n}$ is substitution summable for any $n \geq 1$. Let $t \in \mathrm{T}_\Sigma(V)$. We obtain

$$\sum_{s \trianglelefteq_V t} S^{\infty z}(s) = \sum_{s \trianglelefteq_V t} \lim_{n\to\infty} S^{\cdot z n}(s) = \lim_{n\to\infty} \sum_{s \trianglelefteq_V t} S^{\cdot z n}(s) \leq 1. \qquad \blacktriangleleft$$

▶ Remark. In [7] an alternative iteration for tree series was proposed: Suppose that $S\colon \mathrm{T}_\Sigma(V) \to \mathbb{K}$ is a weighted tree series, where $\mathbb{K}$ is a semiring, with $S(z) = 0_\mathbb{K}$. They define $S_z^{0,\mathrm{F}} = \mathbb{0}$, $S_z^{n+1,\mathrm{F}} = S \cdot_z (S_z^{n,\mathrm{F}} + \mathbb{1}_{\{z\}})$, and $S_z^{*,\mathrm{F}}(t) = S_z^{\mathrm{height}(t)+1,\mathrm{F}}(t)$. As can be seen from the definition of $S_z^{n+1,\mathrm{F}}$, there is a choice for every leaf labelled by z to either continue the iteration or to just stop and attach the weight one to this position. There is no such choice with $S^{\infty z}$ – the iteration always has to continue. Nevertheless, when no leaf is labelled z, the iteration in $S_z^{*,\mathrm{F}}$ cannot stop at a z labelled leaf and thus equals to $S^{\infty z}$, i.e., $S^{\infty z}(t) = S_z^{*,\mathrm{F}}(t)$ for all $t \in \mathrm{T}_\Sigma(V)$ with $z \notin t(\mathrm{dom}(t))$ if $S(z) = 0$ and S substitution summable.

The usual Kleene-iteration $L^{*,z}$ of a tree language L with $z \notin L$ can be characterised as the unique solution of the equation $X = L \cdot_z X \cup \{z\}$. An analogous fixed-point characterisation can also be given for $S^{\infty z}$:

▶ **Corollary 7.** *Let $z \in V$ and S be a substitution summable probabilistic tree series with $S(z) < 1$. Then, $S^{\infty z}$ is the unique solution of the equation $X = S \cdot_z X$.*

Proof. By Lemma 6, $S^{\infty z} = \lim_{n\to\infty} S^{\cdot_z n}$ is well-defined. Thus, $S^{\infty z}$ is a solution of $X = S \cdot_z X$. Consider a tree series T with $T = S \cdot_z T$, i.e., $T = S^{\cdot_z n} \cdot_z T$ for all $n \geq 0$. We obtain

$$\begin{aligned} T(t) &= \lim_{n\to\infty} \big(S^{\cdot_z n} \cdot_z T\big)(t) = \lim_{n\to\infty} \sum_{s \trianglelefteq_z t} S^{\cdot_z n}(s) \prod_{x \in \mathrm{dom}_z(s)} T(t|_x) \\ &= \sum_{s \trianglelefteq_z t} \Big(\lim_{n\to\infty} S^{\cdot_z n}(s)\Big) \prod_{x \in \mathrm{dom}_z(s)} T(t|_x) = (S^{\infty z} \cdot_z T)(t) = S^{\infty z}(t), \end{aligned}$$

where the last equality holds as $S^{\infty z} \cdot_z T = S^{\infty z}$. This is due to the proof of Lemma 6, where we established that $S^{\infty z}(t) = 0$ if t contains the symbol z at any leaf. ◀

3.2 Syntax and Semantics of Regular Expressions

The syntax of regular tree expressions is based on weighted regular tree expressions with two differences: First, we use infinity-iteration instead of Kleene-iteration, and second, we do not allow arbitrary summation of terms. Instead, we use two restricted rules for sums, which either decide based on the root symbol or model probabilistic branching. Furthermore, we do not allow (direct) summation of variables. Unfortunately, these restrictions remove the closure of the set of expressions under associativity, commutativity and distributivity. Hence, we explicitly add these rules to the syntax. Though these rules are not needed to add expressiveness, we include the rules nevertheless to make it possible to write more natural expressions. This is formalised below.

▶ **Definition 8.** The set pRTE of *probabilistic regular tree expressions* is the smallest set $\mathcal{E}$ that is closed under the following grammar rules (where $p \in [0,1]$, $\Sigma' \subseteq \Sigma$ and $z \in V$)

$$E ::= \mathbb{0} \mid z \mid \sum_{f\in\Sigma'} f(E,\dots,E) \mid pE + (1-p)E \mid E \cdot_z E \mid E^{\infty z},$$

and is also closed under the following identities. Each identity states that an expression containing the left side of an identity as a subexpression is in $\mathcal{E}$ if and only if the same expression, but with this subexpression replaced by the right side of the identity, is in $\mathcal{E}$. The following identities model associativity of $\cdot$, $+$ and $\cdot_z$, commutativity of $+$:

$$\begin{aligned} (E+F)+G &= E+(F+G), & E+F &= F+E, \\ (E \cdot_z F) \cdot_z G &= E \cdot_z (F \cdot_z G), & (p_1p_2)E &= p_1(p_2E). \end{aligned}$$

Next, we give the following distributivity identities:

$$\begin{aligned} (E+F) \cdot_z G &= E \cdot_z G + F \cdot_z G, & p(E+F) &= pE + pF, \\ f(\cdots, E+F, \cdots) &= f(\cdots, E, \cdots) + f(\cdots, F, \cdots), & (p_1+p_2)E &= p_1E + p_2E. \end{aligned}$$

Moreover, we add the following identities involving $\mathbb{0}$:

$$f(\cdots, \mathbb{0}, \cdots) = \mathbb{0}, \qquad \mathbb{0} \cdot_z E = \mathbb{0}.$$

The semantics of pRTE is defined inductively on the structure of the expression: Let $t \in \mathrm{T}_\Sigma(V)$ we let $\|\mathbb{0}\|(t) = 0$, $\|z\|(t) = \mathbb{1}_{\{z\}}(t)$ for $z \in \Sigma_0 \cup V$, and

$$\|f(E_1,\dots,E_n)\|(t) = \begin{cases} \prod_{i=1}^n \|E_i\|(t_i) & \text{if } t = f(t_1,\dots,t_n) \text{ for } f \in \Sigma \cup V \\ 0 & \text{otherwise} \end{cases}$$

$$\begin{aligned} \|E+F\|(t) &= \|E\|(t) + \|F\|(t) & \|pE\|(t) &= p\|E\|(t) \\ \|E \cdot_z f\|(t) &= (\|E\| \cdot_z \|F\|)(t) & \|E^{\infty z}\|(t) &= \|E\|^{\infty z}(t). \end{aligned}$$

Remark, that the syntax of pRTE is decidable. First check whether the input string is well-formed, then maximally expand it using the expanding distributivity rules (e.g. $(E+F)\cdot_z G \to E\cdot_z G + F\cdot_z G$). Finally, check the resulting string using the context-sensitive grammar arising from all rules except the expanding ones.

▶ **Example 9.** Let $\Sigma = \Sigma_2 \cup \Sigma_0$ with $\Sigma_2 = \{\mathsf{f}\}$ and $\Sigma_0 = \{\mathsf{a}, \mathsf{b}\}$. Furthermore, let y and z be variables. Consider the expression

$$E = \big(1/2\ \mathsf{f}(\mathsf{y},\mathsf{z}) + 1/2\ \mathsf{f}(\mathsf{z},\mathsf{y}) + \mathsf{a}\big)^{\infty_\mathsf{y}} \cdot_\mathsf{z} \big(\mathsf{f}(\mathsf{z},\mathsf{z}) + \mathsf{a} + \mathsf{b}\big)^{\infty_\mathsf{z}}.$$

Let the first factor be denoted by E_1 and the second factor by E_2. Then E_1 stochastically chooses a branch whose leaf nodes are labelled by a and z, as every variable y must be eventually substituted by a for the iteration to stop. Thus, E_1 assigns the probability $(1/2)^n$ to trees of the form $f_1(f_2(\cdots f_n(\mathsf{a})\cdots))$, where $f_i(t) = \mathsf{f}(t,\mathsf{z})$ or $f_i(t) = \mathsf{f}(\mathsf{z},t)$, and 0 to every tree not of this form. The expression E_2 assigns probability 1 to every tree. Thus, we obtain $\|E\|(t) = \sum_{x\in \mathrm{dom}_a(t)} (1/2)^{|x|}$.

3.3 Expressive equivalence to automata

The first part of this section will give an inductive construction to translate probabilistic regular tree expressions to top-down probabilistic tree automata. Afterwards, we show the converse direction. As a first step, we transfer the concept of substitution summability to automata.

▶ **Definition 10.** A probabilistic tree automaton $A = (Q, \delta, \mu, F)$ is called *substitution summable* if the $|V|+1$ sets F_{Σ_0}, $(F_{\{z\}})_{z\in V}$ are pairwise disjoint and every state in F_V is a sink state, where $F_W = \{\, q \in Q \mid (q,a) \in F \text{ for some } a \in W \,\}$ for $W \subseteq \Sigma_0 \cup V$.

▶ **Lemma 11.** *Let A be a PTA. If A is substitution summable, so is $\|A\|$.*

The class of substitution summable tree series is closed under the operations $\cdot_z$ and $^{\infty_z}$. The same statement holds for the class of substitution summable PTA.

▶ **Lemma 12.** *Let A_1, A_2 be PTA and A_1 be substitution summable. Then $\|A_1\| \cdot_z \|A_2\|$ can be recognized by a probabilistic tree automaton B. If A_2 is substitution summable, so is B.*

Proof. The construction is based on the weighted case, for details see [7]. The automaton B is the disjoint union of A_1, but with the states in $(F_1)_z$ removed, and A_2. The initial distribution of A_1 is used. Every transition into a state in $(F_1)_z$ is redirected to A_2 according to the initial distribution of A_2. Thus, the automaton B simulates runs of A_1 followed by runs of A_2. As A_1 is substitution summable, whenever a simulated run of A_1 could enter a state in F_z the only possible choice is to continue the run in A_2. The substitution summability of A_2 directly carries over to B. ◀

▶ **Lemma 13.** *Let A be a substitution summable PTA. Then $\|A\|^{\infty_z}$ is recognizable by a substitution summable PTA.*

Proof. Let $A = (Q, \delta, \mu, F)$ and $F_z = \{\, q \in Q \mid (q,z) \in F \,\}$. We may assume $\mu(F_z) < 1$. Let $\eta = \frac{1}{1-\mu(F_z)}$. We define the automaton $A' = (Q', \delta', \mu', F')$ by $Q' = Q \setminus F_z$, $\mu'(q) = \eta\mu(q)$, $F' = F \setminus F_z$, and

$$\delta'(q,f)(q_1,\ldots,q_n) = \sum_{r_1,\ldots,r_n\in Q} \delta(q,f)(r_1,\ldots,r_n) \prod_{i=1}^{n} \kappa(r_i,q_i),$$

where $\kappa(r,q) = \mathbb{1}_{\{q\}}(r) + \mathbb{1}_{F_z}(r)\eta\mu(q)$. The automaton A' simulates A until it can enter a state in F_z. As A is substitution summable the only possibility for A in such a state is to accept a leaf labelled with z. Instead, A' resets the simulated run of A using the initial distribution, thus starting a new iteration. The additional factor η is inserted to model arbitrary many substitutions of z by itself, each of them having the probability of $\mu(F_z)$. ◀

▶ **Lemma 14.** *Let E be a probabilistic regular tree expression. There is a substitution summable probabilistic tree automaton A such that $\|E\| = \|A\|$.*

Proof. We show that the set of expressions whose semantics is recognizable by an automaton satisfies all closure properties of pRTE and hence contains all expressions.

Clearly $\mathbb{0}$ and $\mathbb{1}_{\{z\}}$, for $z \in V$, are recognizable. Next, we consider the expression $E = \sum_{f\in\Sigma'} f(E_1^f, \dots, E_{ar(f)}^f)$ for some $\Sigma' \subseteq \Sigma$ and expressions E_i^f with $f \in \Sigma'$ and $1 \leq i \leq ar(f)$. Assume there are automata A_i^f with $\|A_i^f\| = \|E_i^f\|$ for each f and i. An automaton A recognizing E is constructed by taking the disjoint union of the automata A_f^i together with a new initial state q_0. If A reads a symbol $g \in \Sigma'$ with $ar(g) > 0$ in q_0, it simulates each automaton A_i^g at node i for $i = 1, \dots, ar(g)$. Furthermore, every symbol $a \in \Sigma' \cap \Sigma_0$ is accepted in q_0.

For $E = pE_1 + (1-p)E_2$ consider automata A_1 and A_2 recognizing $\|E_1\|$ and $\|E_2\|$, respectively. The automaton A is the disjoint union of A_1 and A_2, but with the initial distribution $p\mu_1 + (1-p)\mu_2$.

By Lemmas 12 and 13 we have the closure under tree concatenation and infinity iteration. Finally, note that the associativity, commutativity, and distributivity rules do not change the semantics of the expression and hence no automata construction is necessary. ◀

▶ **Example 15.** We consider the expression E from Example 9. Using the constructions described above we obtain a PTA recognizing $\|E\|$. The steps are shown in Figure 1:

(a) Probabilistic automata recognizing the constant series equal to 1

(b) The automata obtained for the expression $1/2\ (\mathsf{f}(\mathsf{y},\mathsf{z}) + \mathsf{a}) + 1/2\ (\mathsf{f}(\mathsf{z},\mathsf{y}) + \mathsf{a})$ using the constructions from Lemma 14.

(c) Lemma 13 is applied to the automaton from (b).

(d) Lemma 12 is applied to the automata from (c) and (a). This automaton recognizes $\|E\|$.

▶ **Lemma 16.** *Let A be a tree automaton. There is an expression E with $\|E\| = \|A\|$.*

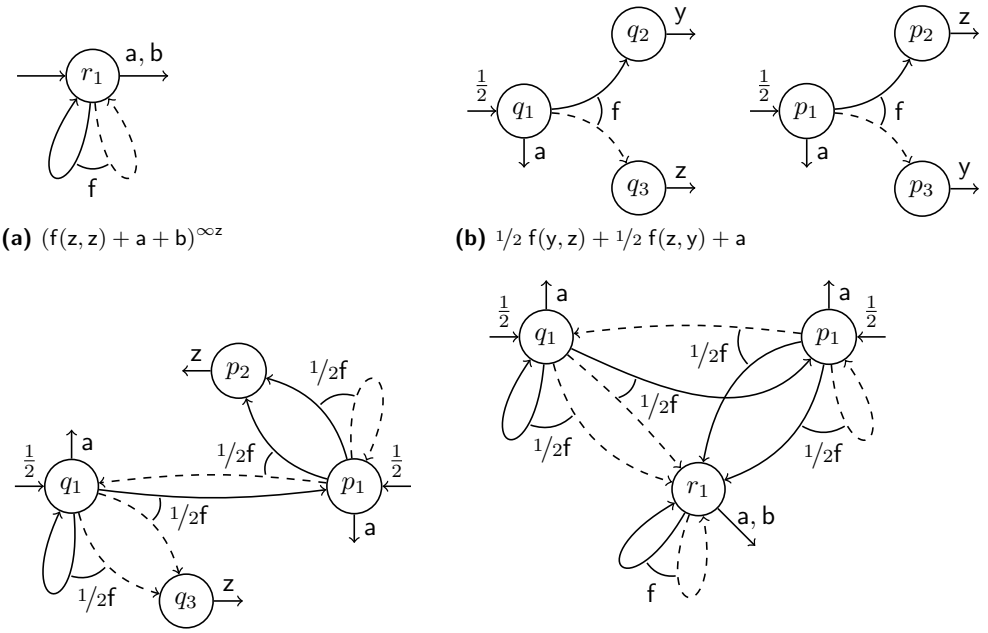
Proof. Let $A = (Q, \delta, \mu, F)$ and assume $Q = \{1, \dots, n\}$. Let $t \in \mathrm{T}_\Sigma(Q)$ and $i, k \in Q$. Define the set $R_i^k(t)$ to contain all runs $\rho\colon \mathrm{dom}(t) \to Q$ such that the following conditions hold:

1. $\rho(\varepsilon) = i$
2. $\rho(x) \leq k$ for all $x \in \mathrm{dom}_\Sigma(t) \setminus \{\varepsilon\}$
3. $\rho(x) = t(x)$ for all $x \in \mathrm{dom}_Q(t)$
4. $(\rho(x), t(x)) \in F$ for all $x \in \mathrm{dom}_{\Sigma_0}(t)$.

For a run ρ let $\mathrm{prob}(\rho) = \prod_{x\in\mathrm{inner}(t)} \delta(\rho(x), t(x))(\rho(x1), \dots, \rho(xn_x))$. We inductively construct expressions E_i^k over $\mathrm{T}_\Sigma(Q)$ such that

$$\|E_i^k\|(t) = \begin{cases} 0 & \text{if } t(\varepsilon) \in Q \text{ or } t(x) \leq k \text{ for a } x \in \mathrm{dom}_Q(t) \\ \sum_{\rho\in R_i^k(t)} \mathrm{prob}(\rho) & \text{otherwise.} \end{cases} \tag{1}$$

Intuitively, the index k is the largest state number that has already been handled in the construction. Trees must not contain handled states.



(a) $(\mathsf{f}(\mathsf{z},\mathsf{z}) + \mathsf{a} + \mathsf{b})^{\infty\mathsf{z}}$ **(b)** $1/2\ \mathsf{f}(\mathsf{y},\mathsf{z}) + 1/2\ \mathsf{f}(\mathsf{z},\mathsf{y}) + \mathsf{a}$

(c) $(1/2\ \mathsf{f}(\mathsf{y},\mathsf{z}) + 1/2\ \mathsf{f}(\mathsf{z},\mathsf{y}) + \mathsf{a})^{\infty\mathsf{y}}$ **(d)** $(1/2\ \mathsf{f}(\mathsf{y},\mathsf{z}) + 1/2\ \mathsf{f}(\mathsf{z},\mathsf{y}) + \mathsf{a})^{\infty\mathsf{y}} \cdot_{\mathsf{z}} (\mathsf{f}(\mathsf{z},\mathsf{z}) + \mathsf{a} + \mathsf{b})^{\infty\mathsf{z}}$

A transition $\eta = \delta(q, f)(q_1, q_2)$ is drawn as two arrows connected by an arc near the source state q. The arc is labelled by the probability η the letter f. The solid arrow leads to the first state q_1 and the dashed arrow to the second state q_2.

Figure 1 Inductive construction of a tree automaton.

For E_i^0 only trees of height at most 1 can have non-zero values. Thus, E_i^0 can be given directly for every $i \in Q$:

$$\begin{aligned} E_i^0 &= \sum_{\substack{c\colon \Sigma_{>0} \to Q^* \\ |c(f)| = ar(f)}} \left(\prod_{f \in \Sigma_{>0}} \delta(i, f)(c(f)) \right) \left(\sum_{f \in \Sigma_{>0}} f(c(f)) + \sum_{\substack{a \in \Sigma_0 \\ (i,a) \in F}} a \right) \\ &= \sum_{\substack{a \in \Sigma_0 \\ (i,a) \in F}} a + \sum_{f \in \Sigma_{>0}} \sum_{q_1, \ldots, q_{ar(f)} \in Q} \delta(i, f)(q_1, \ldots, q_{ar(f)})\ f(q_1, \ldots, q_{ar(f)}), \end{aligned}$$

where the first line can be directly constructed using the syntax from Definition 8 and the second line is obtained using distributivity and commutativity.

Now, assume the expressions E_i^k have already been constructed. Explicit calculation shows that the expression E_i^{k+1} defined by $E_i^{k+1} = E_i^k \cdot_{k+1} \left(E_{k+1}^k\right)^{\infty(k+1)}$ actually satisfies (1). We obtain the desired expression $E = \sum_{q \in Q} \mu(q) E_q^n$. ◀

Altogether, we have proven the following theorem:

▶ **Theorem 17.** *Let $S\colon \mathrm{T}_\Sigma \to [0,1]$ be a probabilistic tree series. The following statements are equivalent:*

1. *$S = \|A\|$ for some probabilistic tree automaton A,*
2. *$S = \|E\|$ for some probabilistic regular tree expression E.*

4 Probabilistic MSO Logic on Trees

In the first part of this section we give the formal definition of probabilistic MSO logic, basic properties, and an example. To prove the equivalence to probabilistic automata in the third part, we beforehand introduce bottom-up probabilistic tree automata in the second part and prove a Nivat-style theorem.

4.1 Syntax and Semantics of Probabilistic MSO logic

For the rest of this section fix a countable set of first order variables $\mathcal{V}_1$ and a countable set of second order variables $\mathcal{V}_2$. Let $\mathcal{V} = \mathcal{V}_1 \dot\cup \mathcal{V}_2$. Let $t \in \mathrm{T}_\Sigma$ be a tree. A *t-assignment* is a mapping $\sigma\colon \mathcal{V} \to \mathrm{dom}(t) \cup 2^{\mathrm{dom}(t)}$ such that $\sigma(\mathcal{V}_1) \subseteq \mathrm{dom}(t)$ and $\sigma(\mathcal{V}_2) \subseteq 2^{\mathrm{dom}(t)}$. We denote by $\sigma[x \mapsto a]$ the updated assignment which maps x to a and agrees with σ everywhere else.

▶ **Definition 18.** The set of all probabilistic MSO formulas φ is given in BNF by

$$\psi ::= \mathrm{label}_f(x) \mid \mathrm{edge}_i(x,y) \mid x \in X \mid \neg\psi \mid \psi \wedge \psi \mid \forall x.\psi \mid \forall X.\psi$$
$$\varphi ::= \psi \mid \varphi \wedge \varphi \mid \neg\varphi \mid \mathbb{E}_p X.\varphi\,,$$

where $f \in \Sigma$, $i \in \mathbb{N}$, $x \in \mathcal{V}_1$, $X \in \mathcal{V}_2$, and $p \in [0,1]$. The formulas generated by ψ are called Boolean formulas. The set $\mathrm{free}(\varphi)$ is defined as usual for Boolean MSO, conjunction, and negation. Additionally, we define $\mathrm{free}(\mathbb{E}_p X.\varphi) := \mathrm{free}(\varphi) \setminus \{X\}$.

The semantics of a formula φ is a function $[\![\varphi]\!]$ which maps a pair (t,σ), where t is a tree and σ is a t-assignment, to a probability value. The inductive definition is as follows:

$$[\![\psi]\!](t,\sigma) = \begin{cases} 1 & \text{if } (t,\sigma) \models \psi \\ 0 & \text{otherwise} \end{cases} \qquad [\![\varphi_1 \wedge \varphi_2]\!](t,\sigma) = [\![\varphi_1]\!](t,\sigma) \cdot [\![\varphi_2]\!](t,\sigma)$$
$$[\![\neg\varphi]\!](t,\sigma) = 1 - [\![\varphi]\!](t,\sigma)$$
$$[\![\mathbb{E}_p X.\varphi]\!](t,\sigma) = \sum_{M \subseteq \mathrm{dom}(t)} [\![\varphi]\!](t, \sigma[X \mapsto M]) \cdot p^{|M|}(1-p)^{|\mathrm{dom}(t)\setminus M|},$$

where $(t,\sigma) \models \psi$ is the usual satisfaction relation for classical MSO logic.

The semantics of the conjunction and negation are motivated from probability theory by the probabilities of the intersection of independent events and the complementary event, respectively. The semantics of $\mathbb{E}_p X.\varphi$ is as follows: We choose a set $M \subseteq \mathrm{dom}(t)$ using a sequence of independent Bernoulli experiments, i.e, for every tree position an unfair coin is tossed to decide whether to include this position in the set or not. For every such set the probability whether φ holds is computed. Finally, the expected value of these probabilities is calculated.

Disjunction can be defined as usual: $\varphi_1 \vee \varphi_2 = \neg(\neg\varphi_1 \wedge \neg\varphi_2)$. The semantics is then given by $[\![\varphi_1 \vee \varphi_2]\!] = [\![\varphi_1]\!] + [\![\varphi_2]\!] - [\![\varphi_1]\!][\![\varphi_2]\!]$. This resembles the well-known identity for the probability of the union of independent events.

We write $\varphi_1 \equiv \varphi_2$ if $[\![\varphi_1]\!] = [\![\varphi_2]\!]$. The following identities are valid:

$$\psi \wedge (\varphi_1 \vee \varphi_2) \equiv (\psi \wedge \varphi_1) \vee (\psi \wedge \varphi_2) \qquad \mathbb{E}_p X.\neg\varphi \equiv \neg\mathbb{E}_p X.\varphi$$
$$\mathbb{E}_p X.\mathbb{E}_q Y.\varphi \equiv \mathbb{E}_q X.\mathbb{E}_p Y.\varphi \qquad \mathbb{E}_p X.\varphi \equiv \varphi \text{ if } X \notin \mathrm{free}(\varphi),$$
$$\mathbb{E}_p X.(\varphi_1 \wedge \varphi_2) \equiv (\mathbb{E}_p X.\varphi_1) \wedge \varphi_2 \text{ if } X \notin \mathrm{free}(\varphi_2)$$

where φ, φ_1, φ_2 are arbitrary probabilistic MSO formulas and ψ is Boolean.

Using these identities, the following statement can be shown:

▶ **Lemma 19.** *Let φ be a probabilistic MSO formula. There is a Boolean MSO formula ψ, second order variables $X_1, \dots, X_n$, and probability values $p_1, \dots, p_n$ such that $\varphi \equiv \mathbb{E}_{p_1} X_1 \cdots \mathbb{E}_{p_n} X_n.\psi$.*

▶ **Example 20.** Let $\Sigma = \Sigma_2 \cup \Sigma_0$ with $\Sigma_2 = \{\mathsf{g}\}$ and $\Sigma_0 = \{\mathsf{a}, \mathsf{b}\}$. We consider a search for a leaf labelled by a. This search works by iterating the leafs of the tree left to right. At every visited leaf the search stops with probability p. It is successful if the label of the node, where the search stopped, is a. This process can be modelled by the following formula:

$$\varphi = \mathbb{E}_p X.\exists x.x \in X \wedge \mathrm{label}_{\mathsf{a}}(x) \wedge \mathit{leaf}(x) \wedge \big(\forall y.(y \in X \wedge \mathit{leaf}(y)) \implies x \sqsubseteq_{\mathrm{DF}} y\big),$$

where $\sqsubseteq_{\mathrm{DF}}$ is the depth-first traversal order and *leaf* the predicate whether a node is a leaf, both are MSO definable. The Boolean part of φ expresses that the minimal leaf position in X is labelled by a. For the semantics we obtain

$$[\![\varphi]\!](t) = \sum_{x \in \mathrm{leaf}_{\mathsf{a}}(t)} p \cdot (1-p)^{n_x} \text{ where } n_x = |\{y \in \mathrm{leaf}(t) \mid y \sqsubseteq_{\mathrm{DF}} x,\, y \neq x\}|.$$

The tree series $[\![\varphi]\!]$ from Example 20 is not recognizable by a top-down probabilistic tree automaton. Intuitively, when a top-down automaton reaches a leaf node, there is no information available whether there are a-labelled leafs to the left of this node. Therefore, we will define a more expressive probabilistic automata model in the next section.

▶ Remark. The syntax of probabilistic MSO was chosen to be minimal. In fact, some constructs known from weighted MSO logics can be expressed in probabilistic MSO as syntactic macros, i.e., they can be transformed to the syntax of Definition 18.

Multiplication by constants: Consider two PMSO formulas φ and ψ and a probability value p. The formula $\mathbb{E}_p X.(\exists x.\mathit{root}(x) \wedge (x \in X \implies \varphi) \wedge (x \notin X \implies \psi)$, where x and X are new variable symbols and $\mathit{root}(x)$ is a MSO formula, which checks if x is the root position, has as semantics $p[\![\varphi]\!] + (1-p)[\![\psi]\!]$.

Generalised universal first-order quantification: Weighted logics allows universal first order quantification to be applied to arbitrary formulas. We can also introduce such an operator to probabilistic logic. Let φ be a probabilistic MSO formula. The semantics of $\forall x.\varphi$ is given by $[\![\forall x.\varphi]\!](t,\sigma) = \prod_{a \in \mathrm{dom}(t)} [\![\varphi]\!](t, \sigma[x \mapsto a])$. As in the weighted case, $\forall x.$ does not preserve recognizability when arbitrary formulas φ are allowed. Thus, φ is restricted to so-called step formulas. A step formula is build from Boolean MSO formulas and probability constants using only the Boolean operations. Thus, every step formula φ can be written as $\varphi \equiv \bigwedge_{i=1}^{n} (\psi_i \implies p_i)$. We introduce new second order variables $X_1, \dots, X_n$ and define a formula φ' by

$$\varphi' \coloneqq \mathbb{E}_{p_1} X_1. \cdots \mathbb{E}_{p_n} X_n. \forall x. \bigwedge_{i=1}^{n} (\psi_i \implies x \in X_i).$$

It can be shown that $[\![\varphi']\!] = [\![\forall x.\varphi]\!]$ holds.

4.2 A Nivat Theorem

We introduce bottom-up deterministic tree automata, which are a generalisation of bottom-up deterministic tree automata to the probabilistic setting.

▶ **Definition 21.** A *bottom-up probabilistic tree-automaton* is a triple $A = (Q, \delta, F)$ where
1. Q is a finite, non-empty set – the set of states,
2. $\delta = \bigcup_{n\geq 0} \delta_n$ where $\delta_n \colon \Sigma \times Q^n \to \Delta(Q)$ – the transition probability function,
3. $F \subseteq Q$ – the set of final states.

The semantics is given by $\|A\|(t) = \sum_{q\in F} \delta_q(t)$, where

$$\delta_q\big(f(t_1,\dots,t_n)\big) = \sum_{q_1,\dots,q_n\in Q} \delta(f,q_1,\dots,q_n)(q) \prod_{i=1}^{n} \delta_{q_i}(t_i).$$

Although bottom-up PTA are a natural generalisation of deterministic bottom-up tree automata to the probabilistic setting, we only found one other reference to this model [12].

Before we state our Nivat theorem, we introduce some notation: Let Γ be another rank alphabet. A mapping $h \colon \Gamma \to \Sigma$ is called a *relabelling* if $ar_\Gamma(g) = ar_\Sigma(h(g))$ for all $g \in \Gamma$. A relabelling extends uniquely to a function $h \colon \mathrm{T}_\Gamma \to \mathrm{T}_\Sigma$ by $\mathrm{dom}(h(t)) = \mathrm{dom}(t)$ and $h(t)(x) = h(t(x))$ for all $x \in \mathrm{dom}(t)$. Given a finite set M, we can interpret $M \times \mathbb{N}_0$ as an (infinite) rank alphabet by defining $ar((m,k)) = k$. For a tree domain D, a finite set M and a distribution d on M, we define a probability measure $\Pr_d^D$ on $\{t \in \mathrm{T}_{M\times\mathbb{N}_0} \mid \mathrm{dom}(t) = D\}$ by $\Pr_d^D(\{t\}) = \prod_{x\in D} d(\pi_1(t(x)))$, where π_1 is the projection on the first component. Given a relabelling $g \colon \Sigma \to M \times \mathbb{N}_0$ we let $(\Pr_d^D \circ g)(X) = \Pr_d^D(g(X))$ for all $X \subseteq \{t \in \mathrm{T}_\Sigma \mid \mathrm{dom}(t) = D\}$. We write $\Pr_d$ instead of $\Pr_d^D$ if D is understood.

▶ **Theorem 22.** *Let $S \colon \mathrm{T}_\Sigma \to [0,1]$ be a probabilistic tree series.*
1. *S is the behaviour of a bottom-up probabilistic tree automaton if and only if there are*
   - **(a)** *a finite rank alphabet Γ and a finite set M,*
   - **(b)** *a distribution d on M,*
   - **(c)** *relabellings $h \colon \Gamma \to \Sigma$ and $g \colon \Gamma \to M \times \mathbb{N}_0$,*
   - **(d)** *a regular tree language $L \subseteq \mathrm{T}_\Gamma$,*

   such that for all $t \in \mathrm{T}_\Sigma$

$$\|A\|(t) = (\Pr_d \circ g)\big(h^{-1}(\{t\}) \cap L\big)\,. \tag{2}$$

2. *S is the behaviour of a top-down probabilistic tree automaton if and only if conditions a – d hold and additionally*
   - **(e)** *L is top-down deterministic recognizable,*
   - **(f)** *the mapping $\Gamma \to \Sigma \times M$ defined by $f \mapsto (h(f), g(f))$ is injective,*

   and (2) holds.

Proof. We only prove the bottom-up case, the top-down case in analogous. Additional care has to be taken, as $\Pr_d$ contains a probability distribution for every tree node, whereas a top-down PTA only contains one for the inner nodes.

Let $A = (Q, \delta, F)$ be a bottom-up PTA. Define the set M by $M = \prod_{n\geq 0} M_n$ where $M_n = Q^{\Sigma_n \times Q^n}$, i.e., M contains functions mapping tuples $(f, q_1, \dots, q_{ar(f)})$ to states. The probability distribution d is given by

$$d(m) = \prod_{f\in\Sigma} \prod_{q_1,\dots,q_{ar(f)}\in Q} \delta(f,q_1,\dots,q_{ar(f)})(m(f,q_1,\dots,q_{ar(f)})).$$

Let $\Gamma = \Sigma \times M$ with $ar(f,m) = ar(f)$, we set $h(f,m) = f$, and $g(f,m) = (m, ar(f))$. Let the tree language L contain all trees $t \in \mathrm{T}_\Gamma$ for which there is a run $\rho \colon \mathrm{dom}(t) \to Q$ with $\rho(x) = \pi_2(t(x))(\pi_1(t(x)), \rho(x\,1), \dots, \rho(x\ ar(t(x))))$ for all $x \in \mathrm{dom}(t)$, and $\rho(\varepsilon) \in F$. Then, L is regular. One can check that (2) holds using these definitions.

Conversely, consider the rank alphabet $\Gamma' = \Sigma \times M$ and the relabelling $\kappa\colon \Gamma \to \Gamma'$ with $\kappa(a) = (h(a), \pi_1(g(a)))$. Let $A' = (Q', \delta', F')$ be a deterministic bottom-up tree automaton with $\mathrm{L}(A') = \kappa(L)$. We define $A = (Q', \delta, F')$ with $\delta(f, q_1, \ldots, q_n) = \sum\{d(m) \mid \delta'((f,m), q_1, \ldots, q_n) = q\}$. Again, we obtain (2). ◀

Using Theorem 22 we immediately conclude the following corollary.

▶ **Corollary 23.** *Let A be a top-down probabilistic tree automaton. There is a bottom-up probabilistic tree automaton B with $\|B\| = \|A\|$.*

4.3 Equivalence to Tree Automata

▶ **Theorem 24.** *Let $S\colon \mathrm{T}_\Sigma \to [0,1]$. The following statements are equivalent:*

1. *$S = \|A\|$ for a probabilistic bottom-up tree automaton A,*
2. *$S = [\![\varphi]\!]$ for a probabilistic MSO sentence φ.*

Proof. Given a bottom-up PTA A, we apply Theorem 22 to obtain Γ, M, L, d, h and g as in the statement of the theorem such that (2) holds. We may assume $M = \{1, \ldots, m\}$ and $\Gamma = \{1, \ldots, \ell\}$. Choose probability values $p_1, \ldots, p_m$ such that $d(k) = p_k \prod_{i=1}^{k-1}(1-p_i)$. Using the classical Büchi theorem one constructs a Boolean MSO sentence ψ such that $\mathrm{L}(\psi) = L$. Let $X_1, \ldots, X_m, Y_1, \ldots, Y_\ell$ be new set variable symbols. Replace every occurrence of $\mathrm{label}_f(x)$ in ψ by $x \in Y_f$ resulting in a new formula ψ'. Let $\mathit{part}(Y_1, \ldots, Y_k)$ be a MSO formula expressing that $Y_1, \ldots, Y_k$ is a partition of the domain. We define φ as

$$\varphi = \mathbb{E}_{p_1} X_1 \cdots \mathbb{E}_{p_m} X_m \exists Y_1 \cdots \exists Y_\ell . \mathit{part}(Y_1, \ldots, Y_\ell) \wedge \psi'$$
$$\wedge \forall x . \bigwedge_{f \in \Gamma} x \in Y_f \implies \Big(\mathrm{label}_{h(f)}(x) \wedge x \in X_{g(f)} \wedge \bigwedge_{k=1}^{g(f)-1} x \notin X_k\Big).$$

Consider a tree $t \in \mathrm{T}_\Sigma$. The Y_i's encode a tree $\bar{t} \in \mathrm{T}_\Gamma$ with $\bar{t} \models \psi$, i.e., $\bar{t} \in L$. The second line of the formula states that $t = h(\bar{t})$ and chooses the X_i's such that the minimal k with $x \in X_k$ is $g(\bar{t}(x))$ for every $x \in \mathrm{dom}(t)$. Hence, fixing $\bar{t}$, the probabilities at every position x sum up to $p_{g(\bar{t}(x))} \prod_{i=1}^{g(\bar{t}(x))-1}(1-p_i) = d(g(\bar{t}(x)))$. Thus, we obtain $\Pr_d(\{g(\bar{t})\})$ for the whole tree. Considering arbitrary $\bar{t}$, we conclude that $[\![\varphi]\!]$ equals the right side of (2).

Conversely, let φ be a probabilistic MSO sentence. By Lemma 19, there is an equivalent sentence of the form $\mathbb{E}_{p_1} X_1 \cdots \mathbb{E}_{p_m} X_m . \psi$ where ψ is Boolean. Let $M = \{0,1\}^m$, $\Gamma = \Sigma \times M$, and $h\colon \Gamma \to \Sigma$ and $g\colon \Gamma \to M$ be the natural projections. Define $d \in \Delta(M)$ by $d(m) = (\prod_{i,\, m(i)=1} p_i)(\prod_{i,\, m(i)=0}(1-p_i))$. Again, by the classical Büchi theorem $L = \mathrm{L}(\psi) \subseteq \mathrm{T}_\Gamma$, where the additional components of $\Sigma \times \{0,1\}^m = \Gamma$ encode the values of the X_i's, is regular. One shows that (2) holds in this situation. Thus, by Theorem 22, there is a bottom-up PTA recognizing $[\![\varphi]\!]$. ◀

5 Conclusion and Future Research

We have introduced a probabilistic variant of regular tree expressions and proved that these expressions are expressively equivalent to top-down probabilistic tree automata. Next, we gave an extension of MSO logic to the probabilistic setting. It turned out that top-down PTA are too weak to recognize the semantics of all probabilistic MSO sentences. Thus, we introduced bottom-up probabilistic tree automata. We could show that the class of these automata is expressively equivalent to probabilistic MSO. In order to prove this result we also obtained a Nivat-style theorem for bottom-up PTA.

Future research might look into an extension of these results to unranked trees. There already exists MSO logic on unranked trees for the unweighted [15] as well as for the weighted case [10]. For regular tree expressions there already exist forest expressions [2] and one could extend unweighted ranked regular tree expressions to the unranked case. None of these concepts directly fit into the probabilistic setting. A different, interesting structure is infinite ranked trees. Probabilistic tree automata for infinite trees have been given in [5]. The extension of probabilistic regular tree expressions to infinite trees looks promising. For probabilistic MSO logic there does not seem to be a proper automata model, but one could still get the equivalence to the tree series defined by the Nivat decomposition from Theorem 22. A different notion of probabilistic regular expressions on trees has been given in [14]. These expressions use pebbles and are tree-walking. It has been shown [3] that in the unweighted case pebble tree-walking automata are strictly less expressive than regular tree languages. It remains to be seen if this inclusion also holds in the probabilistic case.

References

1 Parvaneh Babari and Manfred Droste. A Nivat theorem for weighted picture automata and weighted MSO logic. In *Proc. of LATA 2015*, volume 8977 of *LNCS*, pages 703–715. Springer, 2015.

2 Mikołaj Bojańczyk. Forest expressions. In *Computer Science Logic*, volume 4646 of *LNCS*, pages 146–160. Springer, 2007.

3 Mikołaj Bojańczyk, Mathias Samuelides, Thomas Schwentick, and Luc Segoufin. Expressive power of pebble automata. In *ICALP 2006*, volume 4051 of *LNCS*, pages 157–168. Springer, 2006.

4 Benedikt Bollig, Paul Gastin, Benjamin Monmege, and Marc Zeitoun. A probabilistic Kleene theorem. In *Proc. of ATVA 2012*, volume 7561 of *LNCS*, pages 400–415. Springer, 2012.

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

6 H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available on: `http://www.grappa.univ-lille3.fr/tata`, 2008. release November 18, 2008.

7 Manfred Droste, Christian Pech, and Heiko Vogler. A Kleene theorem for weighted tree automata. *Theory of Computing Systems*, 38(1):1–38, 2005.

8 Manfred Droste and Vitaly Perevoshchikov. A Nivat theorem for weighted timed automata and weighted relative distance logic. In *Proc. of ICALP 2014*, volume 8573 of *LNCS*, pages 171–182. Springer, 2014.

9 Manfred Droste and Heiko Vogler. Weighted tree automata and weighted logics. *Theoretical Computer Science*, 366(3):228 – 247, 2006. Automata and Formal Languages.

10 Manfred Droste and Heiko Vogler. Weighted logics for unranked tree automata. *Theory Comput. Syst.*, 48(1):23–47, 2011.

11 Clarence A. Ellis. Probabilistic tree automata. *Information and Control*, 19(5):401 – 416, 1971.

12 Olympia Louscou-Bozapalidou. Stochastic tree functions. *International Journal of Computer Mathematics*, 52(3-4):149–160, 1994.

13 M. Magidor and G. Moran. Probabilistic tree automata and context free languages. *Israel Journal of Mathematics*, 8(4):340–348, 1970.

14 Benjamin Monmege. *Specification and verification of quantitative properties: expressions, logics, and automata.* PhD thesis, ENS Cachan, 2013.

15 Frank Neven and Thomas Schwentick. Query automata over finite trees. *Theor. Comput. Sci.*, 275(1-2):633–674, 2002.

16 Maurice Nivat. Transductions des langages de Chomsky. *Annales de l'institut Fourier*, 18(1):339–455, 1968.

17 Thomas Weidner. Probabilistic automata and probabilistic logic. In *Proc. of MFCS 2012*, volume 7464 of *LNCS*, pages 813–824. Springer, 2012.

18 Thomas Weidner. Probabilistic ω-regular expressions. In *Proc. of LATA 2014*, volume 8370 of *LNCS*, pages 588–600. Springer, 2014.