Series, Weighted Automata, Probabilistic Automata and Probability Distributions for Unranked Trees.

Édouard Gilbert, Rémi Gilleron, Marc Tommasi

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Series, Weighted Automata, Probabilistic Automata and Probability Distributions for Unranked Trees

Édouard Gilbert — Rémi Gilleron — Marc Tommasi

N° 7200

November 2008

Thème SYM

ISSN 0249-6399 ISRN INRIA/RR--7200--FR+ENG

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE
INRIA
centre de recherche LILLE - NORD EUROPE

Series, Weighted Automata, Probabilistic Automata and Probability Distributions for Unranked Trees

Édouard Gilbert, Rémi Gilleron, Marc Tommasi *

Thème SYM — Systèmes symboliques
Équipe-Projet Mostrare

Rapport de recherche n° 7200 — November 2008 — 23 pages

Abstract: We study tree series and weighted tree automata over unranked trees. The message is that recognizable tree series for unranked trees can be defined and studied from recognizable tree series for binary representations of unranked trees. For this we prove results of [1] as follows. We extend hedge automata – a class of tree automata for unranked trees – to weighted hedge automata. We define weighted stepwise automata as weighted tree automata for binary representations of unranked trees. We show that recognizable tree series can be equivalently defined by weighted hedge automata or weighted stepwise automata. Then we consider real-valued tree series and weighted tree automata over the field of real numbers. We show that the result also holds for probabilistic automata – weighted automata with normalisation conditions for rules. We also define convergent tree series and show that convergence properties for recognizable tree series are preserved via binary encoding. From [21], we present decidability results on probabilistic tree automata and algorithms for computing sums of convergent series. Last we show that streaming algorithms for unranked trees can be seen as slight transformations of algorithms on the binary representations.

Key-words: Tree automata, weighted automata, XML

* This work was supported by the ANR Lampada project ANR-09-EMER-007

Centre de recherche INRIA Lille – Nord Europe
Parc Scientifique de la Haute Borne
40, avenue Halley, 59650 Villeneuve d'Ascq
Téléphone : +33 3 59 57 78 00 — Télécopie : +33 3 59 57 78 50

Résumé : We study tree series and weighted tree automata over unranked trees. The message is that recognizable tree series for unranked trees can be defined and studied from recognizable tree series for binary representations of unranked trees. For this we prove results of [1] as follows. We extend hedge automata – a class of tree automata for unranked trees – to weighted hedge automata. We define weighted stepwise automata as weighted tree automata for binary representations of unranked trees. We show that recognizable tree series can be equivalently defined by weighted hedge automata or weighted stepwise automata. We also show that the result holds for probabilistic automata – weighted automata over the field of real numbers with normalisation condition for rules. We also claim that convergence properties for recognizable tree series are preserved. Last we show that streaming algorithms for unranked trees can be seen as slight transformations of algorithms on the binary representations.

Mots-clés : Automates d'arbres, Automates pondérés, XML

1 Introduction

Tree structured data is commonly used in computer science. For instance, tree structures are the favorite data model for Internet data. They are also of great interest in natural language processing and bio-computing. Because of the rapid development of these application fields, we are faced with very large data sets of tree structured data. A key challenge is to design methods and algorithms to represent, query, transform, rank and mine into such very large data sets.

The approach we develop in this paper is to model large sets of tree structured data with probabilistic models. The main problem is the inference of such probabilistic models from data. This problem heavily relies on the choice of an adequate representation formalism for probability distributions over tree structured data. In the ranked case, most of the known approaches ([2] or [3]) consider the inference of probability distributions defined by deterministic probabilistic tree automata. On the one hand determinism eases the design of machine learning algorithms. On the other hand, determinism restricts the class of probability distributions: already in the string case, there are probability distributions defined by probabilistic automata which can not be defined by deterministic probabilistic automata [4]. Also, non-deterministic automata are useful for labeling trees because the question is to find the most likely labeling – i.e. the most likely run. Moreover, already in the string case, automata with non negative weights are less expressive than automata with real weights [4]. Non deterministic weighted tree automata correspond to recognizable tree series over the field of real numbers and, recently, inference of probability distributions defined by recognizable tree series over the field of real numbers has been considered [5, 6].

Motivated by XML applications, we consider the inference of probability distributions over sets of unranked trees (i.e. trees where a node can have an unbounded number of direct subtrees). Thus, the first question is to define recognizable tree series for unranked trees. For this, we define weighted automata for unranked trees.

Along this line, hedge automata [7] are a popular model for unranked trees ([8], [9, Chapter 8]). We extend both hedge automata and weighted tree automata defining weighted hedge automata. This leads to a rather intricate automata model and properties of recognizable tree series can not be easily shown. Thus, we consider binary representations of unranked trees. Several bijections which map unranked trees to ranked ones have been defined [9, Chapter 8]. Among them, an algebraic representation similar to the currying transformation of functions has been defined. This algebraic representation has the advantage of defining a bijection between a set of unranked trees over an alphabet Σ and a set of ranked trees over a ranked alphabet containing only one binary symbol and constants in Σ. We define weighted stepwise automata as weighted tree automata over those algebraic representations. We show that weighted hedge automata and weighted stepwise automata define the same sets of recognizable tree series.

Then, we consider tree series over the field of real numbers. As a consequence, weighted hedge automata and weighted stepwise automata define the same sets of recognizable stochastic tree series. We also show that the equivalence between weighted hedge automata and weighted stepwise automata also holds for probabilistic automata, i.e. when normalisation conditions over rules

occur. We define convergent tree series ordering trees by increasing size and we show that the definition of recognizable tree series for unranked trees allows to define convergence of tree series in a consistent way. We also recall algorithms to compute sums of recognizable tree series defined by probabilistic tree automata with results from [21]. Using the same paper, we show that it is decidable whether a probabilistic tree automaton defines a probability distribution.

Last, we study algorithmic properties of the algebraic representation. Algorithms for ranked trees can be extended to unrankeds tree by applying them on binary representations; we show that in many cases, they can also be slightly transformed to apply directly to unranked trees.

Related work In [10], following [11] for the non-weighted case, the authors provided bisimulation minimisation results for different classes of weighted unranked tree automata. Along the way, they proved the equivalence between weighted automata models (weighted hedge automata are called weighted unranked tree automata (wuta) in [10], weighted stepwise automata are called weighted stepwise unranked tree automata (wsuta) in [10]). While our message is to say that all results can be obtained via a binary representation of unranked trees, they do not put such an evidence in their paper. They consider weights from a semiring and do not consider the field of real numbers. Thus, they do not consider probabilistic automata and do not consider convergence properties of tree series.

2 Preliminaries

Let $\mathbf{N}$ be the set of positive integers. Let $\mathbf{N}^*$ be the set of finite strings over $\mathbf{N}$ and let $\epsilon \in \mathbf{N}^*$ be the empty string. We denote by $p.p'$ the concatenation of two strings $p, p' \in \mathbf{N}^*$. A subset P of $\mathbf{N}^*$ is prefix-closed if $p.p' \in P$ implies that $p \in P$. A position set P is a non-empty prefix-closed finite set such that for every $p.i \in P$, we have $p.j \in P$ whenever $1 \leq j \leq i$. Let $\mathcal{F}$ be a finite set of symbols, a (finite ordered rooted) tree t is a mapping from a position set $Pos(t)$ into $\mathcal{F}$. Each element in $Pos(t)$ is a position. It identifies a node of t associated with a symbol $f = t(p)$ in $\mathcal{F}$. The rank of p in t is the number of direct successors (or children) of p, that is the number of positions $p.i \in Pos(t)$ where $i \in \mathbf{N}$. A position p such that $p.1 \notin Pos(t)$ corresponds to a leaf of t, otherwise it corresponds to an internal node. The *size* $|t|$ of a tree t is the cardinality of its position set. The *height* $\mathsf{height}(t)$ of a tree t is the maximal length of strings in its position set.

In the following, we consider two sorts of trees: ranked and unranked trees. In the ranked case, the alphabet can be partitioned into $\mathcal{F} = \bigcup_{n=0}^{N} \mathcal{F}_n$ such that for every n and $f \in \mathcal{F}_n$, the rank of any node labeled by f is n. In that case, $\mathcal{F}$ is said to be a *ranked alphabet* and $f \in \mathcal{F}_n$ is a symbol of rank n. The set $T_{\mathcal{F}}$ is the set of *ranked trees on* $\mathcal{F}$. Also, $T_{\mathcal{F}}$ can be defined as the smallest set such that

$$\forall n \leq N \ \forall f \in \mathcal{F}_n \ \forall (t_1, \ldots, t_n) \in T_{\mathcal{F}} \quad f(t_1, \ldots, t_n) \in T_{\mathcal{F}}.$$

In the unranked case, the rank of nodes is not determined by the symbol. While the rank is finite for every node, it is unbounded. To distinguish between ranked

and unranked trees, we denote by Σ the set of symbols and by T^u_Σ the set of unranked trees over Σ. Similarly, T^u_Σ can be defined as the smallest such that

$$\forall n \in \mathbf{N}\ \forall f \in \Sigma\ \forall (t_1, \ldots, t_n) \in T^u_\Sigma \quad f(t_1, \ldots, t_n) \in T^u_\Sigma.$$

Let $\mathbf{K}$ be a commutative semiring. A $\mathbf{K}$-tree series over a set of trees is a function r from the given set of trees to $\mathbf{K}$. We will consider tree series over $T_\mathcal{F}$ or T^u_Σ. When $\mathbf{K}$ is the Boolean semiring, the tree series actually define tree languages. In the following, we will mainly consider the case of tree series over $\mathbf{K} = \mathbf{R}$ or $\mathbf{K} = \mathbf{R}_+$. We first consider *recognizable tree series* over ranked trees defined by means of weighted tree automata [12].

Definition 1. *A* weighted tree automaton *is a tuple $\mathcal{A} = (\mathcal{F}, Q, F, \delta)$ such that $\mathcal{F}$ is a finite ranked alphabet, Q a finite set of states, $F : Q \to \mathbf{K}$ a final function and $\delta : (\bigcup_{n\in\mathbf{N}} \mathcal{F}_n \times Q^n) \times Q \to \mathbf{K}$ a transition function.*

Definition 2. *Let $t \in T_\mathcal{F}$ be a ranked tree and $\mathcal{A} = (\mathcal{F}, Q, F, \delta)$ an automaton on $\mathcal{F}$. A* run of $\mathcal{A}$ on t *is a function from $Pos(t)$ to Q, i.e. every position of t is associated with a state from Q. The weight $\mathsf{weight}(\mathcal{A}, t, r, p)$ of a run r on a subtree of t at position p is computed as follows: if the rank of p in t is 0, then $\mathsf{weight}(\mathcal{A}, t, r, p) = \delta(t(p), r(p))$; otherwise, let $n > 0$ be the rank of p, $\mathsf{weight}(\mathcal{A}, t, r, p)$ is inductively defined by*

$$\mathsf{weight}(\mathcal{A}, t, r, p) = \delta\Big(t(p), r(p.1), \ldots, r(p.n), r(p)\Big) \cdot \prod_{i=1}^{n} \mathsf{weight}(\mathcal{A}, t, r, p.i).$$

The weight of a run r on t is $\mathsf{weight}(\mathcal{A}, t, r) = F(r(\epsilon)) \times \mathsf{weight}(\mathcal{A}, t, r, \epsilon)$, i.e. the final function is applied at the root position. The weight of a tree t is the sum of weights of all runs of $\mathcal{A}$ on t, i.e. $\mathsf{weight}(\mathcal{A}, t) = \sum_r \mathsf{weight}(\mathcal{A}, t, r)$. A weighted automaton $\mathcal{A}$ defines a recognizable tree series $r_\mathcal{A}$, which maps every t in $T_\mathcal{F}$ on $\mathsf{weight}(\mathcal{A}, t)$.

We should note that recognizable tree series are equivalent to *rational tree series* defined in [13] (where they are called recognizable formal power tree series) defined by linear representations.

3 Weighted Hedge Automata

When considering tree languages, there exist several models of unranked tree automata which were proved to have the same expressive power (see [9, Chapter 8]). Hedge automata are popular in the XML community. Roughly speaking, they combine tree-automata rules for the vertical recursion, and string automata over an alphabet of states for the horizontal recursion. We now extend hedge automata to the weighted case (a similar definition was proposed in [10]). For the horizontal recursion, we use weighted string automata defining rational (string) series over an alphabet of states. We also add weights to tree-automata like rules.

Definition 3. *A* weighted hedge automaton *is a tuple $\mathcal{H} = (\Sigma, Q_h, F_h, \delta_h)$ where Σ is a finite alphabet, Q_h a finite set of states, $F_h : Q_h \to \mathbf{K}$ a final function. $\delta_h : \Sigma \times Q_h^* \times Q_h \to \mathbf{K}$ is a transition function which can be described*

as follows. For every $f \in \Sigma$ and target state $q \in Q_h$ there is a $\mathbf{K}$*-rational string series $\pi_{f,q}$ over Q_h^* and a weight $w_{f,q} \in \mathbf{K}$ such that, for $u \in Q_h^*$,*

$$\delta_h(f, u, q) = w_{f,q} \cdot \pi_{f,q}(u).$$

A tuple $(f, q, \pi_{f,q}, w_{f,q})$ is called a rule *and is written $f(\pi_{f,q}) \to q\ [w_{f,q}]$. Moreover, the (string) series $\pi_{f,q}$ is assumed to be defined by a weighted string automaton $\mathcal{S}_{f,q} = (Q_h, Q_{f,q}, \iota_{f,q}, \varphi_{f,q}, \tau_{f,q})$, where Q_h is the alphabet, $Q_{f,q}$ is a set of states, and $\iota_{f,q}$, $\varphi_{f,q}$ and $\tau_{f,q}$ are, respectively, the initial, transition and final functions.*

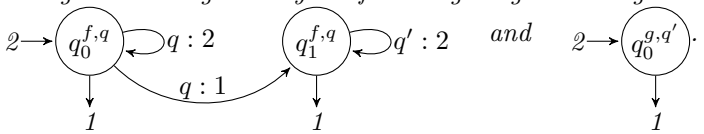
In general, it is assumed that the state sets $Q_{f,q}$ are pairwise disjoint. Moreover, we assume that there is only one rule for every $f \in \Sigma$ and every $q \in Q_h$. This is done without loss of generality because of the closure of rational (string) series under sum. It is important to note that the weighted string automata are non deterministic and that they can not be replaced by a deterministic one. Indeed, the class of (string) series defined by deterministic weighted automata is stricly included in the class of (string) series defined by non deterministic ones.

In order to define how weighted hedge automata compute weights of input unranked trees, we must define runs. But, it should be noted that, in the unranked case, a run of an automaton $\mathcal{H}$ on a tree is not completely defined by a labeling of the unranked input tree by states in the state set Q_h of $\mathcal{H}$. This is because a run must also memorize configurations of the weighted string automata over strings in Q_h^*. Thus, we consider inner positions in the trees that will memorize states associated with runs of the string automata $\mathcal{S}_{f,q}$.

The set $IPos(t)$ of *inner positions* in a tree t is a subset of $Pos(t) \times (\mathbf{N} \cup \{0\})$ defined by $IPos(t) = \{(p, k) \mid p \in Pos(t)\,,\ 0 \leq k \leq rank(p)\}$. An inner position (p, k) is denoted by $p[k]$. Inner positions of the tree $f(f, f(f, f), g)$ are given in Figure 1.

Now, a *run of $\mathcal{H}$ on t* defines a computation of a weighted hedge automaton on a tree t. A run is a function $\sigma : IPos(t) \to \bigcup Q_{f,q}$ which associates to each inner position the state of the automaton $\mathcal{S}_{f,q}$. A run also defines a labeling ρ_σ of the unranked input tree t by states in the state set Q_h of $\mathcal{H}$. Indeed, for every node p of t of rank n, there is a unique state q in the state set Q_h of $\mathcal{H}$ such that, for every $k \in [0, n]$, we have $\sigma(p[k]) \in Q^{f,q}$ where $f = t(p)$. We denote this state q by $\rho_\sigma(p)$. A run of an automaton on the tree $f(f, f(f, f), g)$ is shown in Figure 1. It should be noted that in [10], a run of a weighted hedge automaton (called weighted unranked tree automaton) is defined to be the mapping ρ_σ.

Example 1. *Figure 1 represents a run of the weighted hedge automaton with two rules $f(L_1) \to q$ [2] and $g(L_2) \to q'$ [1] where L_1 and L_2, respectively, are the string series recognized by the following weighted string automata:*



Definition 4. *Let $\mathcal{H}$ be a weighted hedge automaton defined as in Definition 3. Let t be an unranked tree, let p be a position of t, let $f = t(p)$, and let n be the rank of p. Let σ be a run of $\mathcal{H}$ on t, let $q = \rho_\sigma(p)$. The weight* $\mathsf{weight}(\mathcal{H}, t, \sigma, p)$ *of σ at position p and the weight* $\mathsf{weight}(\mathcal{H}, t, \sigma, p, m)$ *of σ at internal position $p[m]$ for $0 \leq m \leq n$ are inductively defined by*

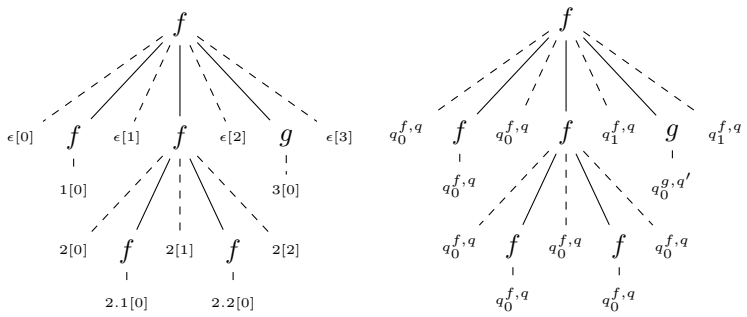


Figure 1: left: inner positions of $f(f, f(f, f), g)$; right: a run of the automaton given in Example 1 over $f(f, f(f, f), g)$.

$$\mathit{weight}(\mathcal{H}, t, \sigma, p, 0) = w_{f,q} \times \iota_{f,q}(\sigma(p[0])) \tag{1}$$

$$\mathit{weight}(\mathcal{H}, t, \sigma, p, m) = \mathit{weight}(\mathcal{H}, t, \sigma, p, m-1) \times \varphi_{f,q}(\sigma(p[m-1]), \rho_\sigma(p.m), \sigma(p[m])) \times \mathit{weight}(\mathcal{H}, t, \sigma, p.m),\ 0 < m \leq n \tag{2}$$

$$\mathit{weight}(\mathcal{H}, t, \sigma, p) = \mathit{weight}(\mathcal{H}, t, \sigma, p, n) \times \tau_{f,q}(\sigma(p[n])). \tag{3}$$

The weight of a run σ on t is $\mathit{weight}(\mathcal{H}, t, \sigma) = F_h(\rho_\sigma(\epsilon)) \times \mathit{weight}(\mathcal{H}, t, \sigma, \epsilon)$. *The weight of a tree t is the sum of the weights of all runs of $\mathcal{H}$ on t, i.e.* $\mathit{weight}(\mathcal{H}, t) = \sum_\sigma \mathit{weight}(\mathcal{H}, t, \sigma)$. *A weighted hedge automaton $\mathcal{H}$ defines a recognizable tree series $r_\mathcal{H}$ which maps every t in T^u_Σ to* $\mathit{weight}(\mathcal{H}, t)$.

4 Forget Unranked Automata Models

Weighted hedge automata is an ad hoc automata model for unranked trees. Definitions are intricate because of the string automata over an alphabet of states. Thus, properties of recognizable tree series for unranked trees can not be easily shown. It is simpler to consider tree automata over binary representations of unranked trees. In this section, we define the binary representation, then we define weighted stepwise automata as weighted tree automata over those binary representations representations. Last, we show that weighted hedge automata and weighted stepwise automata define the same sets of recognizable tree series.

4.1 Weighted Stepwise Automata

First, let us define the binary representation we use. The operator @ adds a new child at the end of the list of children of an unranked tree:

$$f \mathbin{@} t = f(t)\ \forall f \in \Sigma\ \forall t \in T^u_\Sigma$$
$$f(t_1, \ldots, t_n) \mathbin{@} t = f(t_1, \ldots, t_n, t)\ \forall f \in \Sigma\ \forall t_1, \ldots, t_n, t \in T^u_\Sigma$$

Syntactically, given an alphabet Σ, we consider a ranked alphabet $\mathcal{F}^@ = \mathcal{F}_0 \cup \mathcal{F}_2$ where $\mathcal{F}_0 = \Sigma$ and $\mathcal{F}_2 = \{@\}$. Then, every unranked tree can be represented by a binary tree over $\mathcal{F}^@$. This can be formally defined with the mapping $t \mapsto \overline{t}$

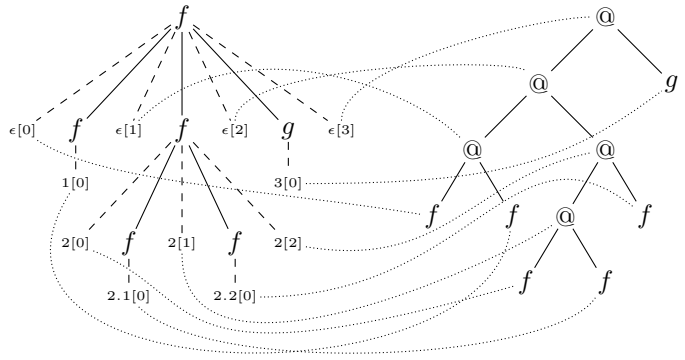


Figure 2: Bijection between inner positions of $t = f(f, f(f, f), g)$ and positions of $\overline{t}$.

from T^u_Σ to $T_{\mathcal{F}^@}$ such that $\overline{f} = f$ and $\overline{f(t_1, \ldots, t_n)} = @(\overline{f(t_1, \ldots, t_{n-1})}, \overline{t_n})$. It should be noted that the mapping $t \to \overline{t}$ is a bijection between T^u_Σ to $T_{\mathcal{F}^@}$.

Example 2. *The binary representation of the unranked tree $f(f, f(f, f), g)$ from Example 1 is $@(@(@(f, f), @(@(f, f), f)), g)$.*

There exists a bijection between the set $IPos(t)$ of inner positions of an unranked tree t and the set $Pos(\overline{t})$ of positions of its binary representation $\overline{t}$. An example is given in Figure 2. This bijection is denoted by ψ_t and defined by:

- for $\epsilon[k] \in IPos(t)$, $\psi_t(\epsilon[k]) = 1^{n-k}$ where n is the rank of ϵ in t and 1^{n-k} denotes the concatenation of $n-k$ 1. It should be noted that $\psi_t(\epsilon[n]) = \epsilon$.
- For $p.i[k] \in IPos(t)$ where $p \in \mathbf{N}^*$ and $i \in \mathbf{N}$, $\psi_t(p.i[k]) = \psi_t(p[i]).2.\psi_{t'}(\epsilon[k])$ where t' is the subtree of t rooted at position $p.i$.

We define *weighted stepwise automata* as weighted tree automata (over ranked trees) that operate on binary representations of unranked trees.

4.2 From Weighted Hedge Automata to Weighted Stepwise Automata

Proposition 1. *For every weighted hedge automaton $\mathcal{H}$ over Σ, there is a weighted stepwise automaton $\mathcal{A}$ over $\mathcal{F}^@$ such that, for every t in T^u_Σ, $r_\mathcal{H}(t) = r_\mathcal{A}(\overline{t})$.*

Proof. It is easy to verify that any weighted hedge or stepwise automaton can be augmented with new states and rules of weight 0 without changing the series recognized. Therefore, we will use only one set of states.

Let $\mathcal{H} = (\Sigma, Q_h, F_h, \delta_h)$ be a weighted hedge automaton, with weighted string automata $\mathcal{S}_{f,q} = (Q_h, Q_{f,q}, \iota_{f,q}, \varphi_{f,q}, \tau_{f,q})$. We modify the definition of $\mathcal{H}$ by defining the weighted string automata on the state set $Q = \bigcup_{f \in \Sigma, q \in Q} Q_{f,q}$ by: $\iota_{f,q}(q) = \iota_{f,q}(q)$ if $q \in Q^{f,q}$, and 0 otherwise; $\varphi_{f,q}(q_1, q', q_2) = \varphi_{f,q}(q_1, q', q_2)$ if $q_1, q_2 \in Q^{f,q}$, and 0 otherwise ;$\tau_{f,q}(q) = \tau_{f,q}(q)$ if $q \in Q^{f,q}$, and 0 otherwise. Clearly, this does not affected the series recognized by $\mathcal{H}$.

Let us note that two inner positions $p[m-1]$ and $p.m[n]$ where n is the rank of position $p.m$ in t are mapped by ψ_t, respectively, to the left child and to the right child of $\psi_t(p[m])$ in $\bar{t}$. More precisely a triplet of position $(p'.1, p'.2, p')$ is in one to one correspondence with a triplet of the form $(p[m-1], p.m[n], p[m])$ where n is the arity of the node $p.m$ and $\psi_t(p[m]) = p'$. Lastly, $(p'.1, p'.2, p')$ is subject of a rule in the stepwise case, and its antecedent by ψ_t is also subject of a rule in the hedge case. The proof directly follows these observations.

We define a weighted stepwise automaton $\mathcal{A} = (\mathcal{F}^{@}, Q, F, \delta)$ where $Q = \bigcup_{f\in\Sigma, q\in Q} Q_{f,q}$ and for all $q_i, q_j, q_k \in Q$:

- $F(q_i) = \tau_{f,q}(q_i) \times F_h(q)$
- $\delta(f, q_i) = w_{f,q} \times \iota_{f,q}(q_i)$;
- $\delta(@, q_i, q_k, q_j) = \tau_{g,q}(q_k) \times \varphi_{f,q'}(q_i, q, q_j)$.

We must prove that, for every t in T^u_Σ, $r_\mathcal{H}(t) = r_\mathcal{A}(\bar{t})$. For every unranked tree t, the mapping ψ_t defines a one to one correspondence between runs of $\mathcal{H}$ on t and runs of $\mathcal{A}$ on $\bar{t}$. Let σ be a run of $\mathcal{H}$ on t, we denote by $\psi_t(\sigma)$ the corresponding run of $\mathcal{A}$ on $\bar{t}$. In order to prove that $r_\mathcal{H}$ and $r_\mathcal{A}$ are equal, it remains to show that, for every t and for every run σ of $\mathcal{H}$ on t, $\mathsf{weight}(\mathcal{H}, t, \sigma) = \mathsf{weight}(\mathcal{A}, \bar{t}, \psi_t(\sigma))$. Thus, let us consider an unranked tree t and a run σ of $\mathcal{H}$ over t. We prove that, for every position p

$$\mathsf{weight}(\mathcal{H}, t, \sigma, p, n) = \mathsf{weight}(\mathcal{A}, \bar{t}, \psi_t(\sigma), \psi_t(p[n])) \tag{4}$$

where n is the rank of p in t and $t(p) = f$. We proceed by structural induction.
Base. Let p be a a position of rank 0. There is only one inner position $p[0]$ and $\psi_t(p[0])$ is a leaf position in $\bar{t}$. Then

$$\begin{aligned}\mathsf{weight}(\mathcal{H}, t, \sigma, p, 0) &= w_{f,q} \times \iota_{f,q}(\sigma(p[0])) \text{ (def. of } \mathsf{weight}) \\ &= \delta(f, \sigma(p[0])) \text{ (def. of } \mathcal{A}) \\ &= \mathsf{weight}(\mathcal{A}, \bar{t}, \psi_t(\sigma), \psi_t(p[0])) \text{ (def. of } \mathsf{weight})\end{aligned} \tag{5}$$

Structural Induction. Let us consider a position p of rank $n > 0$ and let us assume that (4) holds for every position $p.i$ in t. We prove by induction that, for every m such that $0 \leq m \leq n$, we have:

$$\mathsf{weight}(\mathcal{H}, t, \sigma, p, m) = \mathsf{weight}(\mathcal{A}, \bar{t}, \psi_t(\sigma), \psi_t(p[m])). \tag{6}$$

Let $m = 0$, we use (5) thus $\mathsf{weight}(\mathcal{H}, t, \sigma, p, 0) = \mathsf{weight}(\mathcal{A}, \bar{t}, \psi_t(\sigma), \psi_t(p[0]))$.

Let $0 < m \leq n$, let us suppose that (6) holds for $m-1$. We denote by q_m the state $\rho_\sigma(p.m)$.

$$\begin{aligned}\mathsf{weight}(\mathcal{H}, t, \sigma, p, m) = \mathsf{weight}&(\mathcal{H}, t, \sigma, p, m-1) \\ &\times \varphi_{f,q}(\sigma(p[m-1]), q_m, \sigma(p[m])) \\ &\times \mathsf{weight}(\mathcal{H}, t, \sigma, p.m) \text{ (def. of } \mathsf{weight})\end{aligned} \tag{7}$$

Let us consider the three terms in this product. By induction hypothesis:

$$\mathsf{weight}(\mathcal{H}, t, \sigma, p, m-1) = \mathsf{weight}(\mathcal{A}, \bar{t}, \psi_t(\sigma), \psi_t(p[m-1])) \tag{8}$$

Let n' be the rank of position $p.m$ in t, then as $q_m = \rho_\sigma(p.m)$, we have $q_m = \sigma(p.m[n'])$ and by definition of δ in the construction of $\mathcal{A}$ from $\mathcal{H}$:

$$\delta(@, \sigma(p[m{-}1]), \sigma(p.m[n']), \sigma(p[m])) = \varphi_{f,q}(\sigma(p[m{-}1]), q_m, \sigma(p[m])) \times \tau_{f,q_m}(q_m). \tag{9}$$

Thus, thanks to the definition of weight in (3) and to the structural induction hypothesis, we get:

$$\begin{aligned} \mathsf{weight}(\mathcal{H}, t, \sigma, p.m) &= \mathsf{weight}(\mathcal{H}, t, \sigma, p.m, n') \cdot \tau_{f,q_m}(q_m) \\ &= \mathsf{weight}(\mathcal{A}, \bar{t}, \psi_t(\sigma), \psi_t(p.m[n'])) \cdot \tau_{f,q_m}(q_m). \end{aligned} \tag{10}$$

Using equations (7), (8), (9), (10), we obtain that

$$\begin{aligned} \mathsf{weight}(\mathcal{H}, t, \sigma, p, m) = \mathsf{weight}(\mathcal{A}, \bar{t}, \psi_t(\sigma), \psi_t(p[m-1])) \\ \times \delta(@, \sigma(p[m-1]), \sigma(p.m[n']), \sigma(p[m])) \\ \times \mathsf{weight}(\mathcal{A}, \bar{t}, \psi_t(\sigma), \psi_t(p.m[n'])). \end{aligned}$$

Now, it remains to recall that $\psi_t(p[m-1]) = \psi_t(p[m]).1$ and $\psi_t(p.m[n']) = \psi_t(p[m]).2$, and then using the definition of weight for $\mathcal{A}$, we obtain:

$$\mathsf{weight}(\mathcal{H}, t, \sigma, p, m) = \mathsf{weight}(\mathcal{A}, \bar{t}, \psi_t(\sigma), \psi_t(p[m]))$$

Thus we have proved that (4) holds for every position p in t. Now let $q = \rho_\sigma(\epsilon)$ and let n be the rank of ϵ in t. Now using the definition of weights in weighted hedge automata, we have

$$\begin{aligned} \mathsf{weight}(\mathcal{H}, t, \sigma) &= F_h(q) \times \mathsf{weight}(\mathcal{H}, t, \sigma, \epsilon) \\ &= F_h(q) \times \mathsf{weight}(\mathcal{H}, t, \sigma, \epsilon, n) \times \tau_{f,q}(\sigma(\epsilon[n])) \end{aligned} \tag{11}$$

Also, by definition of weights in weighted tree automata

$$\begin{aligned} \mathsf{weight}(\mathcal{A}, \bar{t}, \psi_t(\sigma)) &= F(\psi_t(\sigma)(\epsilon)) \times \mathsf{weight}(\mathcal{A}, \bar{t}, \psi_t(\sigma), \epsilon) \\ &= \tau_{f,q}(\psi_t(\sigma)(\epsilon)) \times F_h(q) \times \mathsf{weight}(\mathcal{A}, \bar{t}, \psi_t(\sigma), \epsilon). \end{aligned} \tag{12}$$

Recall that $\psi_t(\epsilon[n]) = \epsilon$ and $\sigma(\epsilon[n]) = \psi_t(\sigma)(\epsilon)$. Thus using (4), we obtain $\mathsf{weight}(\mathcal{H}, t, \sigma) = \mathsf{weight}(\mathcal{A}, \bar{t}, \psi_t(\sigma))$ which concludes the proof. □ □

4.3 From Stepwise Weighted Automata to Weighted Hedge Automata

Proposition 2. *For every weighted stepwise automaton $\mathcal{A}$ over an alphabet $\{@\} \cup \Sigma$, there is a weighted hedge automaton $\mathcal{H}$ over Σ such that, for every $\bar{t}$ in $T_{\mathcal{F}^@}$, $r_\mathcal{A}(\bar{t}) = r_\mathcal{H}(t)$ where t is the unranked tree which binary representation is $\bar{t}$.*

Proof. Let $\mathcal{A} = (\mathcal{F}^{@}, Q, F, \delta)$ be a weighted stepwise automaton over $F^{@} = \{@\} \cup \Sigma$. Let $\mathcal{H} = (\Sigma, Q_h, F_h, \delta_h)$ be the weighted hedge automaton defined by: $Q_h = Q$, $F_h = F$, and δ_h contains rules

$$f(\pi_{f,q}) \to q \ [1]$$

where each series $\pi_{f,q}$ is defined by a weighted string automaton $\mathcal{S}_{f,q} = (Q, Q, \iota_f, \varphi, \tau_q)$ over alphabet Q with state set Q, and functions defined by: ι_f by $\iota_f(q) = \delta(f, q)$, τ_q by $\tau_q(q) = 1$ and $\tau_q(q') = 0$ whenever $q \neq q'$; finally, $\varphi(q_1, q, q_2) = \delta(@, q_1, q, q_2)$.

Note that the states and the alphabet of these automata are similar, that their initial function will only depend on f, the final function, only on q and the transition function is the same for every automaton. The proof is similar to the proof in Subsection 4.2. It is simpler because final weights are equal to 1. □

□

4.4 Probabilistic automata

As said in the introduction, probabilistic automata are used to define probability distributions over sets of trees. We define probabilistic automata over sets of unranked trees and we show that probabilistic hedge automata and probabilistic stepwise automata define the same sets of tree series.

Definition 5. *Let $\mathcal{A}$ be a weighted tree automaton and q a state of $\mathcal{A}$. The state q is* accessible *if and only if there is a symbol $a \in \Sigma$ such that $\delta(a, q) \neq 0$ or if there are two accessible states q_1, q_2 such that $\delta(@, q_1, q_2, q) \neq 0$. Similarly, q is* productive *if and only if $F(q) \neq 0$ or there are states q_1, q_0 such that q_0 is productive and $\delta(@, q, q_1, q_0) \neq 0$ or $\delta(@, q_1, q, q_0) \neq 0$.*

The automaton $\mathcal{A}$ is trimmed *if every state of $\mathcal{A}$ is both accessible and productive.*

A similar definition can be given for hedge automata. Checking whether an automaton is trimmed can be done with the same algorithms than in the non weighted case (see [9]).

Definition 6. *A* probabilistic stepwise automaton $\mathcal{A}$ *is a weighted stepwise automaton $\mathcal{A} = (\mathcal{F}^{@}, Q, F, \delta)$ such that:*

- $\forall f \in \Sigma$, $\forall q \in Q$, $\delta(f, q) \in \mathbf{R}_+$;
- $\forall q_1, q_2, q \in Q$, $\delta(@, q_1, q_2, q) \in \mathbf{R}_+$;
- $\forall q \in Q$, $\sum_{q_1, q_2 \in Q} \delta(@, q_1, q_2, q) + \sum_{f \in \Sigma} \delta(f, q) = 1$;
- $\sum_{q \in Q} F(q) = 1$.

Definition 7. *A* probabilistic hedge automaton $\mathcal{H}$ *is a weighted hedge automaton $\mathcal{H} = (\Sigma, Q_h, F_h, \delta_h)$ as in Definition 3 such that:*

- $\forall f \in \Sigma$, $\forall q \in Q_h$, $w_{f,q} \in \mathbf{R}_+$;
- $\forall q \in Q_h$, $\sum_{f \in \Sigma} w_{f,q} = 1$;
- $\sum_{q \in Q_h} F_h(q) = 1$;

- $\forall q \in Q$, $\forall f \in \Sigma$, $S_{f,q}$ *is a probabilistic string automaton.*

We prove that probabilistic hedge automata and probabilistic stepwise automata are equally expressive. To do that, we first observe a fact that follows the construction given in the proof of Proposition 2.

Proposition 3. *For every trimmed probabilistic hedge automaton $\mathcal{H}$ over Σ, there is a probabilistic trimmed stepwise automaton $\mathcal{A}$ over $\mathcal{F}^{\mathbb{Q}}$ such that for every $t \in T^u_\Sigma$, $r_\mathcal{H}(t) = r_\mathcal{A}(\bar{t})$. Vice-versa, for every trimmed probabilistic stepwise automaton $\mathcal{A}$ over $\mathcal{F}^{\mathbb{Q}}$ there is a trimmed probabilistic hedge automaton $\mathcal{H}$ over Σ satisfying the same relation.*

Proof. It is easy to prove that, given a trimmed probabilistic hedge automaton $\mathcal{H}$, the equivelent trimmed stepwise weighted automaton $\mathcal{A}$ defined in the proof of Proposition 1 is probabilistic.

The converse transformation is tedious because we must modify the construction given in the proof of Proposition 2 because of the normalization conditions required by probabilistic automata. It should be noted that we use existence of an inverse for $\times$.

Let $\mathcal{A} = (\mathcal{F}^{\mathbb{Q}}, Q, F, \delta)$ be a trimmed probabilistic stepwise automaton over $\mathcal{F}^{\mathbb{Q}}$ and $\mathcal{H} = (\Sigma, Q, F, \delta^h)$ be the weighted hedge automaton defined in the proof of Proposition 2. Let $\mathcal{S}_{f,q}$ be the automaton associated with the rule $f(\pi_{f,q}) \to q$ in $\mathcal{H}$. Let

$$s_{f,q} = \sum_{u \in Q^*} \pi_{f,q}(u) \ .$$

According to [4, Prop. 17], $s_{f,q}$ is an element of $\mathbf{R}_+$. We first prove that

Fact 1. $\sum_{f \in \Sigma} s_{f,q} = 1$.

Proof. Given a state $\widehat{q} \in Q$ of $\mathcal{A}$, let $\mathcal{S}_{\widehat{q}} = (\Sigma_{\widehat{q}}, Q_{\widehat{q}}, \iota_{\widehat{q}}, \varphi_{\widehat{q}}, \tau_{\widehat{q}})$ be the weighted string automaton defined by

- $\Sigma_{\widehat{q}} = Q$,
- $Q_{\widehat{q}} = Q \cup \Sigma$,
- $\iota(f) = 1$ whenever $f \in \Sigma$, $\iota(q) = 0$ otherwise,
- $\delta(f, \epsilon, q) = w_{f \to q}$ whenever $f \in \Sigma$ and $\delta(q_1, q_2, q_3) = w_{@(q_1,q_2) \to q_3}$ whenever q_1 and q_3 both are in Q.
- $\tau(\widehat{q}) = 1$, $\tau(q) = 0$ when $q \neq \widehat{q}$.

Then $\mathcal{S}_{\widehat{q}}$ is a probabilistic automaton. Let $\pi_{\widehat{q}}$ be the series defined by $\mathcal{S}_{\widehat{q}}$. By definition, $\mathcal{S}_{\widehat{q}}$ is clearly equivalent to the union of all $\mathcal{S}_{f,\widehat{q}}$. Thus for every $u \in Q^*$, $\sum_{f \in \Sigma} \pi_{f,\widehat{q}}(u) = \pi_{\widehat{q}}(u)$.

Moreover, $\sum_{u \in Q^*} \pi_q(u) = 1$ [14]. Thus

$$\sum_{f \in \Sigma} s_{f,\widehat{q}} = \sum_{f \in \Sigma} \sum_{u \in Q^*} \pi_{f,\widehat{q}}(u) = \sum_{u \in Q^*} \sum_{f \in \Sigma} \pi_{f,\widehat{q}}(u) = \sum_{u \in Q^*} \pi_{\widehat{q}}(u) = 1.$$

□ □

Now, let us define the automaton $\mathcal{H}' = (\Sigma, Q, F, \delta'^h)$ by defining rules

$$f(\pi'_{f,q}) \to q\ [s_{f,q}]$$

for all $s_{f,q} \neq 0$. The series $\pi'_{f,q}$ is defined as in the proof of Proposition 2, except for the termination function $\tau'_{f,q}$ which is defined by $\tau'_{f,q} = \tau_q \cdot s_{f,q}^{-1}$.

$\mathcal{H}'$ satisfies the conditions of Definition 7:

- $\forall f \in \Sigma$, $\forall q \in Q$, $w_{f,q} \in \mathbf{R}_+$;
- $\forall q \in Q$, $\sum_{f\in\Sigma} w_{f,q} = 1$;
- According to Fact 1,$\sum_{q\in Q} F(q) = 1$.
- The series $\pi'_{f,q}$ is an $\mathbf{R}_+$-string series and $\sum_{u\in Q^*} \pi'_{f,q}(u) = s_{f,q}^{-1} \cdot \sum_{u\in Q^*} \pi_{f,q}(u) = 1$. As a consequence, $\pi'_{f,q}$ can be defined by a probabilistic automaton [4, Prop. 18].

Finally for every t, $r_{\mathcal{H}'}(t) = r_{\mathcal{H}}(t) = r_{\mathcal{A}}(\bar{t})$. □ □

5 Convergence of real-valued tree series

5.1 Convergence and Absolute Convergence

For applications, we are interested in recognizable tree series over $\mathbf{R}$ which define probability distributions. But such series may map trees to negative numbers. This makes harder the computation of the sum of a series. Therefore, a first step is to define convergent tree series according to some fixed ordering over trees. We propose to order trees by increasing size:

Definition 8. *Let r be an $\mathbf{R}$-tree series over $T_{\mathcal{F}}$ (or over T^u_Σ), r is* convergent *if the sequence of partial sums $\left(\sum_{|t|\leq n} r(t)\right)_{n\geq 1}$ has a finite limit. If r is convergent then*

$$\sum_{t\in T_{\mathcal{F}}} r(t) = \sum_{n=1}^{n=\infty} \left(\sum_{|t|=n} r(t) \right) .$$

We have chosen this definition because it is robust with respect to the binary representation of unranked trees:

Proposition 4. *Let r be an $\mathbf{R}$-tree series over T^u_Σ and let $\overline{r}$ be the $\mathbf{R}$-tree series over $T_{\mathcal{F}^@}$ defined by: $\forall \bar{t} \in T_{\mathcal{F}^@}$, $\overline{r}(\bar{t}) = r(t)$ where $\bar{t}$ is the binary representation of t. Then r is convergent if and only if $\overline{r}$ is convergent. Note that, given an $\mathbf{R}$-tree series $\overline{r}$ over $T_{\mathcal{F}^@}$, the $\mathbf{R}$-tree series r over T^u_Σ is uniquely defined.*

The proposition holds because, for every $t, t' \in T^u_\Sigma$, $|t| = |t'|$ if and only if $\left|\bar{t}\right| = \left|\overline{t'}\right|$. Indeed, for every $t \in T^u_\Sigma$, $\left|\bar{t}\right| = 2\,|t| - 1$.

Absolute convergence of tree series can be defined: an $\mathbf{R}$-tree series r is said to *converge absolutely* if $\sum_{t\in T_{\mathcal{F}}} |r(t)| < \infty$. An absolutely convergent $\mathbf{R}$-tree series is convergent according to our definition and also according to any reordering of terms in the infinite sum.

5.2 Strong Convergence

The notion of *strong convergence* was introduced in [15]:

Definition 9. *An* $\mathbf{R}$*-tree series* r *is said to be* strongly convergent *if the series* y *defined by* $y(t) = |t| \times r(t)$ *is convergent. If* r *is strongly convergent then*

$$\sum_{t \in T_{\mathcal{F}}} y(t) = \sum_{t \in T_{\mathcal{F}}} |t| \times r(t) = \sum_{n=1}^{n=\infty} \left(\sum_{|t|=n} n \times r(t) \right).$$

First, we prove that:

Proposition 5. *If an* $\mathbf{R}$*-tree series* r *is strongly convergent then* r *is convergent.*

Proof. Let r be an $\mathbf{R}$-tree series. Let (r_n) and (y_n) be the sequences of partial sums defined by:

$$r_n = \sum_{\substack{t \in T_{\mathcal{F}} \\ |t|=n}} r(t) \qquad\qquad y_n = n \times r_n = \sum_{\substack{t \in T_{\mathcal{F}} \\ |t|=n}} n \times r(t).$$

The strong convergence of r ensures that the series $\sum y_n$ converges. Because $r_n = y_n \times \frac{1}{n}$ and the sequence $\frac{1}{n}$ decreases towards zero, Abel's test ensures the series $\sum r_n$ to be convergent. Thus the $\mathbf{R}$-tree series r is convergent. □ □

The definition of strong convergence is robust with respect to the binary representation of unranked trees:

Proposition 6. *Let* r *be an* $\mathbf{R}$*-tree series on* T^u_Σ *and let* $\overline{r}$ *be the* $\mathbf{R}$*-tree series on the binary representations. Then* r *is strongly convergent if and only if* $\overline{r}$ *is strongly convergent.*

Proof. Let r be an $\mathbf{R}$-tree series on T^u_Σ and let $\overline{r}$ be the equivalent series on the binary representations, i.e. $\forall t \in T^u_\Sigma$, $\overline{r}(\overline{t}) = r(t)$. Let us consider the sequences of partial sums defined by: $r_n = \sum_{|t|=n} r(t)$; $\overline{r}_n = \sum_{|\overline{t}|=n} \overline{r}(\overline{t})$; $y_n = n \cdot r_n$; and $\overline{y}_n = n \cdot \overline{r}_n$. We have $\left|\overline{t}\right| = 2\,|t| - 1$ and trees in $T_{\mathcal{F}^{@}}$ are binary. Thus, $\overline{y}_{2n-1} = (2n-1) \sum_{|t|=2n-1} r(t)$ and $\overline{y}_n = 0$ whenever $n \notin 2\mathbf{N} - 1$.

If the $\mathbf{R}$-tree series $\overline{r}$ is strongly convergent then $\sum \overline{y}_n$ converges. From Proposition 5, we obtain that $\sum \overline{r}_n = \sum r_n$ is convergent. Because $\sum r_n$ is convergent and $\sum (2n-1) r_n$ is convergent, as the sum of convergent series is convergent, the series $\sum ((2n-1) r_n + r_n)$ is convergent. This leads to say that the series $\sum 2 \cdot y_n$ is convergent. Thus $\sum y_n$ converges, i.e. the $\mathbf{R}$-tree series r is strongly convergent.

Conversely, if the $\mathbf{R}$-tree series r is strongly convergent then $\sum y_n$ converges. $\sum r_n$ converges and as a consequence $\sum (2n-1) r_{2n-1} = \sum \overline{y}_n$ converges. Thus the $\mathbf{R}$-tree series $\overline{r}$ is strongly convergent. □ □

We will present in the next section a recognizable $\mathbf{R}_+$-tree series which is convergent but not strongly convergent.

6 Recognizable Stochastic Tree Series

A *recognizable stochastic tree series* r is a recognizable $\mathbf{R}$-tree series over $T_{\mathcal{F}}$ which defines a probability distribution over $T_{\mathcal{F}}$, i.e. $\forall t \in T_{\mathcal{F}}$, $0 \leq r(t) \leq 1$ and $\sum_{t \in T_{\mathcal{F}}} r(t) = 1$. A recognizable tree series is called a *recognizable stochastic tree language* in papers by Denis et al. For probabilistic context-free grammars, the term *consistent* was introduced in [16] for probabilistic context-free grammars defining probability distributions, while Chi et al [17, 18] use the term of *proper* distributions.

Recognizable stochastic tree series can be defined by weighted tree automata over $\mathbf{R}$, or by weighted tree automata over $\mathbf{R}_+$, or by probabilistic tree automata. We will compare the corresponding classes of probability distributions in Section 7. First, we consider the question whether a weighted tree automaton defines a stochastic tree series, we present some decidability results and show how to compute sums of series.

6.1 Probabilistic Tree Automata and Stochastic Tree Series

In the string case, every probabilistic automaton over Σ defines a probability distribution over Σ^*. In the tree case, this property does not hold. Indeed, let us consider the example used in [16][1].

Let $\mathcal{F}^{@} = \{@, a\}$, and let $\mathcal{A}_\alpha = (\mathcal{F}^{@}, Q, F, \delta_\alpha)$ be a family of probabilistic stepwise automata parameterized by $\alpha \in [0, 1]$ and defined by: $Q = \{q\}$, $F(q) = 1$, $\delta_\alpha(a, q) = 1 - \alpha$, and $\delta_\alpha(@, q, q, q) = \alpha$. Let r_α be the recognizable tree series defined by $\mathcal{A}_\alpha$. Let $s_n = \sum_{\mathsf{height}(t) \leq n} r_\alpha(t)$. It is easy to show: $\forall n$, $s_n \leq \min\{1, \frac{1-\alpha}{\alpha}\}$ and $s_{n+1} = \alpha \times s_n^2 + (1 - \alpha)$. Thus r_α is convergent and $\sum_{t \in T_{\mathcal{F}^{@}}} r_\alpha(t) = 1$ if and only if $\alpha \leq \frac{1}{2}$. In the case $\alpha > \frac{1}{2}$, the limit is strictly lesser than 1 because infinite trees have a non zero weight. Thus,

Fact 2. *There is a probabilistic tree automaton which does not define a stochastic tree series.*

We will consider in the next subsection whether it is decidable that a probabilistic automaton defines a stochastic tree series. But, before we compare the convergence notions. Let us again consider series r_α, and let us define $u_n = \sum_{\mathsf{height}(t) \leq n} n \times r_\alpha(t)$, algebraic computation allows to show that $u_{n+1} = 2\alpha \times s_n \times u_n + \alpha \times s_n^2 + (1 - \alpha)$. When $\alpha \leq \frac{1}{2}$, s_n converges to 1, then the limit u of u_n must satisfy $u = 2\alpha \times u + 1$. Thus, for $\alpha = \frac{1}{2}$, the series is not strongly convergent. Thus,

Fact 3. *There is a recognizable stochastic tree series defined by a probabilistic tree automaton which is not strongly convergent.*

Let us summarize the example. For the series r_α defined as above:

- For every $\alpha < \frac{1}{2}$, the series r_α is stochastic, it is strongly convergent and the sum u is equal to $\frac{1}{1-2\alpha}$ which is the expected value of the size of trees according to the probability distribution defined by r_α;

[1] The original example was given for probabilistic context-free grammars.

- For $\alpha = \frac{1}{2}$, the series r_α is stochastic but it is not strongly convergent
- For every $\alpha > \frac{1}{2}$, as $r_\alpha = \frac{1-\alpha}{\alpha} \times r_{1-\alpha}$, the series r_α is strongly convergent but it is not a stochastic series because its sum $\frac{1-\alpha}{\alpha}$ is strictly lower than 1.

Last, we show that

Fact 4. *There exists a strongly convergent tree series which is not absolutely convergent.*

Indeed, consider a series r such that $r_n = \sum_{|t|=n} r(t) = \frac{(-1)^n}{n \cdot \ln n}$. Then $\sum n \cdot r_n = \sum \frac{(-1)^n}{\ln n}$, which is convergent. However, $\sum |r_n| = \sum \frac{1}{n \cdot \ln n}$ is not convergent. We are not aware of a recognizable strongly convergent tree series which is not absolutely convergent.

6.2 Decidability Results

As weighted tree automata (even probabilistic tree automata) do not necessarily define stochastic series, we consider the decision problems:

Nonnegativity problem

Instance: a weighted tree automaton $\mathcal{A}$ over $\mathbf{R}$.

Answer: "yes" if and only if the recognizable tree series $r_\mathcal{A}$ is nonnegative, i.e. $\forall t \in T_\mathcal{F}$, $r_\mathcal{A}(t) \geq 0$.

Stochasticity problem for weighted tree automata

Instance: a weighted tree automaton $\mathcal{A}$ over $\mathbf{R}$.

Answer: "yes" if and only if the recognizable tree series $r_\mathcal{A}$ is stochastic.

The nonnegativity problem is undecidable. Indeed, the problem is undecidable for weighted string automata. It is immediate from the undecidability of the emptiness problem for (string) probabilistic automata (proof in [19, 20]). A consequence, shown in [4], is that it is undecidable whether a weighted string automaton over $\mathbf{R}$ defines a probability distribution. Therefore, the stochasticity problem for weighted tree automata is also undecidable.

Let us now consider the – specific to trees – decision problem:

Stochasticity problem for probabilistic tree automata

Instance: a probabilistic tree automaton $\mathcal{A}$.

Answer: "yes" if and only if the recognizable tree series $r_\mathcal{A}$ is stochastic.

This problem is decidable in polynomial time as a consequence of deciding in polynomial time whether the probability of termination of 1-exit recursive Markov chain is 1, this is proved in [21] and a decision algorithm for probabilistic context-free grammars is proposed. We adapt it (it is very easy) to probabilistic tree automata. Let $\mathcal{A} = (\mathcal{F}, Q, F, \delta)$ be a probabilistic automaton over $\mathcal{F}$ and let us suppose that $Q = \{q^1, \ldots, q^n\}$. We can view $\mathcal{A}$ as a top-down automaton

(equivalently a regular tree grammar): consider the mapping δ as a set Δ of rules of the following form

$$rule : q_{rule} \to f(q_{rule1}, \ldots, q_{rulen})\ (w_{rule})\ ,$$

where $w_{rule} = \delta(f, q_{rule1}, \ldots, q_{rulen}, q_{rule})$; consider states in Q such that $F(Q) \neq 0$ as initial states. Let $A = (a_{ij})_{1 \leq i,j \leq n}$ be the matrix defined by

$$a_{ij} = \sum_{rule \in \Delta, q_{rule} = q^i} w_{rule} \times n_{rule}(q^j)$$

where $n_{rule}(q^j)$ is the number of occurrences of state q^j in the right-hand side of $rule$, i.e. in the state sequence $(q_{rule1}, \ldots, q_{rulen})$. The spectral radius $\rho(A)$ of a square matrix A is the maximum among the modulus of eigenvalues (real or complex) of A. Moreover, we can suppose that $F(q^1) = 1$. Following [21], a polynomial time algorithm for the stochasticity problem for probabilistic tree automata is:

Input: a probabilistic tree automaton $\mathcal{A}$ over $\mathcal{F}$.
Output: *yes* if $\mathcal{A}$ defines a probability distribution, *no* otherwise.
1. Compute accessible states and remove all states unaccessible.
2. Compute productive states. If there is a state which is not productive then return *no*.
3. For the remaining probabilistic weighted automaton, compute the matrix A and its spectral radius $\rho(A)$. If $\rho(A) > 1$ then return *no*, otherwise return *yes*.

Figure 3: decision algorithm for stochasticity of probabilistic tree automata

The correctness of the algorithm is a direct consequence of the proof of correctness of a similar algorithm for stochastic context-free grammars given in [21], itself obtained as a consequence of [Theorem 8.1, [21]] for 1-exit recursive markov chains. First, let us note that computations in steps 1 and 2 can be done with classical algorithms for (non weighted) tree automata. It is because for probabilistic tree automata checking accessibility and productivity of states can be done only with the structure of rules. It is no more true when weights can be negative. Indeed, testing whether a state is productive can not be done only on the structure of rules but sums must be computed and compared with 0. Second, for step 3, it is decidable in polynomial time whether the spectral radius of a square matrix is strictly less than 1. The correctness is related to the polynomial system of equations associated with a probabilistic tree automaton that we discuss in the next subsection.

Example. Let us consider a family of probabilistic automata $\mathcal{A}_{\alpha,\beta}$ parameterized by α, β with state set $Q = \{q^1, q^2\}$, $\mathcal{F} = \{@, a, b\}$, $F(q^1) = \frac{1}{3}, F(q^2) = \frac{2}{3}$, and $\Delta_{\alpha,\beta} = \{q^1 \to a(1-\alpha); q^1 \to @(q^1, q^2)(\alpha); q^2 \to b(1-\beta); q^2 \to @(q^2, q^2)(\beta)\}$. The states q^1 and q^2 are accessible. When $\alpha = 1$ (respectively $\beta = 1$), the state q^1 (respectively q^2) is not productive and the automaton does not define a probability distribution. When $\alpha \neq 1$ and $\beta \neq 1$, the matrix $A_{\alpha,\beta}$ is

$$A_{\alpha,\beta} = \begin{pmatrix} \alpha & \alpha \\ 0 & 2\beta \end{pmatrix},$$

the spectral radius of $A_{\alpha,\beta}$ is $\max\{\alpha, 2\beta\}$, and $\rho(A_{\alpha,\beta}) > 1$ if and only if $\alpha > 1$ and $\beta > \frac{1}{2}$. If we summarize, we get that $\mathcal{A}_{\alpha,\beta}$ defines a probability distribution if and only if $\alpha < 1$ and $\beta \leq \frac{1}{2}$.

We conclude by a property of strongly convergent tree series. Let us consider a probabilistic tree automaton $\mathcal{A}$ and the matrix A, and let us suppose that $\rho(A) < 1$. Then A^n converges to 0 and the expected value for the size of trees is finite and can be computed as $C \times (I - A)^{-1} \times B$ where $B = (1, \ldots, 1)^t$ and $C = (F(q^1), \ldots, F(q^n))$. This expected value is also equal to $\sum_{t \in T_{\mathcal{F}}} |t| \times r_{\mathcal{A}}(t)$.

Example (continued). When $\alpha < 1$ and $\beta < \frac{1}{2}$, we have

$$(I - A)^{-1} = \begin{pmatrix} \frac{1}{1-\alpha} & \frac{\alpha}{(1-\alpha)(1-2\beta)} \\ 0 & \frac{1}{1-2\beta} \end{pmatrix}$$

and the expected value for the size of trees is $\frac{1}{3}\left(\frac{1}{1-\alpha} + \frac{\alpha}{(1-\alpha)(1-2\beta)}\right) + \frac{2}{3}\left(\frac{1}{1-2\beta}\right)$.

6.3 Computing the Sum of a Tree Series

Let r be a recognizable tree series defined by a weighted tree automaton $\mathcal{A}$. For every state q^i, the *state probability* of q^i is the sum of the series for automaton $\mathcal{A}^i$ obtained from $\mathcal{A}$ by modifying F in F^i defined by $F^i(q^i) = 1$. For every state q^i, we define a variable x_i. We can construct a system of polynomial equations $\vec{x} = P(\vec{x})$ over variables x_i. If exist, the state probabilities are solutions of this system of equations. When $\mathcal{A}$ is probabilistic, according to [21], the system has only positive coefficients, there is a least fixed point solution, and the least fixed point solution is the vector of state probabilities. Thus, the system of equations gives rise to an iterative numerical algorithm with which to approximate the least fixed point, thus to compute the sum of a recognizable tree series defined by a probabilistic tree automaton. In the same paper, they also provide an iterative numerical algorithm, based on a decomposed Newton's method.

Example (continued). For our running example, the system of polynomial equations is:

$$\begin{cases} x_1 &= \alpha \times x_1 \times x_2 + (1 - \alpha) \\ x_2 &= \beta \times x_2^2 + (1 - \beta) \end{cases}$$

It can be solved analytically leading to $x_2 = \min\{1, \frac{1-\beta}{\beta}\}$, $x_1 = \frac{1-\alpha}{1-\alpha \times x_2}$. When the automaton is strongly consistent, i.e. when $\alpha < 1$ and $\beta < \frac{1}{2}$, then $x_1 = x_2 = 1$. Let s be the sum of the series $r_{\mathcal{A}_{\alpha,\beta}}$, we have $s = \frac{1}{3} \times x_1 + \frac{2}{3} \times x_2 = 1$. In general, solutions can be approximated by computing x_{1_n} and x_{2_n} for n large enough where $x_{1_1} = 1 - \alpha$, $x_{2_1} = 1 - \beta$, $x_{1_{n+1}} = \alpha \times x_{1_n} \times x_{2_n} + (1 - \alpha)$, and $x_{2_{n+1}} = \beta \times x_{2_n}^2 + (1 - \beta)$. We should also note that it is proved (Theorem 3.2 in [21]) that it requires an exponential number of iterations to compute the sum of the series xithin k bits of precision when $\beta = \frac{1}{2}$.

7 Expressiveness

7.1 Tree Series and Weighted Automata

A recognizable $\mathbf{R}$-tree series is a series defined by a weighted tree automaton over $\mathbf{R}$. It is said to be *nonnegative* if $\forall t \in T_{\mathcal{F}}$, $0 \leq r(t)$. We recall that it is said to be stochastic if $\forall t \in T_{\mathcal{F}}$, $0 \leq r(t) \leq 1$ and $\sum_{t \in T_{\mathcal{F}}} r(t) = 1$. We compare classes of tree series depending on the class of weighted tree automata used to define them.

Let us consider tree series defined over $T_{\mathcal{F}}$ where $\mathcal{F}$ is a ranked alphabet. We use the letter $\mathcal{R}$ for recognizable tree series, $\mathcal{N}$ for non negative tree series, $\mathcal{S}$ for stochastic tree series. We use WTA for weighted tree automata over $\mathbf{R}$, WTA_+ for weighted tree automata over $\mathbf{R}_+$, and PTA for probabilistic weighted tree automata. We precise the class of weighted automata between parenthesis. For instance, $\mathcal{N}(WTA)$ is the set of nonnegative recognizable tree series over $T_{\mathcal{F}}$ defined by weighted tree automata over $\mathbf{R}$. We have the following inclusions

Proposition 7.

1. $\mathcal{R}(PTA) \subset \mathcal{R}(WTA_+) \subset \mathcal{R}(WTA)$
2. $\mathcal{N}(PTA) \subset \mathcal{N}(WTA_+) \subset \mathcal{N}(WTA)$
3. $\mathcal{S}(PTA) = \mathcal{S}(WTA_+) \subset \mathcal{S}(WTA)$
4. $\mathcal{S}(PTA) \subset \mathcal{N}(PTA) = \mathcal{R}(PTA)$

Proof.

1. obvious considering the series r_2 defined by $\forall t \in T_{\mathcal{F}}$, $r_2(t) = 2$ which is in $\mathcal{R}(WTA_+)$ but not in $\mathcal{R}(PTA)$, and the series r_{-1} defined by $\forall t \in T_{\mathcal{F}}$, $r_{-1}(t) = -1$ which is in $\mathcal{R}(WTA)$ but not in $\mathcal{R}(WTA_+)$.

2. $\mathcal{N}(PTA) \subset \mathcal{N}(WTA_+)$ using r_2. More interestingly, $\mathcal{N}(WTA_+) \subset \mathcal{N}(WTA)$ states that negative weights allows to define more nonnegative recognizable tree series. It is a consequence of the following inclusion.

3. $\mathcal{S}(WTA_+) \subset \mathcal{S}(WTA)$ states that negative weights allow to define stochastic recognizable tree series which can not be defined with nonnegative weights only. This is because this result already holds for string series as shown in [4]. The equality $\mathcal{S}(PTA) = \mathcal{S}(WTA_+)$ can be proved in two steps: first show that if a stochastic tree series is defined by a weighted tree automaton with nonnegative weights then there exists a trimmed weighted tree automaton with nonnegative weights computing the series. Then rules can be normalized. It should be noted that we do not know if the stochasticity problem for weighted tree automata with nonnegative weights is decidable. It should also be noted that normalization implies to compute sums of series.

   In [15], the authors introduce *normalized* weighted tree automata (denoted by WTA_n) for stochastic tree series where all tree series associated with states (for a state q, consider the series defined by the same automaton except that $F(q) = 1$) are stochastic. They show that $\mathcal{S}(WTA_n) = \mathcal{S}(WTA)$. It can be deduced that $\mathcal{S}(PTA) = \mathcal{S}(WTA_+)$.

4. $\mathcal{S}(PTA) \subset \mathcal{N}(PTA)$ because there are tree series defined by probabilistic tree automata for which the sum of weights of all terms is strictly less than 1.

□

7.2 Recognizable Tree Series and Recognizable Tree Languages

The characteristic series r_L of a language $L \subseteq T_{\mathcal{F}}$ is defined by $r_L(t) = 1$ if $t \in L$, and 0 otherwise. The characteristic series of a recognizable tree language is recognizable. An algebraic is given in [13]. Also it is easy to define a weighted tree automaton over $\mathbf{R}$ computing the tree series r_L given a bottom-up deterministic automaton for L.

The converse is false. Indeed, consider $\mathcal{F} = \{@, a, b\}$ and the weighted tree automaton $\mathcal{A}_a = (\mathcal{F}, Q, F, \delta)$ defined by: $Q = \{q_1, q_2\}$, $F(q_1) = 0$, $F(q_2) = 1$, $\delta(a, q_1) = 1$, $\delta(a, q_2) = 1$, $\delta(b, q_1) = 1$, $\delta(@, q_1, q_2, q_2) = 1$, $\delta(a, q_2, q_1, q_2) = 1$, and $\delta(a, q_1, q_1, q_1) = 1$, otherwise δ is zero. The automaton is non deterministic. It computes the series r_a defined by $r_a(t)$ is the number of occurrences of a in t because the automaton guesses one occurrence of a and there is one run for each occurrence of a which leads to state q_2. Now, similarly, the series r_b is recognizable. Thus the series $r_a - r_b$ is recognizable. But the set of trees such that $(r_a - r_b)(t) \neq 0$ (called the support of $r_a - r_b$) is not a recognizable tree language.

Let us also recall that the series *height* where $height(t)$ is the maximal length of paths in t is not recognizable. this can be proved by using a pumping lemma for recognizable tree series [13].

7.3 Tree Series and Deterministic Automata

As we have defined recognizable tree series by means of automata, the question of determinism arises. A weighted tree automaton $\mathcal{A} = (\mathcal{F}, Q, F, \delta)$ is said to be *(bottom-up) deterministic* if for every symbol $f \in \mathcal{F}$, for every tuple $(q_1, \ldots, q_n)$, there is a unique sate q such that $\delta(f, q_1, \ldots, q_n, q) \neq 0$. Let us use $WDTA$ for weighted deterministic tree automata over $\mathbf{R}$ and $PDTA$ for probabilistic deterministic weighted tree automata. While deterministic and non deterministic tree automata are equivalent, this is no more true for weighted tree automata.

Proposition 8.

1. $\mathcal{R}(WDTA) \subset \mathcal{R}(WTA)$
2. $\mathcal{S}(PDTA) \subset \mathcal{S}(PTA)$

Proof. 1. $\mathcal{R}(WDTA) \subset \mathcal{R}(WTA)$ because the series *size* defined by, for every $t \in T_{\mathcal{F}}$, $size(t) = |t|$ is recognizable. For instance, when $\mathcal{F} = \{@, a, b\}$, $size = 2(r_a + r_b) - 1$. And, in general $size = \sum_{f \in \mathcal{F}} r_f$, where r_f computes the number of occurrences of f in a tree. And r_f can be easily shown recognizable. The series *size* can not be defined by a deterministic weighted tree automaton. This is proved using a Myhill-Nerode theorem for recognizable tree series [22].

2. This inclusion already holds for strings [15].

□

A weighted tree automaton $\mathcal{A} = (\mathcal{F}, Q, F, \delta)$ is said to be *top-down deterministic* if for every state $q \in Q$, for every symbol $f \in \mathcal{F}$, there is a unique tuple $(q_1, \ldots, q_n)$ of states such that $\delta(f, q_1, \ldots, q_n, q) \neq 0$. Top-down determinism is a very strong condition. The set of recognizable (stochastic) tree series defined by top-down deterministic automata is strictly included in the set $\mathcal{R}(WDTA)$ ($\mathcal{R}(PDTA)$).

8 Implementing Tree Series for Unranked Trees

8.1 Size and memory

While it could seem that unranked trees are more efficient in memory than their binary counterpart, this would not be true. Indeed, consider for example that trees are written in a linear memory space of symbols. A binary tree written on such memory with the symbols $\mathcal{F} = \Sigma \cup \{@\}$ would only occupy as many symbols as it counts node. Indeed, reading would be unambiguous as the rank of the tree is fixed. Working directly on ranked trees would require to mark the end of a list of children by a special symbol /, which would need to appear exactly once per node. In this context, memory usages are the same.

Example 3. *While* $@\,@\,abc$ *is unambiguously parsed as* $@(@(a, b), c)$, $ffab$ *can be parsed as* $f(f, a, b)$, $f(f(a, b))$, $f(f(a(b)))$, *... However, if we mark the end of the children lists by /, then, e.g.,* $ffa/b///$ *can only be parsed as* $f(f(a, b))$.

8.2 Algorithmic considerations

In this section, we focus on algorithmic implications of the ranked/unranked correspondence. We show that any stream view algorithm working on unranked trees can be reduced to a stream view algorithm on binary representations.

XML data processing is done either through the stream view or the tree view. The former corresponds to SAX programming and the latter to DOM programming. Even in the latter case, inputs are serialized in a stream and need to be loaded with the former view anyway.

In the stream view processing, operations are realized in prefix, postfix or infix places. Each of these places corresponds to inner positions. We have seen in Section 4 and Figure 2 that inner positions correspond to nodes in the binary representation. Moreover, the order nodes are seen in the stream corresponds to a depth-first left to right traversal which is preserved in the binary representation. Thus operations realized at prefix, postfix or infix places can be interpreted as operations on the binary representation.

For instance, the evaluation of the weight of an unranked tree by a stepwise weighted automaton can be computed in the stream. Actions are performed when entering or leaving a node of the tree (the open and close tags in the stream):

- When entering a node labeled f, push on the stack the vector u whose components correspond to states such that the rules for f are $f \to q\ [u_q]$

- When leaving a node, pop w then v from the stack and push u such that $u_q = \sum_{\substack{q_1 \in Q \\ q_2 \in Q}} M^q_{q_1,q_2} v_{q_1} w_{q_2}$ where the rules for @ are $@(q_1, q_2) \to q\ [M^q_{q_1,q_2}]$.

After reading a whole tree, there will be a single vector u on the stack. The weight of the tree is then $\sum_{q \in Q} F(q) u_q$.

Conclusion

We have proved that binary representations of unranked trees are useful. Indeed, we have proved that the notions of recognizable tree series, stochastic tree series, convergence of tree series can be equivalently defined over unranked trees or over their binary representations. Moreover, we have seen that binary representations are rightly usable in streaming algorithms even when only having access to XML files without making the representation explicit.

However, there is still much research to do in this field both on the formal language side and on the machine learning side. For instance, many decision problems for convergence of tree series remain open and must be investigated. We have defined recognizable tree series for unranked trees with weighted tree automata. But defining recognizable tree series for unranked trees with linear representations remains challenging. Also learning tree series for unranked trees must be studied in more details.

References

[1] Denis, F., Habrard, A., Gilbert, É., Gilleron, R., Tommasi, M.: On probability distributions for trees: Representations, inference and learning. In: poster in NIPS Workshop on Representations and Inference on Probability Distributions. (2007)

[2] Carrasco, R.C., Oncina, J., Calera-Rubio, J.: Stochastic inference of regular tree languages. Machine Learning **44** (2001) 185–197

[3] Maletti, A.: Learning deterministically recognizable tree series — revisited. In: Proceedings 2nd International Conference on Algebraic Informatics. Volume 4728 of LNCS. (2007) 218–235

[4] Denis, F., Esposito, Y.: Rational stochastic languages. Fundamenta Informaticae **86** (2006) 41–77

[5] Habrard, A., Oncina, J.: Learning multiplicity tree automata. In: Grammatical Inference: Algorithms and Applications. Volume 4021 of LNCS. (2006) 268–280

[6] Denis, F., Habrard, A.: Learning rational stochastic tree languages. In: Algorithmic learning theory. Volume 4754 of LNAI., Springer-Verlag (2007) 242–256

[7] Thatcher, J.W.: Characterizing derivation trees of context-free grammars through a generalization of automata theory. Journal of Computer and System Sciences **1** (1967) 317–322

[8] Brüggemann-Klein, A., Wood, D., Murata, M.: Regular tree and regular hedge languages over unranked alphabets: Version 1 (April 07 2001)

[9] Comon, H., Dauchet, M., Gilleron, R., Jacquemard, F., Lugiez, D., Löding, C., Tison, S., Tommasi, M.: Tree automata techniques and applications. Available on: `http://tata.gforge.inria.fr/` (2007)

[10] Hogberg, J., Maletti, A., Vogler, H.: Bisimulation minimisation of weighted automata on unranked trees. To appear in Fundamenta Informaticae (2009)

[11] Martens, W., Niehren, J.: On the minimization of XML schemas and tree automata for unranked trees. Journal of Computer and System Sciences **73**(4) (2007) 550–583

[12] Borchard, B.: The Theory of Recognizable Tree Series. PhD thesis, Technische Universität Dresden (2004)

[13] Berstel, J., Reutenauer, C.: Recognizable formal power series on trees. Theorical computer science **18** (1982) 115–148

[14] Dupont, P., Denis, F., Esposito, Y.: Links between probabilistic automata and hidden markov models: probability distributions, learning models and induction algorithms. Pattern Recognition **38** (2005)

[15] Denis, F., Gilbert, É., Habrard, A., Ouardi, F., Tommasi, M.: Relevant representations for the inference of rational stochastic tree languages. In: Grammatical Inference: Algorithms and Applications. Volume 5278 of LNCS. (2008)

[16] Wetherell, C.S.: Probabilistic languages: A review and some open questions. ACM Comput. Surv. **12**(4) (1980) 361–379

[17] Chi, Z.: Statistical properties of probabilistic context-free grammars. Computational Linguistics **25**(1) (1999) 131–160

[18] Chi, Z., Geman, S.: Estimation of probabilistic context-free grammars. Computational Linguistics **24**(2) (1998) 299–305

[19] Paz, A.: Introduction to Probabilistic Automata. Academic Press (1971)

[20] Blondel, V.D., Canterini, V.: Undecidable problems for probabilistic automata of fixed dimension. Theory of computing systems **36**(3) (2003) 231–245

[21] Etessami, K., Yannakakis, M.: Recursive markov chains, stochastic grammars, and monotone systems of nonlinear equations. Journal of the ACM **56**(1) (2009) 1–65

[22] Borchardt, B.: The myhill-nerode theorem for recognizable tree series. In: Proceedings of Developments in Language Theory (DLT'O3). Volume 2710 of Lecture Notes in Computer Science. (2003) 146–158

Centre de recherche INRIA Lille – Nord Europe
Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399