A Probabilistic Kleene Theorem

Benedikt Bollig, Paul Gastin, Benjamin Monmege, Marc Zeitoun

A Probabilistic Kleene Theorem*

Benedikt Bollig[1], Paul Gastin[1], Benjamin Monmege[1], and Marc Zeitoun[2]

[1] LSV, ENS Cachan, CNRS & Inria, France,
firstname.lastname@lsv.ens-cachan.fr
[2] LaBRI, Univ. Bordeaux & CNRS, France, mz@labri.fr

Abstract. We provide a Kleene Theorem for (Rabin) probabilistic automata over finite words. Probabilistic automata generalize deterministic finite automata and assign to a word an acceptance probability. We provide probabilistic expressions with probabilistic choice, guarded choice, concatenation, and a star operator. We prove that probabilistic expressions and probabilistic automata are expressively equivalent. Our result actually extends to two-way probabilistic automata with pebbles and corresponding expressions.

1 Introduction

Kleene's Theorem states the equivalence of rational and recognizable languages in the free monoids. Naturally, this fundamental result has been generalized to various settings and in particular to quantitative extensions of classical languages, called formal power series [24,23,4].

The present paper aims at a probabilistic counterpart of Kleene's Theorem. There are actually a variety of models for probabilistic systems, comprising Segala systems, generative systems, stratified systems, Markov chains, etc. (see [29,25] for overviews). Those models may involve non-determinism and *generate* some behavior according to probability distributions over states. Alternatively, they may make a probabilistic decision depending on the input letter, like (reactive) probabilistic automata [20]. The latter go back to Rabin [21] and are an object of ongoing research considering decision problems such as emptiness, language equivalence [24,28,9,10,18], and the value 1 problem [17].

Our starting point of view is that expressions and automata shall represent quantitative properties of words. In particular, rather than at bisimulation equivalence, we are looking at language equivalence in terms of formal power series (i.e., mappings from strings to elements from the real-valued interval $[0,1]$). This actually has an immediate impact on the choice of both the automaton model and the syntax of expressions that are supposed to characterize it. On the automata side, a probabilistic decision should depend on the *given* input. Therefore, we consider probabilistic automata. On the specification side, we would like to adopt concepts from rational expressions. In this paper, we actually provide a simple fragment of classical weighted rational expressions over the

* Supported by ANR 2010 BLAN 0202 01 FREC, LIA InForMel.

nonnegative real numbers, including a star operator and concatenation. The star operator has to be handled with care, though. It comes with a subtle restriction to make sure that an expression associates with every word a probability. In this way, we obtain a class of probabilistic expressions that have the same expressive power as probabilistic automata. Translations forth and back are effective so that decidability results for automata directly carry over to expressions.

Actually, we prove a more general result. Expressions are extended in such a way that they capture two-way probabilistic automata [14,22] and automata with pebbles (similar to two-way word automata and tree-walking automata). Our expressions can then be considered as a probabilistic generalization of XPath [19,27]. Note that (non-probabilistic) two-way automata are in fact an appropriate machine model for compiling XPath queries [5]. The concept presented in this paper may therefore constitute a first step towards probabilistic database query languages: an expression is considered as a query, and an equivalent automaton can be used as a tool for evaluating queries efficiently (see [16] for recent developments on weighted query evaluation).

Related Work. It has to be noted that there have been numerous approaches to characterizing probabilistic systems in terms of algebraic expressions and process calculi [29,7,12,11]. A unifying framework is due to Silva et al. [26], who consider probabilistic systems in a general coalgebraic setting. This allows them to derive algebraic expressions and a corresponding Kleene Theorem, as well as full axiomatizations for many of those (and even for weighted automata over arbitrary semirings). Their and above-mentioned works are mainly aiming at axiomatization of probabilistic-system behaviors in terms of bisimulation equivalence, so their focus is on *system* models including non-determinism and generative probability distributions. In this paper, we consider probabilistic automata, which are a more appropriate machine model for our purpose, i.e., for *evaluating* queries. Moreover, while the syntax of process-algebraic expressions is tailored to modeling probabilistic systems and uses action prefixing, fixed points, and process variables, we provide expressions with concatenation and a proper Kleene star. Thus, our expressions are closer to language-theoretic operations and more convenient to use in query languages. So far, there have been only few attempts to define quantitative query languages. In [15], Flesca et al. introduce a weighted XPath. Their approach, however, does not extend to probabilistic automata. Note that the fact that we also consider two-way devices distinguishes our work from all above-mentioned references. To the best of our knowledge, we present the first Kleene-Schützenberger correspondence for probabilistic two-way automata.

Outline. In Section 2, after some motivating example, we recall the definition of (reactive) probabilistic automata, introduce our probabilistic regular expressions, and present a corresponding Kleene Theorem. A part of the proof of this theorem is postponed to Section 4, where a more general result is shown for two-way probabilistic automata and corresponding generalized expressions (which are introduced in Section 3): automata and expressions are expressively equivalent and can be transformed effectively into each other.

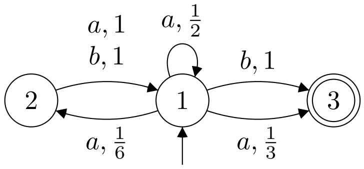


Fig. 1. A probabilistic automaton equivalent to $\left[\frac{1}{6}a(a+b)+\frac{1}{2}a\right]^* \cdot (\frac{1}{3}a+b)$

2 Probabilistic Automata and Expressions

Preliminaries. We fix a finite alphabet A. We restrict our attention to nonempty words over A, i.e., sequences $w = a_0 \cdots a_{n-1} \in A^+$ with $n \geq 1$ and $a_i \in A$ for every i. The *length* n of w is denoted $|w|$, and $\mathrm{pos}(w) = \{0, \ldots, |w|\}$ is the set of *positions* of w. The last position of index n is added to facilitate the definition of runs in two-way automata. To be able to deal with infinite sums over the non-negative real numbers, we extend $\mathbb{R}_{\geq 0}$ to $\mathbb{R}_{\geq 0}^{\infty} = \mathbb{R}_{\geq 0} \cup \{\infty\}$. In other words, we consider the *continuous semiring* $(\mathbb{R}_{\geq 0}^{\infty}, +, \cdot, 0, 1)$ where ∞ is assigned to any infinite sum that does not converge.

2.1 Probabilistic Automata

We consider classical probabilistic finite automata (PFA) [21,20]. A PFA over alphabet A is a tuple $\mathcal{A} = (Q, \iota, Acc, \mathbb{P})$ where Q is the finite set of states, $\iota \in Q$ is the initial state, and $Acc \subseteq Q$ is the set of final states. Moreover, $\mathbb{P} : Q \times A \times Q \to [0,1]$ is a function that assigns a probability to each transition. PFA are *reactive* automata, whose probabilistic choice depends on the current input letter. Thus, we require that $\sum_{q' \in Q} \mathbb{P}(q, a, q') \leq 1$ for all $(q, a) \in Q \times A$. We will moreover assume that a final state has no outgoing transitions with positive probability: $\mathbb{P}(q, a, q') > 0$ implies $q \notin Acc$. An example PFA is depicted in Fig. 1 where 1 is the initial state and 3 is the only final state. Transitions with probability 0 are omitted.

An *accepting run* of $\mathcal{A}$ over $w = a_0 \cdots a_{n-1} \in A^+$ is a sequence of transitions $\rho = \delta_1 \cdots \delta_n$ such that $\delta_i = (q_{i-1}, a_{i-1}, q_i)$ with $q_0 = \iota$ and $q_n \in Acc$. Given a run ρ, we set $\mathbb{P}(\rho) = \prod_{i=1}^{n} \mathbb{P}(\delta_i)$. The semantics of the PFA $\mathcal{A}$ is the mapping (series) $[\![\mathcal{A}]\!] : A^+ \to [0,1]$ given by $[\![\mathcal{A}]\!](w) = \sum_{\rho} \mathbb{P}(\rho)$, where the sum ranges over all accepting runs ρ over w. For instance, given the automaton of Fig. 1, we have $[\![\mathcal{A}]\!](aab) = \frac{1}{4} + \frac{1}{6} = \frac{5}{12}$.

2.2 Probabilistic Expressions

While PFAs are a machine model, we are aiming at denotational probabilistic regular expressions with the same expressiveness as PFAs. We start with the definition of classical weighted expressions (WEs) given by the syntax

$$E ::= s \mid a \mid E + E \mid E \cdot E \mid E^*$$

with $s \in \mathbb{R}^{\infty}_{\geq 0}$ and $a \in A$. In the following, we often simply write EF instead of $E \cdot F$. Also, we let $E^0 \overset{\text{def}}{=} 1$ and $E^{m+1} = EE^m$ for $m \geq 0$. The semantics of a WE E is a mapping $[\![E]\!] : A^+ \to \mathbb{R}^{\infty}_{\geq 0}$ which is defined inductively by

$$[\![s]\!](w) = \begin{cases} s & \text{if } w = \varepsilon \\ 0 & \text{otherwise} \end{cases} \qquad [\![a]\!](w) = \begin{cases} 1 & \text{if } w = a \\ 0 & \text{otherwise} \end{cases}$$

$$[\![E_1 + E_2]\!](w) = [\![E_1]\!](w) + [\![E_2]\!](w) \quad [\![E^*]\!](w) = \textstyle\sum_{m \in \mathbb{N}} [\![E^m]\!](w)$$

$$[\![E_1 \cdot E_2]\!](w) = \sum_{w=uv} [\![E_1]\!](u) \cdot [\![E_2]\!](v)\,.$$

In the following, we consider expressions modulo the following trivial identities:

$$0 + E \equiv E + 0 \equiv E \qquad E \cdot 0 \equiv 0 \cdot E \equiv 0 \qquad E \cdot 1 \equiv 1 \cdot E \equiv E \qquad 0^* \equiv 1$$

We introduce below probabilistic regular expressions (PREs) as a fragment of WEs. We have to restrict WEs since otherwise values greater than 1 could be obtained. For instance, the WE $(a + ab)(ba + a)$ should not be a PRE since it evaluates to 2 on the word aba. The restriction will be both on sum and star. Since we aim at PREs which are equivalent to PFAs, let us examine first which type of WEs are obtained from PFAs. A transition $\delta = (q, a, q')$ with probability $\mathbb{P}(\delta) = s$ could be denoted by the expression sa. Applying classical algorithms to build a regular expression from finite-state automata, we then obtain, for the automaton in Fig. 1, the expression $\left[\frac{1}{6}a(a+b) + \frac{1}{2}a\right]^* \cdot (\frac{1}{3}a + b)$. Now, the expression $\left[\frac{1}{6}a(a+b) + \frac{1}{2}a\right]^* \cdot (a + b)$, obtained by changing the subexpression $\frac{1}{3}a$ into a, should be disallowed, because it corresponds to an automaton violating condition $\sum_{q' \in Q} \mathbb{P}(q, a, q') \leq 1$. On the other hand, $\left[\frac{1}{6}a(a+b) + \frac{1}{2}a\right]^* \cdot (\frac{1}{3}a + \frac{1}{2}b)$ would be acceptable: we obtain a corresponding PFA from the automaton depicted in Fig. 1 by setting $\mathbb{P}(1, b, 3) = \frac{1}{2}$.

Definition 1. Probabilistic regular expressions (PREs) *is the fragment of* WEs *built inductively as follows:*

(Atoms) $s \in [0, 1]$ *and* $a \in A$ *are* PREs.

$(+_a)$ *If* $(E_a)_{a \in A}$ *are* PREs, *then* $\sum_{a \in A} a \cdot E_a$ *is a* PRE.

$(+_s)$ *If* E *and* F *are* PREs *and* $s \in [0, 1]$, *then* $s \cdot E + (1 - s) \cdot F$ *is a* PRE.

$(\cdot)$ *If* E *and* F *are* PREs, *then* $E \cdot F$ *is a* PRE.

$(*)$ *If* $E + F$ *is a* PRE, *then* $E^* \cdot F$ *is a* PRE.

(ACD) *Every* WE *that is obtained from a* PRE *by applying commutativity of* $+$, *associativity of* $+$ *or* $\cdot$, *or distributivity of* $\cdot$ *over* $+$ *is a* PRE.

There are two *guarded* sums. The first one $(+_a)$ is deterministic and guarded by the next letter to be read. The second one $(+_s)$ is probabilistic. Also, the star operation contains an implicit choice which is either to iterate again the expression or to exit the loop. This choice also has to be guarded which is the reason for the precondition $E + F \in \mathsf{PRE}$ in the rule $(*)$. The guard could be deterministic as in $(ab)^*b$ or probabilistic as in $(\frac{1}{3}(aa + bb))^* \frac{2}{3}(a + b)$. Finally, with the above restrictions, we lose the classical ACD identities, hence we enforce these properties explicitely with the ACD-rules which allow to rewrite a PRE in order to apply the star rule as needed.

Since PREs form a fragment of WEs, the semantics is inherited. From Theorem 1 below, PREs are equivalent to PFAs. We deduce that the semantics of $E \in \mathsf{PRE}$ takes values in $[0,1]$, so that one can interpret $[\![E]\!](w)$ as a probability.

Example 1. A simple PRE is $(\frac{1}{3}a)^* \cdot \frac{2}{3}b$, which assigns to a word $a^m b$ the probability $(\frac{1}{3})^m \cdot \frac{2}{3}$, and 0 to words not in a^*b. Moreover, $E = \left[\frac{1}{6}a(a+b) + \frac{1}{2}a\right]^* \cdot (\frac{1}{3}a + b)$ is indeed a PRE for the automaton from Fig. 1. To show that E is a PRE, we use some semantical equivalences such as $sa \equiv as$ or $\frac{5}{6} \equiv \frac{1}{2} + \frac{1}{3}$. The expression $a(\frac{1}{6}(a+b) + \frac{5}{6}) + b$ uses two deterministic sums (first and third) and a probabilistic sum. Using the above semantical equivalences and ACD-rules, we deduce that $\frac{1}{6}a(a+b) + \frac{1}{2}a + \frac{1}{3}a + b$ is a PRE and it remains to apply the star rule to get E.

In order to construct a PRE which is equivalent to a PFA, we need to be able to concatenate a PRE after an arbitrary term of another PRE. This is possible thanks to the following result.

Proposition 1. *If $E + F$ and G are* PREs*, then $E + F \cdot G$ is also a* PRE*.*

For a PFA, we can always find an equivalent PRE, and vice versa. This first theorem, which is non-trivial even in the one-way setting, is generalized in the next sections allowing two-way moves and pebbles.

Theorem 1. PFAs *and* PREs *are effectively equivalent.*

Proof. We only prove the translation from automata to expressions. The other direction will be proved in a more general setting in Section 3.

Let $\mathcal{A} = (Q, \iota, Acc, \mathbb{P})$ be a PFA. For each $q \in Q \setminus Acc$ we construct a PRE $E_q = \sum_{q' \in Acc} E_{q,q'}$ where $[\![E_{q,q'}]\!](w)$ computes the sum of the probabilities of *nonempty* runs over w starting from state q, ending in state q'. Hence, we will obtain the PRE E_ι, which computes exactly the behavior of $\mathcal{A}$.

We follow usual procedures to translate automata into expressions. For $q' \in Q$ and $X \subseteq Q \setminus Acc$, we define $f_{q'}^X = 0$ if $q' \in X$ and 1 otherwise. For $q \in Q \setminus Acc$ and $X \subseteq Q \setminus Acc$, we construct by induction on X a PRE $E_q^X = \sum_{q' \in Q} E_{q,q'}^X f_{q'}^X$ where $E_{q,q'}^X$ is a PRE such that $[\![E_{q,q'}^X]\!](w)$ is the sum of the probabilities of *nonempty* runs over w starting from state q, ending in state q' and using only *intermediary* states in X. Hence, we have $E_q = E_q^{Q \setminus X}$ and $E_{q,q'} = E_{q,q'}^{Q \setminus X}$.

The base of the induction is when $X = \emptyset$. For each state $q \in Q \setminus Acc$ and letter $a \in A$, by definition of PFAs we have $\sum_{q' \in Q} \mathbb{P}(q,a,q') \leq 1$. Hence, using rules $(+_a)$ and $(+_s)$ we obtain the PRE

$$E_q^\emptyset = \sum\nolimits_{a \in A} a \cdot \sum\nolimits_{q' \in Q} \mathbb{P}(q,a,q') = \sum\nolimits_{q' \in Q} E_{q,q'}^\emptyset f_{q'}^\emptyset$$

where the last equality is obtained using ACD-rules and $f_{q'}^\emptyset = 1$.

For the induction step, let $X \cup \{r\} \subseteq Q \setminus Acc$ with $r \notin X$. By induction, we assume that PREs E_q^X have been constructed for all $q \in Q \setminus Acc$, and we construct $E_q^{X\cup\{r\}}$. We have $E_r^X = \sum_{q'\in Q} E^X_{r,q'} f^X_{q'} \in \mathsf{PRE}$ and $f_r^X = 1$ since $r \notin X$. Using rule $(*)$, we get $G_r^X = \big(E^X_{r,r}\big)^* \cdot \big(\sum_{q'\in Q\setminus\{r\}} E^X_{r,q'} f^X_{q'}\big) \in \mathsf{PRE}$. Now, $E_q^X = \sum_{q'\in Q} E^X_{q,q'} f^X_{q'} \in \mathsf{PRE}$ and $f_r^X = 1$. Using Proposition 1, we can plug G_r^X after $E^X_{q,r}$ and we obtain the PRE

$$\begin{aligned} E_q^{X\cup\{r\}} &= E^X_{q,r} \cdot G^X_r + \sum\nolimits_{q'\in Q\setminus\{r\}} E^X_{q,q'} f^X_{q'} \\ &= \sum\nolimits_{q'\in Q\setminus\{r\}} \big(E^X_{q,q'} + E^X_{q,r}\big(E^X_{r,r}\big)^* E^X_{r,q'}\big) f^X_{q'} && \text{(ACD-rules)} \\ &= \sum\nolimits_{q'\in Q} E^{X\cup\{r\}}_{q,q'} f^{X\cup\{r\}}_{q'} \end{aligned}$$

using $f_r^{X\cup\{r\}} = 0$ and $f_{q'}^{X\cup\{r\}} = f^X_{q'}$ if $q' \in Q \setminus \{r\}$. □

With Theorem 1, decidability of the equivalence problem for PFAs carries over to PREs (provided the probabilities in an expression are rational numbers), whereas their threshold problem is undecidable.

Corollary 1.

1. *The equivalence problem for* PREs *is decidable: given* PREs E *and* F, *does* $[\![E]\!] = [\![F]\!]$ *hold?*
2. *The threshold problem for* PREs *is undecidable: given an alphabet* A, *a* PRE E *over* A *and* $0 < s < 1$, *is there a word* $w \in A^+$ *such that* $[\![E]\!](w) \geq s$*?*

Note that PFAs cannot recognize all series recognized by usual Rabin automata, i.e., PFAs without the blocking assumption over accepting states. For example, the map $g\colon A^+ \to [0,1]$, defined by $g(a^n) = 1$ if $n > 0$ and $g(w) = 0$ for all other words w, is not recognizable by a PFA (note that a^*a is not a PRE). However, g is recognized by a Rabin automaton with a single state. To deal with this issue, we can add a fresh symbol $\triangleleft$ at the end of a word. For a function $f\colon A^+ \to [0,1]$, we define $f_\triangleleft\colon (A \cup \{\triangleleft\})^+ \to [0,1]$ by $f_\triangleleft(w\triangleleft) = f(w)$ if $w \in A^+$, and 0 otherwise. For example, the series $g_\triangleleft$ is defined by $a(a^*\triangleleft)$, which is a PRE since $a + \triangleleft \in \mathsf{PRE}$. More generally, we can prove the following:

Proposition 2. *Let* $f\colon A^+ \to [0,1]$. *The function* $f_\triangleleft$ *is recognizable by a* PFA *(or equivalently by a* PRE*) iff* f *is recognizable by a Rabin automaton.*

3 Adding Two-Way Navigation and Pebbles

In this section, we extend probabilistic automata and expressions such that they allow us to navigate in a given word and place pebbles that can be recovered later. Before we extend PFAs and PREs accordingly, let us give a motivating example.

Example 2. Using pebbles in probabilistic expressions or automata is a natural and powerful way to deal with nesting in LTL formulas. Indeed, temporal logics implicitly use a free variable to denote the position where a formula has to be evaluated. We will mark this position with a pebble, say x, in expressions $E_\varphi(x)$ or automata $\mathcal{A}_\varphi(x)$ associated with LTL formulas φ.

Consider an LTL formula $\mathsf{F}\varphi$, for *Finally* φ. Given a word w and a position i in w, we are interested in the probability $\mathbb{P}(\mathsf{F}\varphi, w, i)$ that φ holds in w at position i. For instance, for $\varphi = \frac{1}{3}a$, we should obtain $\mathbb{P}(\mathsf{F}\varphi, abba, 0) = \frac{1}{3} + \frac{2}{3}(0 + \frac{2}{3}(0 + \frac{2}{3}(\frac{1}{3} + 0)))$: either φ is satisfied immediately with probability $\frac{1}{3}$, or it is not (probability $\frac{2}{3}$) so that (product) it has to be satisfied later. More generally, we have

$$\begin{aligned}\mathbb{P}(\mathsf{F}\varphi, w, i) &= \mathbb{P}(\varphi, w, i) + \mathbb{P}(\neg\varphi, w, i) \times \mathbb{P}(\mathsf{F}\varphi, w, i+1)\\ &= \sum_{k \geq i} \mathbb{P}(\varphi, w, k) \times \prod_{i \leq j < k} \mathbb{P}(\neg\varphi, w, j)\,.\end{aligned}$$

For every LTL formula φ, we are aiming at an equivalent expression $E_\varphi(x)$ which evaluates to $\mathbb{P}(\varphi, w, i)$ over word w when pebble x marks position i. For this, we use a new construct, $y!E_\varphi(y)$, which marks the current position with pebble y, and computes φ on the whole word (from beginning to end) with this position marked (this is a non-progressing construct). Let us illustrate this inductive construction for LTL formulas. For *Finally* φ, we set

$$E_{\mathsf{F}\varphi}(x) = \rhd?{\to^*}x?\big(\big(y!E_{\neg\varphi}(y)\big){\to}\big)^*\big(y!E_\varphi(y)\big){\to^*}\lhd?\,.$$

The expression starts at the beginning of the word ($\rhd?$), and moves to the right ($\to^*$) until it discovers the marked position ($x?$). Then, for each $n \geq 0$, it iterates n times the computation of $\neg\varphi$ with the current position marked by y ($y!E_{\neg\varphi}(y)$), moving to the right ($\to$) between two computations. Finally, it computes φ with $y!E_\varphi(y)$ before moving to the last position of the word ($\to^*\lhd?$).

Similarly, for *Globally* φ ($\mathsf{G}\varphi$), we have $\mathbb{P}(\mathsf{G}\varphi, w, i) = \prod_{j \geq i} \mathbb{P}(\varphi, w, j)$, leading to the simpler expression

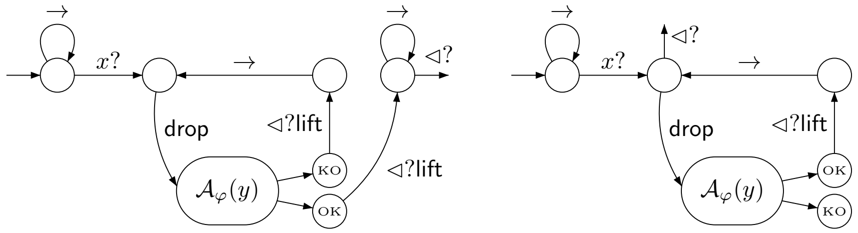
$$E_{\mathsf{G}\varphi}(x) = \rhd?{\to^*}x?\big(\big(y!E_\varphi(y)\big){\to}\big)^*\lhd?\,.$$

The last test ($\lhd?$) is useful to enforce the preceding star operation to capture the whole suffix of the word from the position marked by x.

Finally, based on the equivalence $\varphi \mathbin{\mathsf{U}} \psi \equiv (\neg\psi \wedge \varphi) \mathbin{\mathsf{U}} \psi$, the expression for the *Until* modality is

$$E_{\varphi\mathsf{U}\psi}(x) = \rhd?{\to^*}x?\big(\big(y!(E_{\neg\psi}(y){\leftarrow^*}E_\varphi(y))\big){\to}\big)^*\big(y!E_\psi(y)\big){\to^*}\lhd?\,.$$

In terms of automata, let us assume that, for every formula φ, there is an automaton $\mathcal{A}_\varphi$ with two designated terminal states, OK and KO, such that runs ending in OK (and at the end of the word) compute expression E_φ and those ending in KO compute expression $E_{\neg\varphi}$. These automata are 2-way, and can drop and lift pebbles on word positions. Dropping a pebble resets the control at the beginning. Lifting a pebble can be performed anywhere and resets the control to the position of the last dropped pebble (which then gets removed). The figure below depicts automata for the modalities *Finally* and *Globally*.



3.1 Probabilistic Pebble Automata

Our extended automata navigate (just like extended expressions defined below) inside a word in both directions, denoted → and ←. Motivated by Example 2, we moreover equip automata with $p \in \mathbb{N}$ pebbles, $1, \ldots, p$. Naturally, two-way automata can check if the current position carries some particular letter or pebble, or if it is a border (i.e., the first or last) position of the input word. Thus, they make their probabilistic choice depending on the type of the current position, which is a set containing the letter and pebbles that can be found at the current position. Formally, a *p-type* is a set $t \subseteq A \cup \{\triangleright, \triangleleft\} \cup \{1, \ldots, p\}$ such that (i) $\triangleleft \in t$ implies $t = \{\triangleleft\}$, and (ii) $\triangleleft \notin t$ implies $|t \cap A| = 1$. In other words, t indicates the current letter from A, unless the control is beyond the word ($\triangleleft \in t$). Moreover, it reveals if the current position is the first ($\triangleright \in t$), or the last one ($\triangleleft \in t$), or neither of them ($t \cap \{\triangleright, \triangleleft\} = \emptyset$). Let $\mathcal{T}_p$ be the set of p-types, and let $\mathcal{T} = \bigcup_{p \in \mathbb{N}} \mathcal{T}_p$ denote the set of *types.* The current state and type of a configuration will give rise to a probability function, which triggers a move of the automaton, taken from the set $\mathcal{M} = \{\rightarrow, \leftarrow, \mathsf{drop}, \mathsf{lift}, \mathsf{stay}\}$.

Let us formally define pebble probabilistic automata, which generalize PFAs.

Definition 2. *A* pebble probabilistic automaton (PPA) *over A is a tuple $\mathcal{A} = (p, Q, \iota, Acc, \mathbb{P})$ where $p \in \mathbb{N}$ is the number of pebbles, Q is a finite set of states, $\iota \in Q$ is the initial state, and $Acc \subseteq Q \setminus \{\iota\}$ is the set of final states. Moreover, $\mathbb{P}\colon Q \times \mathcal{T}_p \times \mathcal{M} \times Q \to [0, 1]$ is a transition probability function such that:*

- *For all $\delta = (q, t, d, q')$, if $\mathbb{P}(\delta) > 0$ then $q \notin Acc$ and $\triangleright \in t$ implies $d \neq \leftarrow$ and $\triangleleft \in t$ implies $d \notin \{\rightarrow, \mathsf{drop}\}$.*
- *For all $q \in Q \setminus Acc$ and all types $t \in \mathcal{T}_p$, we have $\sum_{(d,q') \in \mathcal{M} \times Q} \mathbb{P}(q, t, d, q') \leq 1$.*

Next, we define the behavior of PPAs. Fix a word $w = a_0 \cdots a_{n-1}$ and a PPA $\mathcal{A}$ with p pebbles. A *configuration* of $\mathcal{A}$ over w is a triple $\kappa = (q, i, \pi)$ with $q \in Q$, $i \in \mathrm{pos}(w) = \{0, \ldots, n\}$, and $\pi \in \{0, \ldots, n-1\}^{\leq p}$, where $X^{\leq p}$ is the set of words over X of length at most p. Intuitively, q is the current state, i is the current position, and π represents a stack recording the $k = |\pi| \leq p$ positions of the currently dropped pebbles: $\pi = i_1 \cdots i_k$ means that pebble $\ell \in \{1, \ldots, k\}$ is currently at position i_ℓ and pebbles $k+1, \ldots, p$ are currently not dropped. Pebbles are dropped and lifted using a stack policy: if $\pi = i_1 \cdots i_k$, only pebble $k+1$ can be dropped (provided $k+1 \leq p$), and only pebble k can be lifted (provided $k \geq 1$).

Let $\kappa = (q, i, \pi)$ be a configuration with $\pi = i_1 \cdots i_k$. We call κ *initial* if $q = \iota$, $i = 0$, and $\pi = \varepsilon$. It is said to be *final* if $q \in Acc$, $i = n$, and $\pi = \varepsilon$. Moreover, the type of κ is denoted by *type*(κ) and defined as $\{a_i \mid i < n\} \cup \{\triangleright \mid i = 0\} \cup \{\triangleleft \mid i = n\} \cup \big\{\ell \in \{1, \ldots, k\} \mid i_\ell = i\big\}$.

Given $\delta = (q, t, d, q') \in Q \times \mathcal{T}_p \times \mathcal{M} \times Q$ as well as configurations $\kappa = (q, i, \pi)$ and $\kappa' = (q', i', \pi')$, we write $\kappa \xrightarrow{\delta} \kappa'$ if $t = \mathit{type}(\kappa)$ and the following hold:

1. if $d = \rightarrow$ then $i' = i + 1$ and $\pi' = \pi$
2. if $d = \leftarrow$ then $i' = i - 1$ and $\pi' = \pi$
3. if $d = \mathsf{stay}$ then $i' = i$ and $\pi' = \pi$
4. if $d = \mathsf{drop}$ then $i' = 0$ and $\pi' = \pi i$ and $i < n$
5. if $d = \mathsf{lift}$ then $\pi' i' = \pi$.

In other words, drop saves the current position on the stack of pebbles ($\pi' = \pi i$ in 4) and resets the head to the first position ($i' = 0$ in 4). Moreover, $\pi' i' = \pi$ in 5 means that lift pops the last dropped pebble and resets the control to the position where this pebble was dropped.

A *run* of $\mathcal{A}$ over w is a finite sequence $\rho = (\kappa_0, \delta_1, \kappa_1, \ldots, \delta_h, \kappa_h)$ $(h \geq 0)$ of configurations and transitions such that, for all $0 < m \leq h$, we have $\mathbb{P}(\delta_m) > 0$ and $\kappa_{m-1} \xrightarrow{\delta_m} \kappa_m$. We say that ρ is *accepting* if κ_0 is an initial configuration and κ_h is final. The probability of this run is the product of the probabilities of the transitions all along the run: $\mathbb{P}(\rho) = \prod_{m=1}^{h} \mathbb{P}(\delta_m)$. Now, let $[\![\mathcal{A}]\!](w)$ be the sum $\sum_\rho \mathbb{P}(\rho)$, where ρ ranges over the accepting runs of $\mathcal{A}$ over w. Note that the number of accepting runs over a given word may be infinite so that, a priori, $[\![\mathcal{A}]\!]$ is a mapping $A^+ \to \mathbb{R}_{\geq 0}^{\infty}$. However, one can show the following:

Proposition 3. *For every* PPA $\mathcal{A}$ *and every* $w \in A^+$*, we have* $[\![\mathcal{A}]\!](w) \in [0, 1]$.

Proof. Let us define, for every $w \in A^+$, a probability space $(\Omega^w, \mathfrak{E}^w, \mathbb{P}^w)$. The set Ω^w of outcomes is the set of maximal runs of $\mathcal{A}$ over w, starting in the initial configuration (for technical reasons, we have to include infinite ones). Moreover, the set $\mathfrak{E}^w$ of events is the smallest σ-algebra containing, for all finite runs ρ, the *cylinder set* $\mathit{Cyl}(\rho) \overset{\text{def}}{=} \{\rho' \mid \rho' \text{ is a maximal run with prefix } \rho\}$. With this, there is a unique probability measure $\mathbb{P}^w$ for Ω^w and $\mathfrak{E}^w$, which is given by $\mathbb{P}^w(\mathit{Cyl}(\rho)) = 1 \cdot \mathbb{P}(\delta_1) \cdot \ldots \cdot \mathbb{P}(\delta_h)$ for all finite runs $\rho = (\kappa_0, \delta_1, \kappa_1, \ldots, \delta_h, \kappa_h)$. As the set $AR(w)$ of accepting runs over w is countable, it is measurable in the probability space, and its probability is $\mathbb{P}^w(AR(w)) = \sum_{\rho \in AR(w)} \mathbb{P}^w(\mathit{Cyl}(\rho))$. Note that the latter equals $[\![\mathcal{A}]\!](w)$ and that $\mathit{Cyl}(\rho) = \{\rho\}$ for all $\rho \in AR(w)$. □

The proof above establishes a strong connection between PPAs and Markov chains. This connection also provides an algorithm for evaluating a PPA wrt. a given word, which reduces to computing the probability of reaching a final configuration in the *synchronized* Markov chain (see, for example, [3]).

Example 3. Consider the PPA $\mathcal{A}$ (2-way, without pebbles) depicted on the left of Fig. 2 (where $0 < s < 1$) over alphabet $A = \{a\}$. The synchronized Markov chain of $\mathcal{A}$ wrt. a word of length n is depicted on the right of the same figure. This Markov chain represents a random walk over a straight line of bounded length.

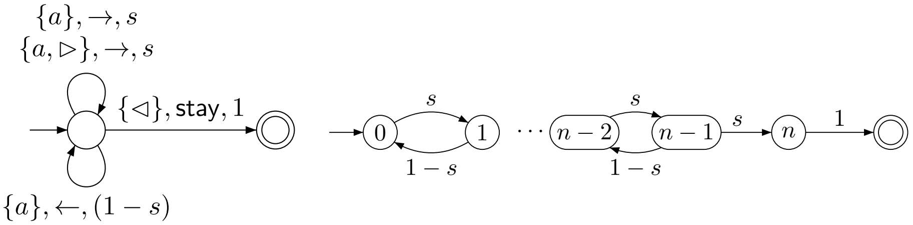

$\{a\}, \rightarrow, s$
$\{a, \triangleright\}, \rightarrow, s$
$\{\triangleleft\}$, stay, 1
$\{a\}, \leftarrow, (1-s)$
0
1
s
$1-s$
$\cdots$
$n-2$
$n-1$
s
$1-s$
s
n
1

Fig. 2. Markov chain obtained by synchronizing $\mathcal{A}$ with a word of length n

3.2 Probabilistic Pebble Expressions

Now, we define probabilistic expressions that capture the expressive power of PPAs. Just like PPAs, expressions are equipped with some pebbles from an infinite supply $\mathcal{P} = \{1, 2, \ldots\}$ (though every expression will only employ a finite number thereof). Also, expressions should be able to move left or right and to check if pebbles are dropped on the current position. Hence, we adopt an XPath-like syntax: we explicitly distinguish progression in the word (with $\leftarrow$ or $\rightarrow$) from tests $\varphi \in \mathsf{Tests}$ checking the current type. We define *test* formulas inductively by

$$\varphi ::= \ a? \mid x? \mid \triangleright? \mid \triangleleft? \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi$$

where $a \in A$ and $x \in \mathcal{P}$. In particular, formulas $\triangleright?$ and $\triangleleft?$ allow an expression to test whether it is at the first or last position of the word. Moreover, $x?$ is true if pebble x is on the current position. Formulas $\varphi \in \mathsf{Tests}$ are interpreted over a type $t \in \mathcal{T}$ in a straightforward manner: for $\theta \in A \cup \{\triangleright, \triangleleft\} \cup \mathcal{P}$, we write $t \models \theta?$ if $\theta \in t$. The semantics of boolean connectives is defined as expected.

We start with *unrestricted* weighted pebble expressions (WPEs) defined by

$$E ::= s \mid \varphi \mid \leftarrow \mid \rightarrow \mid E + E \mid E \cdot E \mid E^* \mid x!E$$

with $s \in \mathbb{R}^{\infty}_{\geq 0}$, $\varphi \in \mathsf{Tests}$ and $x \in \mathcal{P}$. The interpretation of s, sum, concatenation, and star does not change wrt. WEs. The new atomic expressions $\leftarrow$ and $\rightarrow$ perform a step to the left or to the right. The expressions φ and $x!E$, on the other hand, perform *non-progressing* computations (similar to s). The atomic WE a is obtained by checking first if the current letter is a and then moving right, hence the abbreviation $a \overset{\text{def}}{=} a?{\rightarrow}$. The new construct $x!E$ is read "compute E with x assigned to the current position". The computation of E "rescans" the whole word, from position 0 to position $|w|$.

The set of *free* pebbles of an expression $E \in \mathsf{WPE}$ is defined in the obvious way; in particular, we set $\mathit{Free}(x?) = \{x\}$ and $\mathit{Free}(x!E) = \mathit{Free}(E) \setminus \{x\}$.

The semantics $[\![E]\!]$ of an expression E assigns a value from $\mathbb{R}^{\infty}_{\geq 0}$ to each tuple (w, σ, i, j) where $w = a_0 a_1 \cdots \in A^+$, $i, j \in \mathrm{pos}(w)$ and $\sigma : \mathit{Free}(E) \rightarrow \{0, \ldots, |w| - 1\}$. It computes the weight of going from i to j with the pebble assignment σ. Formally, the semantics is given in Table 1. Hereby, we let $\mathit{type}(i) \in \mathcal{T}$ denote the set $\{a_i \mid i \neq |w|\} \cup \{\triangleright \mid i = 0\} \cup \{\triangleleft \mid i = |w|\} \cup \sigma^{-1}(i)$. Moreover, $\sigma[x \mapsto i]$ stands for the assignment coinciding with σ except on x, which is

$$
[\![s]\!](w,\sigma,i,j) = \begin{cases} s & \text{if } j = i \\ 0 & \text{otherwise} \end{cases} \qquad [\![\varphi]\!](w,\sigma,i,j) = \begin{cases} 1 & \text{if } j = i \text{ and } \mathit{type}(i) \models \varphi \\ 0 & \text{otherwise} \end{cases}
$$

$$
[\![\rightarrow]\!](w,\sigma,i,j) = \begin{cases} 1 & \text{if } j = i+1 \\ 0 & \text{otherwise} \end{cases} \qquad [\![\leftarrow]\!](w,\sigma,i,j) = \begin{cases} 1 & \text{if } j = i-1 \\ 0 & \text{otherwise} \end{cases}
$$

$$
[\![E_1 + E_2]\!] = [\![E_1]\!] + [\![E_2]\!] \qquad [\![E^*]\!](w,\sigma,i,j) = \textstyle\sum_{m\in\mathbb{N}} [\![E^m]\!](w,\sigma,i,j)
$$

$$
[\![E_1 \cdot E_2]\!](w,\sigma,i,j) = \sum_{k\in\mathrm{pos}(w)} [\![E_1]\!](w,\sigma,i,k) \times [\![E_2]\!](w,\sigma,k,j)
$$

$$
[\![x!E]\!](w,\sigma,i,j) = \begin{cases} [\![E]\!](w,\sigma[x\mapsto i],0,|w|) & \text{if } j = i < |w| \\ 0 & \text{otherwise} \end{cases}
$$

Table 1. Semantics of WPEs

mapped to i. If $\mathit{Free}(E) = \emptyset$, then we let $[\![E]\!](w) = [\![E]\!](w,0,|w|)$ (omitting σ, as it is irrelevant).

One can easily check that concatenation, also called Cauchy product, is associative, i.e., $[\![(E_1 \cdot E_2) \cdot E_3]\!] = [\![E_1 \cdot (E_2 \cdot E_3)]\!]$. Hence, one can define E^m by induction: $E^0 = 1$ and $E^{m+1} = E \cdot E^m$. Moreover, $+$ is associative and commutative, concatenation distributes over $+$, and $x!$ distributes over $+$.

As for expressions without pebbles, we need to restrict the sum and star operations to get the *probabilistic* fragment. Doing so, we lose commutativity, associativity and distributivity hence we enforce these properties explicitly. The proof of Proposition 1 cannot be extended to cope with $x!E$, hence we strengthen the rule for concatenation.

Definition 3. Probabilistic pebble expressions (PPEs) *is the fragment of* WPEs *built inductively as follows:*

(Atoms) $s \in [0,1]$, $\varphi \in$ Tests, $\leftarrow$ *and* $\rightarrow$ *are* PPEs.

($+_\varphi$) *If* E *and* F *are* PPEs *and* $\varphi \in$ Tests, *then* $\varphi \cdot E + (\neg\varphi) \cdot F$ *is a* PPE.

($+_s$) *If* E *and* F *are* PPEs *and* $s \in [0,1]$, *then* $s \cdot E + (1-s) \cdot F$ *is a* PPE.

($\cdot$) *If* $E + F$ *is a* PPE *and* G *is a* PPE, *then* $E + F \cdot G$ *is a* PPE.

($*$) *If* $E + F$ *is a* PPE, *then* $E^* \cdot F$ *is a* PPE.

($x!$) *If* E *is a* PPE, *then* $x!E$ *is a* PPE.

(ACD) PPEs *are closed under the following associativity, commutativity and distributivity rules (*ACD*-rules):*

$$
\begin{array}{lrcl}
\mathbf{A}_+ & E + (F + G) \in \mathsf{PPE} & \longleftrightarrow & (E + F) + G \in \mathsf{PPE} \\
\mathbf{C}_+ & E + F \in \mathsf{PPE} & \longleftrightarrow & F + E \in \mathsf{PPE} \\
\mathbf{A}_\cdot & E \cdot (F \cdot G) \in \mathsf{PPE} & \longleftrightarrow & (E \cdot F) \cdot G \in \mathsf{PPE} \\
\mathbf{D}_\cdot & E \cdot (F + G) \in \mathsf{PPE} & \longleftrightarrow & E \cdot F + E \cdot G \in \mathsf{PPE} \\
\mathbf{D}_\cdot & (E + F) \cdot G \in \mathsf{PPE} & \longleftrightarrow & E \cdot G + F \cdot G \in \mathsf{PPE} \\
\mathbf{D}_{x!} & x!(E + F) \in \mathsf{PPE} & \longleftrightarrow & x!E + x!F \in \mathsf{PPE}
\end{array}
$$

The semantics of PPEs is inherited from WPEs and we will show later that when $E \in \mathsf{PPE}$ then $[\![E]\!](w, \sigma, i, j)$ actually always belongs to $[0, 1]$.

Example 4. We continue Example 3. An equivalent PPE to denote the random walk is given by

$$E = (\neg{\lhd}?(s \rightarrow + \, (1-s)\neg{\rhd}?{\leftarrow}))^* \, {\lhd}? \, .$$

Moreover, let $F = \neg{\lhd}?(s \rightarrow + \, (1-s)\neg{\rhd}?{\leftarrow})$ so that $E = F^* \, {\lhd}?$. Notice that E is indeed an expression in PPE because $F + {\lhd}? \in \mathsf{PPE}$. Let w be a word of length $m \geq 2$. We can easily see that, for all $i, j \in \mathrm{pos}(w)$ and all $n \geq |j - i|$, the expression F^n computes a *positive* value on (w, i, j). Therefore, the expression F^* computes an infinite sum on (w, i, j). In the present case $(0 < s < 1)$, the series $\sum_{n \geq 0} [\![F^n]\!](w, i, j)$ converges: with $\alpha = \frac{1-s}{s}$, one can show that $[\![E]\!](w, 0, |w|) = 1/(1 + \alpha + \ldots + \alpha^{|w|})$, and $[\![F^*]\!](w, i, j) \in [0, 1]$.

We define now the multiset $\mathsf{Terms}(E)$ of terms of an expression $E \in \mathsf{WPE}$. This will be crucial for the translation from expressions to automata in the next section. Intuitively, if we suppose that summation is pushed up as much as possible by means of ACD-rules, then the multiset of terms consists of all expressions that occur in this big outermost sum. Formally, the definition is by induction over $E \in \mathsf{WPE}$. When E is an atom, we let $\mathsf{Terms}(E) = \{\!\{E\}\!\}$ be the singleton multiset containing only the atom itself. Moreover,

$$\begin{aligned}
\mathsf{Terms}(E + F) &= \mathsf{Terms}(E) \uplus \mathsf{Terms}(F) \\
\mathsf{Terms}(E \cdot F) &= \{\!\{E' \cdot F' \mid E' \in \mathsf{Terms}(E), F' \in \mathsf{Terms}(F)\}\!\} \\
\mathsf{Terms}(E^*) &= \{\!\{E^*\}\!\} \\
\mathsf{Terms}(x!E) &= \{\!\{x!E' \mid E' \in \mathsf{Terms}(E)\}\!\} \, .
\end{aligned}$$

Note that, if an expression F can be obtained from an expression E through ACD-rules, then we have $\mathsf{Terms}(E) = \mathsf{Terms}(F)$. The converse also holds as can be seen from the following proposition which can be easily proved by structural induction on the expression.

Proposition 4. *Let $E \in \mathsf{WPE}$ with $\mathsf{Terms}(E) = \{\!\{E_i \mid i \in I\}\!\}$. Using ACD-rules, we can rewrite E into $\sum_{i \in I} E_i$. In particular, we have $[\![E]\!] = \sum_{i \in I} [\![E_i]\!]$. Hence, we identify E and $\sum_{i \in I} E_i$.*

4 The Probabilistic Kleene Theorem

We prove in this section that PPAs are effectively equivalent to PPEs. We start with the construction of PPAs from PPEs. The problematic cases are concatenation and iteration due to the precondition $E + F \in \mathsf{PPE}$. To deal with these cases, we construct from PPE E a PPA $\mathcal{A}$ which simultaneously recognizes all *terms* of E.

Theorem 2. *From any expression $E \in \mathsf{PPE}$ we can effectively construct an equivalent PPA $\mathcal{A} = (p, Q, \iota, Acc, \mathbb{P})$. More precisely, if $\mathsf{Terms}(E) = \{\!\{E_i \mid i \in I\}\!\}$, the set of accepting states of $\mathcal{A}$ is $Acc = \{f_i \mid i \in I\}$ and for all $i \in I$ the expression E_i is equivalent to the PPA $\mathcal{A}[f_i] = (p, Q, \iota, \{f_i\}, \mathbb{P})$.*

Proof. The construction is by structural induction on the expression $E \in \mathsf{PPE}$ which may have free pebbles. As usual, the assignment of free pebbles will be encoded in the alphabet of the word read by the automaton $\mathcal{A} \in \mathsf{PPA}$, hence (w, σ) will be interpreted as a word over $A \times 2^{\mathit{Free}(E)}$ in the automata. Then, equivalence $\mathcal{A} \approx E$ means that, for all words $w \in A^+$, assignments $\sigma \colon \mathit{Free}(E) \to \mathrm{pos}(w) \setminus \{|w|\}$, and positions $i, j \in \mathrm{pos}(w)$, the value $[\![E]\!](w, \sigma, i, j)$ is the sum of the probabilities of the runs of $\mathcal{A}$ over (w, σ) starting in the initial state in position i, ending in an accepting state in position j.

The cases $s \in [0,1]$, $\varphi \in \mathsf{Tests}$, $\to$, and $\leftarrow$ are clear. For each atom, the resulting automaton has only two states (it uses stay transitions for s and φ).

Now let $E, E' \in \mathsf{PPE}$ be such that $\mathsf{Terms}(E) = \{\!\{E_i \mid i \in I\}\!\}$ and $\mathsf{Terms}(E') = \{\!\{E'_j \mid j \in J\}\!\}$. By induction hypothesis, we have constructed two suitable PPAs $\mathcal{A} = (p, Q, \iota, \mathit{Acc}, \mathbb{P})$ and $\mathcal{A}' = (p', Q', \iota', \mathit{Acc}', \mathbb{P}')$ with $\mathit{Acc} = \{f_i \mid i \in I\}$ and $\mathit{Acc}' = \{f'_j \mid j \in J\}$. Without loss of generality, we assume that $p = p'$, $Q \cap Q' = \emptyset$ and that a final state may only be reached when no pebble is dropped (if necessary, this may be enforced by keeping the number of dropped pebbles in the states).

Consider $E'' = \varphi \cdot E + \neg\varphi \cdot E'$. We have $\mathsf{Terms}(E'') = \{\!\{\varphi \cdot E_i \mid i \in I\}\!\} \uplus \{\!\{\neg\varphi \cdot E'_j \mid j \in J\}\!\}$. We construct a PPA $\mathcal{A}'' = (p, Q \uplus Q' \uplus \{\iota''\}, \iota'', \mathit{Acc} \uplus \mathit{Acc}', \mathbb{P}'')$. From the new initial state ι'', we add stay transitions with probability 1 going to ι for all types t satisfying φ, and going to ι' for all other types.

The construction for $E'' = s \cdot E + (1-s) \cdot E'$ is similar. We have $\mathsf{Terms}(E'') = \{\!\{s \cdot E_i \mid i \in I\}\!\} \uplus \{\!\{(1-s) \cdot E'_j \mid j \in J\}\!\}$. We add stay transitions with probability s from ι'' to ι and stay transitions with probability $1-s$ from ι to ι'.

For the concatenation, we assume that $E = F + G$ and $E'' = F + G \cdot E'$. We have $I = K \uplus L$ with $\mathsf{Terms}(F) = \{\!\{E_i \mid i \in K\}\!\}$ and $\mathsf{Terms}(G) = \{\!\{E_i \mid i \in L\}\!\}$. Hence, we have $\mathsf{Terms}(E'') = \{\!\{E_i \mid i \in K\}\!\} \uplus \{\!\{E_i \cdot E'_j \mid (i,j) \in L \times J\}\!\}$. The automaton $\mathcal{A}''$ for E'' consists of one copy of $\mathcal{A}$ and a copy $\mathcal{A}'_i$ of $\mathcal{A}'$ for every $i \in L$. First, $\mathcal{A}''$ simulates $\mathcal{A}$ until it reaches some final state f_i of $\mathcal{A}$. Then, if $i \in L$, a stay transition leads with probability 1 into the initial state of $\mathcal{A}'_i$. The final states of $\mathcal{A}''$ consist of $\{f_i \mid i \in K\}$ and a copy of Acc' for each $i \in L$.

For the Kleene star we assume that $E = F + G$ and $E'' = F^* \cdot G$. We have $I = K \uplus L$ with $\mathsf{Terms}(F) = \{\!\{E_i \mid i \in K\}\!\}$ and $\mathsf{Terms}(G) = \{\!\{E_i \mid i \in L\}\!\}$. Hence, we have $\mathsf{Terms}(E'') = \{\!\{F^* \cdot E_i \mid i \in L\}\!\}$. We construct the PPA $\mathcal{A}'' = (p, Q, \iota, \mathit{Acc}'', \mathbb{P}'')$ with $\mathit{Acc}'' = \{f_i \mid i \in L\}$ by adding stay transitions with probability 1 from all states f_i for $i \in K$ back to the initial state ι.

Consider $E'' = x!E$. We have $\mathsf{Terms}(E'') = \{\!\{x!E_i \mid i \in I\}\!\}$. We construct the PPA $\mathcal{A}'' = (p+1, Q \uplus \{\iota''\} \uplus \mathit{Acc}'', \iota'', \mathit{Acc}'', \mathbb{P}'')$ with Acc'' a copy of Acc. To this aim, we shift the numbers of pebbles of $\mathcal{A}$ to $\{2, \ldots, p+1\}$ keeping pebble 1 to be the fresh one, used to mark the position of variable x. We start by dropping pebble 1 with transitions of the form $(\iota'', t, \mathsf{drop}, \iota)$, each with probability 1. At the end of a computation of $\mathcal{A}$, pebble 1 is lifted, again with probability 1, using transitions $(q, \{\lhd\}, \mathsf{lift}, q'')$ where $q'' \in \mathit{Acc}''$ is the copy of $q \in \mathit{Acc}$.

Finally, if E'' is obtainted from E via ACD-rules, we have $\mathsf{Terms}(E'') = \mathsf{Terms}(E)$ so we can keep the same automaton: $\mathcal{A}'' = \mathcal{A}$. □

Note that, even if we start from a PRE, the proof above does not yield a PFA since it adds *stay* transitions. Hence, this proof has to be adapted to translate PREs into PFAs. By Proposition 3, the semantics of automata only assumes values in $[0,1]$. This carries over to PPE.

Corollary 2. *For every* PPE E *and every* $w \in A^+$*, we have* $[\![E]\!](w) \in [0,1]$.

We turn now to the construction of expressions (PPEs) which are equivalent to automata (PPAs). We follow usual procedures for the translation from automata to expressions, ensuring throughout the proof that we produce expressions in PPE. To this aim, we strongly rely on ACD-rules.

Theorem 3. *Let* $\mathcal{A} = (p, Q, \iota, Acc, \mathbb{P})$ *be a* PPA *with* p *pebbles. We can effectively construct a* PPE $E_\iota = \sum_{q \in Acc} E_{\iota,q}$ *such that* $[\![E_{\iota,q}]\!](w, 0, |w|)$ *is the sum of the runs from the initial configuration* $(\iota, 0, \varepsilon)$ *to the final configuration* $(q, |w|, \varepsilon)$*.*

5 Conclusion

In this paper, we presented a probabilistic Kleene Theorem, first for classical probabilistic automata and then for extended automata with two-way navigation and pebbles. This constitutes a first step towards probabilistic XPath, so we aim at extending our work to finite trees and probabilistic tree automata. We also raise the question of whether our technique can be used to obtain ω-expressions for probabilistic Büchi automata, which have attracted a lot of attention [2,1,8]. Just like classical finite automata, weighted automata over semirings enjoy characterizations in terms of monadic second-order logic [13,6]. Continuing this line of research, a recent paper establishes a logical characterization of probabilistic automata [30]. It would be interesting to study whether alternative characterizations exist that use, for example, a transitive-closure operator [6].

References

1. C. Baier, N. Bertrand, and M. Größer. On decision problems for probabilistic Büchi automata. In *Proc. of FoSSaCS'08*, volume 4962 of *LNCS*, pages 287–301. Springer, 2008.
2. C. Baier and M. Größer. Recognizing ω-regular languages with probabilistic automata. In *Proc. of LICS'05*, pages 137–146. IEEE Computer Society, 2005.
3. C. Baier and J.-P. Katoen. *Principles of model checking.* MIT Press, 2008.
4. J. Berstel and Ch. Reutenauer. *Noncommutative rational series with applications*, volume 137 of *Encyclopedia of Mathematics & Its Applications.* Cambridge, 2011.
5. M. Bojańczyk. Tree-walking automata. In *Proc. of LATA'08*, volume 5196 of *LNCS*, pages 1–17. Springer, 2008.
6. B. Bollig, P. Gastin, B. Monmege, and M. Zeitoun. Pebble weighted automata and transitive closure logics. In *Proc. of ICALP'10*, volume 6199 of *LNCS*, pages 587–598. Springer, 2010.
7. P. Buchholz and P. Kemper. Quantifying the dynamic behavior of process algebras. In *Process Algebra and Probabilistic Methods. Performance Modelling and Verification*, volume 2165 of *LNCS*, pages 184–199. Springer, 2001.

8. R. Chadha, A. P. Sistla, and M. Viswanathan. Power of randomization in automata on infinite strings. *Logical Methods in Computer Science*, 7(3:22), 2011.
9. C. Cortes, M. Mohri, and A. Rastogi. L_p distance and equivalence of probabilistic automata. *Int. J. Found. Comput. Sci.*, 18(4):761–779, 2007.
10. C. Cortes, M. Mohri, A. Rastogi, and M. Riley. On the computation of the relative entropy of probabilistic automata. *Int. J. Found. Comput. Sci.*, 19(1):219–242, 2008.
11. Y. Deng and C. Palamidessi. Axiomatizations for probabilistic finite-state behaviors. *Theor. Comput. Sci.*, 373(1-2):92–114, 2007.
12. Y. Deng, C. Palamidessi, and J. Pang. Compositional reasoning for probabilistic finite-state behaviors. In *Processes, Terms and Cycles: Steps on the Road to Infinity*, volume 3838 of *LNCS*, pages 309–337. Springer, 2005.
13. M. Droste and P. Gastin. Weighted automata and weighted logics. In M. Droste, W. Kuich, and H. Vogler, editors, *Handbook of Weighted Automata*, EATCS Monographs in Theoretical Computer Science, chapter 5, pages 175–211. Springer, 2009.
14. C. Dwork and L. Stockmeyer. On the power of 2-way probabilistic finite state automata. In *Proc. of FoCS'89*, pages 480–485. IEEE Computer Society, 1989.
15. S. Flesca, F. Furfaro, and S. Greco. Weighted path queries on semistructured databases. *Inform. and Comput.*, 204(5):679 – 696, 2006.
16. P. Gastin and B. Monmege. Adding pebbles to weighted automata. In *Proc. of CIAA'12*, LNCS. Springer, 2012. To appear.
17. H. Gimbert and Y. Oualhadj. Probabilistic automata on finite words: Decidable and undecidable problems. In *Proc. of ICALP'10*, volume 6199 of *LNCS*, pages 527–538. Springer, 2010.
18. S. Kiefer, A. S. Murawski, J. Ouaknine, B. Wachter, and J. Worrell. On the complexity of the equivalence problem for probabilistic automata. In *Proc. of FoSSaCS'12*, volume 7213 of *LNCS*, pages 467–481. Springer, 2012.
19. M. Marx. Conditional XPath. *ACM Transactions on Database Systems*, 30(4):929–959, 2005.
20. A. Paz. *Introduction to probabilistic automata (Computer science and applied mathematics)*. Academic Press, 1971.
21. M. O. Rabin. Probabilistic automata. *Inform. and Control*, 6:230–245, 1963.
22. B. Ravikumar. On some variations of two-way probabilistic finite automata models. *Theor. Comput. Sci.*, 376:127–136, 2007.
23. J. Sakarovitch. Rational and recognizable power series. In M. Droste, W. Kuich, and H. Vogler, editors, *Handbook of Weighted Automata*, EATCS Monographs in Theoretical Computer Science, chapter 4, pages 103–172. Springer, 2009.
24. M. P. Schützenberger. On the definition of a family of automata. *Inform. and Control*, 4:245–270, 1961.
25. R. Segala. Probability and nondeterminism in operational models of concurrency. In *Proc. of CONCUR'06*, volume 4137 of *LNCS*, pages 64–78. Springer, 2006.
26. A. Silva, F. Bonchi, M. M. Bonsangue, and J. Rutten. Quantitative Kleene coalgebras. *Inf. Comput.*, 209(5):822–849, 2011.
27. B. ten Cate and L. Segoufin. XPath, transitive closure logic, and nested tree walking automata. In *Proc. of PODS'08*, pages 251–260. ACM, 2008.
28. W.-G. Tzeng. A polynomial-time algorithm for the equivalence of probabilistic automata. *SIAM J. Comput.*, 21(2):216–227, 1992.
29. R. J. van Glabbeek, S. A. Smolka, and B. Steffen. Reactive, generative and stratified models of probabilistic processes. *Inform. and Comput.*, 121(1):59–80, 1995.
30. Th. Weidner. Probabilistic automata and probabilistic logic. In *Proc. of MFCS'12*, LNCS. Springer, 2012. To appear.

Appendix

Proposition 1. *Let $E \in \mathsf{PRE}$ with $\mathsf{Terms}(E) = \{\!\{E_i \mid i \in I\}\!\}$. Let $F_i \in \mathsf{PRE}$ for each $i \in I$. We have*

$$\overline{E} = \sum_{i \in I} E_i \cdot F_i \in \mathsf{PRE}\,.$$

In particular, any sum of (some) terms of a PRE is also a PRE: if $I' \subseteq I$ we get $\sum_{i \in I'} E_i \in \mathsf{PRE}$ by letting $F_i = 1$ for $i \in I'$ and $F_i = 0$ otherwise.

Proof (of Proposition 1). We proceed by structural induction over E. This is trivial for atoms and also for ACD-rules since they preserve the multiset of terms. Let $G, H \in \mathsf{PRE}$ with $\mathsf{Terms}(G) = \{\!\{G_j \mid j \in J\}\!\}$ and $\mathsf{Terms}(H) = \{\!\{H_k \mid k \in K\}\!\}$.

Consider $E = sG + (1-s)H$. Then, $\mathsf{Terms}(E) = \{\!\{sG_j \mid j \in J\}\!\} \uplus \{\!\{(1-s)H_k \mid k \in K\}\!\}$. Hence we have $I = J \uplus K$, $E_i = sG_i$ for $i \in J$ and $E_i = (1-s)H_i$ for $i \in K$. By induction, we have $\overline{G} = \sum_{i \in J} G_i \cdot F_i \in \mathsf{PRE}$ and $\overline{H} = \sum_{i \in K} H_i \cdot F_i \in \mathsf{PRE}$. Therefore, $s\overline{G} + (1-s)\overline{H} \in \mathsf{PRE}$. This expression can be rewritten into $\overline{E}$ with ACD-rules.

The proof is similar for $E = \sum_{a \in A} aE_a$.

Consider $E = G \cdot H$ so that $I = J \times K$ and $E_{(j,k)} = G_j \cdot H_k$ for $(j,k) \in I$. By induction, for every $j \in J$, we get $\overline{H}_j = \sum_{k \in K} H_k \cdot F_{(j,k)} \in \mathsf{PRE}$. Again by induction we obtain $\sum_{j \in J} G_j \cdot \overline{H}_j \in \mathsf{PRE}$. This expression can be rewritten into $\overline{E}$ with ACD-rules.

Consider now $E = G^* \cdot H$ assuming $G + H \in \mathsf{PRE}$ instead of $G, H \in \mathsf{PRE}$. We have $I = K$ and $E_i = G^* \cdot H_i$ for $i \in I$. Applying the induction hypothesis to the expression $G + H$ with $F_j = 1$ for $j \in J$, we obtain $\sum_{j \in J} G_j + (\sum_{k \in K} H_k \cdot F_k) \in \mathsf{PRE}$. From the star rule we deduce that $(\sum_{j \in J} G_j)^* \cdot (\sum_{k \in K} H_k \cdot F_k) \in \mathsf{PRE}$. Using ACD-rules, this can be rewritten into $\overline{E} = \sum_{i \in I} G^* \cdot H_i \cdot F_i$. □

From Automata to Expressions.

Let $\mathcal{A} = (p, Q, \iota, Acc, \mathbb{P})$ be a PPA with p pebbles. Without loss of generality, we assume that the number of dropped pebbles is encoded in the states of $\mathcal{A}$. Hence, $Q = \biguplus_{k=0}^{p} Q^{(k)}$ where a state in $Q^{(k)}$ means that $p-k$ pebbles have been dropped and k pebbles may still be dropped. The top level $Q^{(p)}$ of the automaton contains the initial state $\iota \in Q^{(p)}$ and the final states $Acc \subseteq Q^{(p)}$. Moreover, a transition of level k may only test pebbles up to $p-k$: for $q \in Q^{(k)}$, if $\mathbb{P}(q, t, d, q') > 0$ then $t \in \mathcal{T}_{p-k}$.

We also assume that lift transitions are enabled only at the last position of the input word, i.e., only when the type is $t = \{\lhd\}$. For each state $q \in Q^{(k)}$ $(k < p)$ we let $f_q = \sum_{q' \in Q^{(k+1)}} \mathbb{P}(q, \{\lhd\}, \mathsf{lift}, q')$ be the sum of the probabilities of lift transitions starting from q. When $q \in Q^{(p)}$, we let $f_q = 1$ if $q \in Acc$ and $f_q = 0$ otherwise.

For each $q \in Q^{(k)}$ $(k \leq p)$ we construct a PPE $E_q = \sum_{q' \in Q^{(k)}} E_{q,q'} \lhd ? f_{q'}$ where $E_{q,q'}$ is a PPE which computes the sum of the probabilities of runs starting from state q, ending in state q' and visiting only states in $Q^{(\leq k)} = \biguplus_{\ell \leq k} Q^{(\ell)}$.

This construction is by induction on $k \in \{0, \dots, p\}$. Within each level $k \leq p$, we follow usual procedures to translate automata into expressions.

When $k < p$ some pebbles were already dropped hence the PPE E_q will have free pebbles $\mathit{Free}(E_q) \subseteq \{1, \dots, p-k\}$ and we encode with an assignment $\sigma\colon \mathit{Free}(E_q) \to \mathrm{pos}(w) \setminus \{|w|\}$ the positions of the dropped pebbles. We denote by $\pi_\sigma \in \mathrm{pos}(w)^{p-k}$ the corresponding stack recording the positions of pebbles.

For $q \in Q^{(k)}$ and $X \subseteq Q^{(k)}$ we let $f_q^X = \begin{cases} \lhd ? f_q & \text{if } q \in X \\ 1 & \text{otherwise.} \end{cases}$

For $q \in Q^{(k)}$ and $X \subseteq Q^{(k)}$, we construct by induction on X a PPE

$$E_q^X = \lhd ? f_q + \sum_{q' \in Q^{(k)}} E_{q,q'}^X f_{q'}^X$$

where $E_{q,q'}^X$ is a PPE such that $[\![E_{q,q'}^X]\!](w, \sigma, i, j)$ is the sum of the probabilities of *nonempty* runs starting from configuration (q, i, π_σ), ending in configuration (q', j, π_σ) and using only *intermediary* states in $X \cup Q^{(\leq k-1)}$.

The base of the induction is when $k = 0$ and $X = \emptyset$. For each state $q \in Q^{(0)}$ and type $t \in \mathcal{T}_p$, by definition of PPAs we have $\sum_{(d,q') \in \mathcal{M} \times Q} \mathbb{P}(q, t, d, q') \leq 1$. Now, since $q \in Q^{(0)}$, drop moves are disallowed. Moreover, we have assumed that lift moves are only enabled for type $t = \{\lhd\}$. Hence, when $t \neq \{\lhd\}$, using rule $(+_s)$ we deduce that

$$F_{q,t} = \sum_{q' \in Q^{(0)}} \mathbb{P}(q, t, \mathsf{stay}, q') + \mathbb{P}(q, t, \leftarrow, q') \cdot \leftarrow + \mathbb{P}(q, t, \rightarrow, q') \cdot \rightarrow$$

is a PPE. For $t = \{\lhd\}$, using rule $(+_s)$ and the definition of f_q we deduce that

$$F_{q,\lhd} = f_q + \sum_{q' \in Q^{(0)}} \mathbb{P}(q, \{\lhd\}, \mathsf{stay}, q') + \mathbb{P}(q, \{\lhd\}, \leftarrow, q') \cdot \leftarrow$$

is a PPE. Now, using the deterministic sum $(+_\varphi)$ we obtain the PPE

$$E_q^\emptyset = \sum_{t \in \mathcal{T}_p} t? \cdot F_{q,t} = \lhd ? f_q + \sum_{q' \in Q^{(0)}} E_{q,q'}^\emptyset f_{q'}^\emptyset$$

where the last equality is obtained using ACD-rules and $f_{q'}^\emptyset = 1$.

We turn now to the induction step on X. This step is the same for all layers $k \leq p$. So fix $k \leq p$, $X \subseteq Q^{(k)}$ and $r \in Q^{(k)} \setminus X$. By induction, we assume that PPEs E_q^X have been constructed for all $q \in Q^{(k)}$. We construct $E_q^{X \cup \{r\}}$.

We have $E_r^X = \lhd ? f_r + \sum_{q' \in Q^{(k)}} E_{r,q'}^X f_{q'}^X \in \mathsf{PPE}$ and $f_r^X = 1$ since $r \notin X$. With rule $(*)$ we get

$$G_r^X = \left(E_{r,r}^X\right)^* \cdot \left(\lhd ? f_r + \sum_{q' \in Q^{(k)} \setminus \{r\}} E_{r,q'}^X f_{q'}^X\right) \in \mathsf{PPE}\,.$$

Now, $E^X_q = \lhd?f_q + \sum_{q' \in Q^{(k)}} E^X_{q,q'} f^X_{q'} \in \mathsf{PPE}$ and $f^X_r = 1$. Using rule $(\cdot)$ we can plug G^X_r after $E^X_{q,r}$ and we obtain the PPE

$$\begin{aligned}
E^{X\cup\{r\}}_q &= \lhd?f_q + E^X_{q,r} \cdot G^X_r + \sum_{q' \in Q^{(k)} \setminus \{r\}} E^X_{q,q'} f^X_{q'} \\
&= \lhd?f_q + E^X_{q,r} \big(E^X_{r,r}\big)^* \lhd?f_r + \sum_{q' \in Q^{(k)} \setminus \{r\}} \big(E^X_{q,q'} + E^X_{q,r} \big(E^X_{r,r}\big)^* E^X_{r,q'}\big) f^X_{q'} \\
&= \lhd?f_q + \sum_{q' \in Q^{(k)}} E^{X\cup\{r\}}_{q,q'} f^{X\cup\{r\}}_{q'}
\end{aligned}$$

where the last equality uses $f^{X\cup\{r\}}_r = \lhd?f_r$ and $f^{X\cup\{r\}}_{q'} = f^X_{q'}$ if $q' \in Q^{(k)} \setminus \{r\}$.

It remains to deal with $X = \emptyset$ and $0 < k \leq p$. Compared to the case $X = \emptyset$ and $k = 0$ we also have to consider drop transitions which lead to the lower layer. The next pebble to be dropped is $x = p - k + 1$. By induction, we have already constructed expressions $E^{Q^{(k-1)}}_r$ for all states $r \in Q^{(k-1)}$ of the lower layer. To lighten the notation, for $r, r' \in Q^{(k-1)}$, we let $E_r = E^{Q^{(k-1)}}_r$ and $E_{r,r'} = E^{Q^{(k-1)}}_{r,r'}$. Using distributivity $\mathbf{D}_{x!}$ we get the PPE

$$x!E_r = x!(\lhd?f_r) + \sum_{r' \in Q^{(k-1)}} x!(E_{r,r'} \lhd?f_{r'})\,.$$

The first term is equivalent to 0 since we consider only nonempty words. Using the concatenation rule $(\cdot)$ we plug after each term $x!(E_{r,r'} \lhd?f_{r'})$ the PPE $1 = \sum_{q' \in Q^{(k)}} \frac{\mathbb{P}(r', \{\lhd\}, \mathsf{lift}, q')}{f_{r'}}$ and we obtain using ACD-rules the PPE

$$G_r = \sum_{q' \in Q^{(k)}} \sum_{r' \in Q^{(k-1)}} x!(E_{r,r'} \lhd?f_{r'}) \frac{\mathbb{P}(r', \{\lhd\}, \mathsf{lift}, q')}{f_{r'}}\,.$$

Note that $\mathit{Free}(G_r) \subseteq \{1, \ldots, p-k\}$. Now, we proceed as before. For each state $q \in Q^{(k)}$ and type $t \in \mathcal{T}_{p-k}$ we have $\sum_{(d,q') \in \mathcal{M} \times Q} \mathbb{P}(q, t, d, q') \leq 1$. Hence, when $t \neq \{\lhd\}$, using rule $(+_s)$ we deduce that

$$\begin{aligned}
F_{q,t} &= \sum_{q' \in Q^{(k)}} \mathbb{P}(q, t, \mathsf{stay}, q') + \mathbb{P}(q, t, \leftarrow, q') \leftarrow + \mathbb{P}(q, t, \rightarrow, q') \rightarrow + \sum_{r \in Q^{(k-1)}} \mathbb{P}(q, t, \mathsf{drop}, r) G_r \\
&= \sum_{q' \in Q^{(k)}} \Bigg(\mathbb{P}(q, t, \mathsf{stay}, q') + \mathbb{P}(q, t, \leftarrow, q') \leftarrow + \mathbb{P}(q, t, \rightarrow, q') \rightarrow \\
&\qquad\qquad + \sum_{r, r' \in Q^{(k-1)}} \mathbb{P}(q, t, \mathsf{drop}, r) x!(E_{r,r'} \lhd?f_{r'}) \frac{\mathbb{P}(r', \{\lhd\}, \mathsf{lift}, q')}{f_{r'}} \Bigg)
\end{aligned}$$

is a PPE. For $t = \{\lhd\}$ the drop transition is not possible, hence we get as before the PPE

$$F_{q,\lhd} = f_q + \sum_{q' \in Q^{(k)}} \mathbb{P}(q, \{\lhd\}, \mathsf{stay}, q') + \mathbb{P}(q, \{\lhd\}, \leftarrow, q') \leftarrow .$$

Now, using the deterministic sum $(+_\varphi)$ we obain the PPE

$$E_q^\emptyset = \sum_{t \in \mathcal{T}_{p-k}} t? \cdot F_{q,t} = \lhd ? f_q + \sum_{q' \in Q^{(k)}} E_{q,q'}^\emptyset f_{q'}^\emptyset$$

where the last equality is obtained using ACD-rules and $f_{q'}^\emptyset = 1$. Note that $\mathit{Free}(E_q^\emptyset) \subseteq \{1, \ldots, p-k\}$.

For $0 < k \leq p$, the induction on X goes exactly as explained above. Again, for $q, q' \in Q^{(k)}$, we let $E_q = E_q^{Q^{(k)}}$ and $E_{q,q'} = E_{q,q'}^{Q^{(k)}}$. Therefore, for all $k \leq p$ and $q \in Q^{(k)}$ we have constructed a PPE $E_q = \sum_{q' \in Q^{(k)}} E_{q,q'} \lhd ? f_{q'}$ as announced at the beginning of the proof.

Finally, the PPE E_ι is equivalent to the automaton $\mathcal{A}$: for each word w, the value $[\![E_\iota]\!](w, 0, |w|)$ is the sum of the probabilities of all *accepting* runs starting from the initial configuration $(\iota, 0, \varepsilon)$ and ending in a final configuration $(q', |w|, \varepsilon)$ with $q' \in \mathit{Acc}$.