Probabilistic Input-Driven Pushdown Automata

Alex Rose
Department of Mathematics and Computer Science, Saint Petersburg State University, Russia

Alexander Okhotin
Department of Mathematics and Computer Science, Saint Petersburg State University, Russia

Abstract

A probabilistic variant of input-driven pushdown automata (IDPDA), also known as visibly pushdown automata, is introduced. It is proved that these automata can be determinized: an n-state probabilistic IDPDA that accepts each string with probability at least $\lambda + \delta$ or at most $\lambda - \delta$ is transformed to a deterministic IDPDA with at most $(1+\frac{1}{\delta})^{n^2-n}$ states recognizing the same language. An asymptotically close lower bound is provided: for infinitely many n, there is a probabilistic IDPDA with $4n+1$ states and $\delta = \frac{1}{270n}$, such that every equivalent deterministic IDPDA needs at least $7^{n^2/14}$ states. A few special cases of automata with reduced determinization complexity are identified.

2012 ACM Subject Classification Theory of computation → Probabilistic computation; Theory of computation → Formal languages and automata theory

Keywords and phrases Finite automata, probabilistic automata, input-driven automata, visibly pushdown automata, state complexity

Digital Object Identifier 10.4230/LIPIcs.MFCS.2023.78

Funding This work was supported by the Russian Science Foundation, project 23-11-00133.

1 Introduction

In probabilistic models of computation, there are several options for every step of the computation, each with a specified probability. Probabilistic finite automata (PFA) were introduced by Rabin [23] in 1963. Assuming the two-sided bounded-error condition, that is, that every string is accepted either with probability at most $\lambda - \delta$, or with probability at least $\lambda + \delta$, for some λ and δ with $0 < \lambda - \delta < \lambda + \delta < 1$, Rabin proved that every such automaton with n states can be transformed to a deterministic automaton (DFA) with at most $(1 + \frac{1}{\delta})^{n-1}$ states. The lower bound on the determinization complexity has been refined several times: in 1982, Freivalds [11] proved that in the worst case $2^{\Omega(\sqrt{n})}$ states are necessary, in 1996, Ambainis [4] improved this lower bound to $\Omega(2^{n\frac{\log\log n}{\log n}})$. Finally, in 2008, Freivalds [12] established the first exponential lower bound of the order $7^{n/14}$, improved to $2^{n/4}$ under Artin's conjecture from number theory.

A probabilistic version of pushdown automata (PDA) was studied by Freivalds [10], who proved that they can recognize a language not recognized by any nondeterministic PDA. Later, Hromkovič and Schnitger [13] showed that these two models are incomparable in power, whereas probabilistic PDA with one-sided error are weaker than both models, yet stronger than deterministic PDA.

The concept of *input-driven pushdown automata* (IDPDA) was introduced by Mehlhorn [16] in 1980. This is a special case of pushdown automata, in which the operations performed on the stack are determined by the input symbols. Nondeterministic IDPDA were first defined by von Braunmühl and Verbeek [25], who proved that they can be determinized, with an n-state nondeterministic automaton transformed to a deterministic one with 2^{n^2} states and $O(2^{n^2})$ stack symbols. In 2004, Alur and Madhusudan [2] have reintroduced the model under the name of *visibly pushdown automata*, and obtained some important new results: in particular, they proved that determinization requires $2^{\Omega(n^2)}$ states in the worst case, established the closure of the family under all basic language operations, and determined the computational complexity of the inclusion and universality problems for IDPDA. Their paper motivated further research in this area. In particular, several related models were investigated, such as alternating [7], unambiguous [21] and timed input-driven pushdown automata [24, 8, 19], input-driven pushdown automata on infinite strings [3, 15, 22]. This research has also inspired some further models, such as pushdown automata driven by finite transducers [14].

In this paper we introduce and study the probabilistic version of IDPDA, with two-sided bounded-error condition, similar to the classical PFA [23] and probabilistic PDA [10]. Even though input-driven pushdown automata with randomized transitions appeared in several papers on automata on infinite strings and their verification [5, 9, 26], the cited papers were not concerned with language recognition, making their models much different from what is studied in this paper.

The first result of this paper is that probabilistic IDPDA with bounded error define the same class of languages as deterministic IDPDA. To be precise, if a language is recognized by bounded-error n-state PIDPDA with δ-cutpoint, then the same language is recognized by a DIDPDA with $(1+\frac{1}{\delta})^{n^2-n}$ states and const $\cdot\, (1+\frac{1}{\delta})^{n^2-n}$ stack symbols. This is done via considering transition matrices on well-nested strings, and proving that if two matrices are close enough with respect to a certain norm, then they are equivalent under an equivalence relation defined by Alur et al. [1]. This in turn implies that the language is recognizable by an IDPDA.

In Section 5 we also give a lower bound. To do so we essentially find a way to lift lower bounds for PFAs to lower bounds for PIDPDAs. Then we apply the best known lower bound for PFAs – the one due to Freivalds [12].

Finally, in Section 6 we study the special case of automata operating on strings nesting depth one. Three cases of automata with reduced determinization complexity are identified. First, there is an analog of *unary* languages, in which every substring enclosed in brackets is unary, for which determinization requires fewer states than in the general case: only $O((1+\frac{1}{\delta})^n)$. We also prove that the same bound holds for automata with $\Sigma_0 = \{a, b\}$ if the transitions by a and b *commute*. The latter condition means that the automaton "counts" the number of as and bs. Lastly, we show that if the transitions by every symbol in Σ_0 are *deterministic*, then the upper bound can also be significantly reduced to n^n, under no restrictions on the size of Σ_0.

2 Deterministic Input-driven Pushdown Automata

A *deterministic input-driven pushdown automaton* (DIDPDA) [3, 16] is a special case of a deterministic pushdown automaton, in which the input alphabet Σ is split into three disjoint sets of *left brackets* Σ_{+1}, *right brackets* Σ_{-1} and *neutral symbols* Σ_0. The type of the input symbol determines the type of the operation with the stack: on a left bracket from Σ_{+1}, then the automaton always pushes one symbol onto the stack; on a right bracket from Σ_{-1}, the automaton must pop one symbol; finally, on a neutral symbol in Σ_0, the automaton may not use the stack. In this paper, symbols from Σ_{+1} and Σ_{-1} are denoted by left and right angled brackets, respectively ($<$, $>$), whereas lower-case Latin letters from the beginning of the alphabet $(a, b, c, \dots)$ are used for symbols from Σ_0.

▶ **Definition 1.** *A deterministic input-driven pushdown automaton (DIDPDA) is a 6-tuple* $A = (\Sigma, Q, \Gamma, q_0, [\delta_a]_{a\in\Sigma}, F)$*, where*

- $\Sigma = \Sigma_{+1} \cup \Sigma_0 \cup \Sigma_{-1}$ *is an input alphabet split into three disjoint classes;*

- *Q is a finite set of states of the automaton, with an initial state $q_0 \in Q$ and with a subset of accepting states $F \subseteq Q$;*
- *Γ is a finite set of stack symbols,*
- *the transition function by each left bracket symbol $< \in \Sigma_{+1}$ is a function $\delta_< \colon Q \to Q \times \Gamma$, which, for a given current state, provides the next state and the symbol to be pushed onto the stack;*
- *for every right bracket symbol $> \in \Sigma_{-1}$, a function $\delta_> \colon Q \times \Gamma \to Q$ specifies the next state, assuming that the given symbol is popped from the stack;*
- *for a neutral symbol $c \in \Sigma_0$, a function $\delta_c \colon Q \to Q$ provides the next state.*

For a string $w = a_1 \dots a_\ell$, with $a_1, \dots, a_\ell \in \Sigma$, the computation on w starting in a state r_0 with stack contents $\gamma_0 \in \Gamma^*$ is the sequence $\{(r_i, \gamma_i)\}_{i=0}^{\ell} \in Q \times \Gamma^*$, defined as follows.

- *If $a_i \in \Sigma_{+1}$, then $\delta_{a_i}(r_{i-1}) = (r_i, s)$ and $\gamma_i = \gamma_{i-1}s$,*
- *If $a_i \in \Sigma_{-1}$, then $\gamma_{i-1} = \gamma_i s$, for some $s \in \Gamma$, and $r_i = \delta_{a_i}(r_{i-1}, s)$.*
- *If $a_i \in \Sigma_0$, then $\gamma_i = \gamma_{i-1}$ and $r_i = \delta_{a_i}(r_{i-1})$.*

If w is well-nested, then this computation is always defined, and ends in the configuration (r_ℓ, γ_ℓ), with $\gamma_\ell = \gamma_0$.

A well-nested string w is accepted if the computation on w starting in q_0 with the empty stack ends in a configuration (q, ε) with $q \in F$. The set of all accepted strings is denoted by $L(A)$.

One can notice that each word w defines a function $f^w \colon Q \to Q$, such that if A is in state q, it will be in state $f^w(q)$ after reading w. Then $L(A) = \{\, w \mid w \text{ is well-nested and } f^w(q_0) \in F \,\}$.

For every language L, we define the following equivalence relation on the set of all well-nested strings.

▶ **Definition 2.** *Let L be a set of well-nested strings. Let w_1 and w_2 be well-nested. The relation $\approx_L$ on the set of well-nested strings is defined by $w_1 \approx_L w_2$ if, for every two strings u, v with uv well-nested, the string uw_1v is in L if and only if uw_2v is in L.*

If the language L is understood from the context, the relation $\approx_L$ is denoted by $\approx$.

▶ **Lemma 3** (Alur et al. [1])**.** *L is recognized by an IDPDA if and only if there is only a finite number of the equivalence classes with respect to this equivalence relation.*

Moreover, DIDPDA can be chosen such that it has m states and $|\Sigma_{+1}| \cdot m$ stack symbols, where m is the number of the equivalence classes. Also, if L is recognized by an IDPDA with n states, then the number of equivalence classes is not more than n^n.

3 Probabilistic Input-driven Automata

Unlike DIDPDAs, probabilistic input-driven pushdown automata may have multiple available transitions, with a probability of making each of them. Nevertheless, whether the automaton pushes, pops or leaves the stack intact, is still determined by the current input symbol. In the end, the input string is accepted if and only if the probability of reaching an accepting state after reading the string is sufficiently large.

▶ **Definition 4.** *Let S be a countable or a finite set. Let $D(S)$ denote the set of probability distributions on S, that is, $D(S) = \{\, p \colon S \to [0,1] \mid \sum_{x \in S} p(x) = 1 \,\}$.*

▶ **Definition 5.** *A probabilistic input-driven pushdown automaton (PIDPDA) is a 6-tuple $A = (\Sigma, Q, \Gamma, q_0, [\delta_a]_{a \in \Sigma}, F)$, where*

- $\Sigma = \Sigma_{+1} \sqcup \Sigma_0 \sqcup \Sigma_{-1}$ *is an input alphabet split into three disjoint classes;*
- Q *is a finite set of states of the automaton, with an initial state* $q_0 \in D(Q)$ *and with a subset of accepting states* $F \subseteq Q$*;*
- Γ *is a finite set of stack symbols,*
- *the transition function by each left bracket symbol* $< \in \Sigma_{+1}$ *is a function* $\delta_<\colon Q \to D(Q \times \Gamma)$*, which, for a given current state* $p \in Q$*, assigns a probability to each pair* (q, s)*, that is, the probability of pushing* s *onto the stack and entering the state* q*;*
- *for every right bracket symbol* $> \in \Sigma_{-1}$*, a function* $\delta_>\colon Q \times \Gamma \to D(Q)$ *specifies the probabilities of entering each state, assuming that the given symbol is popped from the stack;*
- *for a neutral symbol* $c \in \Sigma_0$*, a function* $\delta_c\colon Q \to D(Q)$ *provides the probabilities of the next state.*

For a string $w = a_1 \dots a_\ell$*, with* $a_1, \dots, a_\ell \in \Sigma$*, a computation sequence on* w *is a sequence* $\{(r_i, \gamma_i)\}_{i=0}^{\ell} \in Q \times \Gamma^*$*, which satisfies the following conditions.*
- *if* $a_i \in \Sigma_{+1}$*, then* $\gamma_i = \gamma_{i-1}s$*, for some* $s \in \Gamma$*, and the probability of this step is defined by* $p_i = \delta_{a_i}(r_{i-1})(r_i, s)$*,*
- *if* $a_i \in \Sigma_{-1}$*, then* $\gamma_{i-1} = \gamma_i s$*, for some* $s \in \Gamma$*, and the probability of this step is* $p_i = \delta_{a_i}(r_{i-1}, s)(r_i)$*.*
- *if* $a_i \in \Sigma_0$*, then* $\gamma_i = \gamma_{i-1}$ *and the probability is* $\delta_{a_i}(r_{i-1})(r_i)$*.*

The probability of such a sequence is the product $p_1 \cdot \ldots \cdot p_\ell$ *of probabilities of individual steps.*

The probability of going from configuration (q, γ) *to configuration* (q', γ') *by reading a string* w *is the sum of probabilities of all computation sequences on* w *that start with* (q, γ) *and end with* (q', γ')*.*

The probability of accepting a well-nested string w *is the probability of going from its initial configuration to any accepting configuration.*

$$\Pr(A \text{ accepts } w) = \sum_{q \in F} \Pr(A \text{ goes from } (q_0, \varepsilon) \text{ to } (q, \varepsilon) \text{ by reading } w)$$

▶ **Definition 6.** *A probabilistic input-driven pushdown automaton* A *is said to have a* δ*-cut-point* λ*, with* $\delta > 0$ *and* $\lambda \in [0, 1]$*, if, for every well-nested* w*, either* $\Pr(A \text{ accepts } w) \geqslant \lambda + \delta$ *or* $\Pr(A \text{ accepts } w) \leqslant \lambda - \delta$*.*

▶ **Definition 7.** *The language* $L(A)$ *recognized by an automaton* A *with a* δ*-cut-point* λ *is the set of all well-nested strings* w *for which* $\Pr(A \text{ accepts } w) \geqslant \lambda + \delta$*.*

Similarly to the deterministic case, each well-nested word w defines a stochastic matrix P^w of order $|Q| \times |Q|$, where $P^w_{q,r} = \Pr(A \text{ goes into } q \text{ from } r \text{ after reading } w)$. This is a generalization of functions f^w for deterministic IDPDAs.

4 Determinization and Upper Bound

▶ **Theorem 8.** *Let* A *be a probabilistic IDPDA with a* δ*-cut-point* λ *and* n *states. Then there exists a deterministic IDPDA that recognizes the same language and has at most* $(1 + \frac{1}{\delta})^{n^2-n}$ *states and at most* $|\Sigma_{+1}| \cdot (1 + \frac{1}{\delta})^{n^2-n}$ *stack symbols.*

To compare, Rabin's [23] transformation of a PFA to a DFA uses only $(1 + \frac{1}{\delta})^{n-1}$ states. Rabin's argument estimates the number of Myhill–Nerode equivalence classes, which is sufficient to describe the computation of a finite automaton. The computations of input-driven pushdown automata are harder to simulate, and require more involved equivalence classes of Alur et al. ($\approx$, see Definition 2).

The main idea of the proof is that if, for two well-nested strings, w and w', the corresponding stochastic matrices P^w and $P^{w'}$ are close under a certain metric, then they must be equivalent in the sense of Definition 2, that is, replacing any substring w with w' does not change the acceptance status of a string.

We shall use the following lemma which provides an upper bound on the maximum number of stochastic matrices at least 2δ apart in the metric given by the norm $||\xi|| = \max_{1 \leqslant k \leqslant n} \sum_{\ell=1}^{n} |\xi_{k\ell}|$. The proof uses Rabin's idea involving volumes.

▶ **Lemma 9.** *Let $\xi^{(1)}, \ldots, \xi^{(m)}$ be stochastic matrices of order n, such that $||\xi^{(\ell)} - \xi^{(k)}|| \geqslant 2\delta$ for $\ell \neq k$. Then $m \leqslant (1 + \frac{1}{\delta})^{n^2 - n}$.*

Proof. We view $n \times n$ matrices as points in $\mathbb{R}^{n^2}$. Let $M_n(\mathbb{R}_{\geqslant 0})$ denote the set of such matrices with non-negative elements. For $r > 0$ let us define

$$S(r) = \left\{ t \in M_n(\mathbb{R}_{\geqslant 0}) \;\middle|\; \sum_{1 \leqslant j \leqslant n} t_{ij} = r \text{ for } 1 \leqslant i \leqslant n \right\}$$

In particular, it follows from the definition that $S(1)$ is the set of stochastic matrices.

▷ Claim 10. If $r < \delta$, then $\xi^{(k)} + S(r)$ and $\xi^{(\ell)} + S(r)$ are disjoint for $k \neq \ell$.

Proof. Suppose $(\xi^{(k)} + S(r)) \cap (\xi^{(\ell)} + S(r)) \neq \varnothing$. Then there exist $x, y \in S(r)$ such that $\xi^{(k)} + x = \xi^{(\ell)} + y$. By definition, the norm of every element in $S(r)$ is equal to r, therefore $||\xi^{(\ell)} - \xi^{(k)}|| = ||x - y|| \leqslant ||x|| + ||y|| = 2r < 2\delta$, which contradicts the assumption that $||\xi^{(\ell)} - \xi^{(k)}|| \geqslant 2\delta$. ◁

Clearly, $\xi^{(1)}, \ldots, \xi^{(m)} \in S(1)$ because they are stochastic. Thus, $\xi^{(1)} + S(r), \ldots, \xi^{(m)} + S(r) \subseteq S(1) + S(r) = S(1 + r)$, where the latter equality follows from the definition of $S(r)$. By the claim, the sets $\xi^{(1)} + S(r), \ldots, \xi^{(m)} + S(r)$ are pairwise disjoint, and all of them are contained in $S(r + 1)$.

Now the plan is to use volumes of the sets $\xi^{(k)} + S(r)$ to prove that only a limited number of such sets may fit into $S(1 + r)$. Since the n^2-dimensional volume of $S(r)$ is 0, the first step is determine the right dimension. Let d be the dimension of $S(r)$, which is the same as the dimension of $S(1)$, since these sets are the same up to scaling. It is claimed that $d = n^2 - n$. The set $S(1)$ is contained in the $(n^2 - n)$-dimensional (affine) subspace H defined by the equations $t_{i1} + \ldots + t_{in} = 1$, $i = 1, \ldots, n$, so $d \leqslant n^2 - n$. On the other hand, $S(1)$ contains a $(n^2 - n)$-dimensional ball of small radius, confirming that $d = n^2 - n$.

Let V_d denote the d-dimensional volume. The sum of the volumes of the disjoint sets $\xi^{(k)} + S(r)$ does not exceed the volume of the set $S(r + 1)$ they are contained in.

$$V_d(S(r+1)) \geqslant V_d((\xi^{(1)} + S(r)) \cup \cdots \cup (\xi^{(m)} + S(r))) = V_d(\xi^{(1)} + S(r)) + \cdots + V_d(\xi^{(m)} + S(r))$$

Notice that $V_d(\xi^{(1)} + S(r)) = V_d(S(r))$ because $\xi^{(k)} + S(r)$ is a translation of $S(r)$. Hence, the last inequality yields $mV_d(S(r)) \leqslant V_d(S(r + 1))$.

The linear transformation $t \mapsto \frac{r+1}{r}t$ maps $S(r)$ onto $S(r + 1)$ (because if $\sum_{1 \leqslant j \leqslant n} t_{ij} = r$, then $\sum_{1 \leqslant j \leqslant n} \frac{r+1}{r} t_{ij} = \frac{r+1}{r} r = r + 1$), therefore $V_d(S(r + 1)) = (\frac{r+1}{r})^d V_d(S(r))$. But we have already established that $V_d(S(r + 1)) \geqslant mV_d(S(r))$, thus $(1 + \frac{1}{r})^d V_d(S(r)) \geqslant mV_d(S(r))$ and therefore $m \leqslant (1 + \frac{1}{r})^d$ for $0 < r < \delta$. Passing to the limit as r tends to δ, we obtain $m \leqslant (1 + \frac{1}{\delta})^d = (1 + \frac{1}{\delta})^{n^2 - n}$. ◀

The next lemma provides a connection between the matrices P^w and the equivalence classes.

▶ **Lemma 11.** *If* $||P^{w_1} - P^{w_2}|| < 2\delta$ *then* $w_1 \approx w_2$.

Proof. Consider any well-nested strings w and uv. Note that because w is well-nested, after reading uw the stack will be exactly the same as after reading u. Let h be the nesting depth of u.

Then $\Pr(A \text{ accepts } uwv)$ is expressed as the following sum over all possible stack contents after reading u and states before and after reading w.

$$\sum_{\gamma \in \Gamma^h} \sum_{\substack{1 \leqslant i \leqslant n \\ 1 \leqslant j \leqslant n}} \Pr(A \text{ goes to } (i, \gamma) \text{ after reading } u) \cdot P^w_{ij} \cdot \Pr(A \text{ accepts from } (j, \gamma) \text{ after reading } v)$$

For brevity, denote the probabilities of parts of this computation by

$$\begin{aligned} q_i(\gamma) &:= \Pr(A \text{ goes into } (i, \gamma) \text{ after reading } u), \\ r_j(\gamma) &:= \Pr(A \text{ accepts from } (j, \gamma) \text{ after reading } v), \end{aligned}$$

so that the above probability is expressed as follows.

$$\Pr(A \text{ accepts } uwv) = \sum_{\gamma \in \Gamma^h} \sum_{1 \leqslant i,j \leqslant n} q_i(\gamma) \cdot P^w_{ij} \cdot r_j(\gamma)$$

Then the probability $q_i(\gamma)$ depends only on i, γ and u; the probability $r_j(\gamma)$ depends only on j, γ and v; and $\sum_{1 \leqslant i \leqslant n} \sum_{\gamma \in \Gamma^h} q_i(\gamma) = 1$.

Therefore, the difference between the probabilities of accepting uw_1v and uw_2v is estimated as

$$\begin{aligned} \big|\Pr(A \text{ accepts } uw_1v) - \Pr(A \text{ accepts } uw_2v)\big| &= \left| \sum_{\gamma \in \Gamma^*} \sum_{1 \leqslant i,j \leqslant n} q_i(\gamma)(P^{w_1}_{ij} - P^{w_2}_{ij}) r_j(\gamma) \right| \leqslant \\ &\leqslant \sum_{\gamma \in \Gamma^*} \sum_{1 \leqslant i,j \leqslant n} q_i(\gamma) \cdot |P^{w_1}_{ij} - P^{w_2}_{ij}| \cdot r_j(\gamma) \leqslant \\ &\leqslant \sum_{\gamma \in \Gamma^*} \sum_{1 \leqslant i,j \leqslant n} q_i(\gamma) \cdot |P^{w_1}_{ij} - P^{w_2}_{ij}| = \\ &= \sum_{1 \leqslant i \leqslant n} \sum_{\gamma \in \Gamma^*} q_i(\gamma) \sum_{1 \leqslant j \leqslant n} |P^{w_1}_{ij} - P^{w_2}_{ij}| \leqslant \\ &\leqslant \left(\sum_{1 \leqslant i \leqslant n} \sum_{\gamma \in \Gamma^*} q_i(\gamma) \right) \cdot \max_{1 \leqslant i \leqslant n} \sum_{1 \leqslant j \leqslant n} |P^{w_1}_{ij} - P^{w_2}_{ij}| = \\ &= \max_{1 \leqslant i \leqslant n} \sum_{1 \leqslant j \leqslant n} |P^{w_1}_{ij} - P^{w_2}_{ij}| < 2\delta \end{aligned}$$

Since every well-nested string is accepted with the probability of either at least $\lambda + \delta$ or at most $\lambda - \delta$, w_1 and w_2 are accepted or rejected simultaneously, which proves the claim. ◀

Now we are ready to prove the theorem.

Proof of Theorem 8. The proof is by bounding the number of equivalence classes under $\approx$. Suppose there are at least m equivalence classes, then we can take $w_1, \ldots, w_m$ to be the representatives of these classes. This yields m points $P^{w_1}, \ldots, P^{w_m}$ in $[0,1]^{n^2}$. Due to Lemma 11, for $k \neq \ell$, the inequality $||P^{w_\ell}_{ij} - P^{w_k}_{ij}|| \geqslant 2\delta$ holds because $w_k \not\approx w_\ell$. Then, Lemma 9 implies $m \leqslant (1 + \frac{1}{\delta})^{n^2 - n}$. By Lemma 3, there is a DIDPDA with m states and $|\Sigma_{+1}| \cdot m$ stack symbols accepting the same language. ◀

5 Lower bounds

The first exponential lower bounds on the complexity of determinizing an n-state PFA were constructed by Freivalds [12]. His lower bound exists in two versions, both of the form c^n, but with different values of c: one bound holds unconditionally, whereas the other, with a greater base c, relies on Artin's conjecture from number theory. The second bound also uses PFAs with a smaller error probability.

▶ **Theorem 12** (Freivalds [12]). *For infinitely many numbers n, there exists a PFA with n states with a δ_0-cutpoint $\lambda_0 = \frac{1}{2}$, such that any equivalent DFA needs at least c^n states, where $c = 7^{\frac{1}{14}}$ and $\delta_0 = \frac{1}{270}$. If Artin's conjecture is true, then the estimate holds for $c = 2^{\frac{1}{4}}, \delta_0 = \frac{1}{36}$.*

We will use this result to construct a lower bound in our setting.

▶ **Theorem 13.** *For infinitely many numbers n, there exists a PIDPDA with $4n + 1$ states and a $\delta(n) = \frac{\delta_0}{n}$-cutpoint $\lambda(n) = \frac{1}{2n}$, such that every equivalent DIDPDA needs at least c^{n^2} states, where $c = 7^{\frac{1}{14}}$ and $\delta_0 = \frac{1}{270}$. If Artin's conjecture is true, then the estimate holds for $c = 2^{\frac{1}{4}}, \delta_0 = \frac{1}{36}$.*

Proof. For infinitely many numbers n, Freivalds constructed a language K_n such that:

- it can be recognized by a PFA A with a δ_0-cutpoint $\lambda_0 = \frac{1}{2}$ and n states,
- any DFA recognizing K_n requires at least c^n states.

The latter means that there exists a set $\{ u_i \mid 1 \leqslant i \leqslant \lceil c^n \rceil \}$ of at least c^n strings such that for every two strings from this set, u_{i_1} and u_{i_2} with $i_1 \neq i_2$, there exists a separating string v with one of the concatenations $u_{i_1}v$, $u_{i_2}v$ in K_n and the other not in K_n. Let $\{ v_j \mid 1 \leqslant j \leqslant m \}$ be a finite set of such separating strings, so that for all i_1 and i_2 with $i_1 \neq i_2$ there exists $j \in \{1, \dots, m\}$ with $u_{i_1}v_j \in K_n$ if and only if $u_{i_2}v_j \notin K_n$.

In the PFA defined by Freivalds, let Q be its set of states, and consider the probability distribution on the set of states after reading each string u_i from the initial state, as well as the probability of accepting each string v_j from each state, and denote them by the following vectors $p_i \in \mathbb{R}^n$ and $r_j \in \mathbb{R}^n$.

$$(p_i)_q = \Pr(A \text{ goes into } q \text{ after reading } u_i)$$
$$(r_j)_q = \Pr(A \text{ accepts from } q \text{ after reading } v_j)$$

Then the probability of accepting each concatenation u_iv_j is a scalar product $\langle p_i, r_j \rangle$, and since this is a bounded-error PFA with δ_0-cutpoint $\lambda_0 = \frac{1}{2}$, the following two properties must hold.

(i) For every i, j either $\langle p_i, r_j \rangle \geqslant \lambda_0 + \delta_0$ or $\langle p_i, r_j \rangle \leqslant \lambda_0 - \delta_0$.

(ii) For every $i_1 \neq i_2$ there exists j such that $\langle p_{i_1}, r_j \rangle \geqslant \lambda_0 + \delta_0$ and $\langle p_{i_2}, r_j \rangle \leqslant \lambda_0 - \delta_0$ (or $\langle p_{i_2}, r_j \rangle \geqslant \lambda_0 + \delta_0$ and $\langle p_{i_1}, r_j \rangle \leqslant \lambda_0 - \delta_0$).

Now these strings u_i and v_j, along with the PFA, are used to construct the desired probabilistic input-driven automaton. It is constructed over an alphabet with a single left bracket,

$$\Sigma_{+1} = \{<\},$$

with a large set of neutral symbols each encoding an n-tuple of strings u_i,

$$\Sigma_0 = \{ a_{i_1,\dots,i_n} \mid 1 \leqslant i_1, \dots, i_n \leqslant \lceil c^n \rceil \},$$

and with right brackets representing separating strings applied to one particular component of an n-tuple.

$$\Sigma_{-1} = \{ >_{k,j} \mid k \in \{1, 2, \dots, n\};\ 0 \leqslant j \leqslant m \}$$

Define the new language as $L_n = \{ <a_{i_1,\dots,i_n}>_{k,j} \mid u_{i_k} v_j \in K_n \}$. Then, in order to test the membership of a concatenation $u_{i_k} v_j$ in K_n, a deterministic IDPDA will have to remember the entire n-tuple of strings, whereas a probabilistic automaton can randomly choose k in the beginning, and then simulate Freivalds' automaton on the k-th component of the tuple.

▷ Claim 14. There exists a PIDPDA B with $4n + 1$ states and a $\delta(n)$-cutpoint $\lambda(n)$ recognizing L_n.

Proof. An n-state PIDPDA, which assumes three-symbol input strings of the form $<a_{i_1,\dots,i_n}>_{k,j}$, is constructed first; later it will be extended to check the form of the string.

The automaton operating on well-formed strings uses the same set of states Q as Freivalds' automaton. Assume that the states are numbers: $Q = \{1, \dots, n\}$, and let n be the only accepting state. The same set Q is also used as the stack alphabet.

In the initial state, the automaton reads the left bracket $<$ and equiprobably chooses the next state s and pushes s onto the stack. Next, it encounters a symbol $a_{i_1,\dots,i_n}$ in the state s, and simply replicates the probability distribution of Freivalds' automaton on the string u_{i_s}.

$$\delta'_{a_{i_1,\dots,i_n}}(s)(t) = (p_{i_s})_t$$

Finally, upon reading a right bracket $>_{k,j}$ in a state t, the automaton should decide whether to enter the accepting state n. It pops the number s from the stack, and if it does not match k, the automaton rejects (by entering n with probability 0). If s equals k, then the automaton accepts with the same probability $(r_j)_t$, with which Freivalds' automaton accepts the string v_j from the state t.

$$\delta'_{>_{k,j}}(t, s)(n) = \begin{cases} 0, & \text{if } k \neq s \\ (r_j)_t, & \text{if } k = s \end{cases}$$

$$\delta'_{>_{k,j}}(t, s)(1) = 1 - \delta'_{>_{k,j}}(t, s)(n)$$

$$\delta'_{>_{k,j}}(t, s)(\ell) = 0 \qquad (2 \leqslant \ell \leqslant n - 1)$$

It is claimed that this automaton accepts a string $<a_{i_1,\dots i_n}>_{k,j}$ with probability $\frac{1}{n}\langle p_{i_k}, r_j \rangle$. Indeed, the randomly chosen number s matches k with probability $\frac{1}{n}$, and, provided that it happened, the probability of acceptance is

$$\sum_{q \in Q} \Big(\Pr(B \text{ goes from } k \text{ to } q \text{ after reading } a_{i_1,\dots,i_n}) \cdot$$

$$\cdot \Pr(B \text{ accepts from } q \text{ upon reading } >_{k,j} \text{ with } k \text{ in the stack}) \Big) =$$

$$= \sum_{q \in Q} (p_{i_k})_q \cdot (r_j)_q = \langle p_{i_k}, q_j \rangle$$

By (ii), the scalar product $\langle p_{i_k}, q_j \rangle$ is either at least $\lambda_0 + \delta_0$ (if $u_{i_k} v_j \in K_n$), or at most $\lambda_0 - \delta_0$ (if $u_{i_k} v_j \notin K_n$). Therefore, the overall probability is either at least $\frac{\lambda_0}{n} + \frac{\delta_0}{n}$, or at most $\frac{\lambda_0}{n} - \frac{\delta_0}{n}$.

The construction above works for words in $<\Sigma_0\Sigma_{-1}$, other words may violate the δ-cutpoint condition. To eliminate them we can take a 4-state partial DFA that verifies that the input string is indeed of the form $<\Sigma_0\Sigma_{-1}$, and take the direct product of the automaton above and this DFA. One extra dead state is added for ill-formed strings. ◁

▷ Claim 15. Any DIDPDA recognizing L_n has at least $|\Sigma_0| \geqslant c^{n^2}$ states.

Proof. Since all well-formed strings begin with the same left bracket $<$, a deterministic automaton cannot store any information on the stack: it always pushes the same stack symbol. Consider the state of the DIDPDA after reading $<a_{i_1,\ldots,i_n}$. it is claimed that this state must be different for different symbols.

Suppose the contrary, that for some two symbols $a_{i_1,\ldots,i_n} \neq a_{i'_1,\ldots,i'_n}$, the state after reading them is the same state q. Let k be such that $i_k \neq i'_k$, and let v_j be the separating string for u_{i_k} and $u_{i'_k}$ with respect to Freivalds' automaton, that is, exactly one of the strings $u_{i_k}v_j$, $u_{i'_k}v_j$ is in K_n. Then, by the definition of L_n, exactly one of the strings $<a_{i_1,\ldots,i_n}>_{k,j}$ and $<a_{i'_1,\ldots,i'_n}>_{k,j}$ is in L_n. However, since the IDPDA is in the same configuration before reading $>_{k,j}$ on either string, it either accepts both strings or rejects both of them. The contradiction obtained shows that the number of states in the DIDPDA is at least $|\Sigma_0|$. ◁

The theorem follows from the two claims. ◀

The upper bound on the size of the constructed automaton gives $(1+\frac{1}{\delta(n)})^{(4n+1)^2-(4n+1)} = (1+\frac{n}{\delta_0})^{16n^2+4n} \leqslant (\frac{n+1}{\delta_0})^{16n^2+4n} = c^{(16n^2+4n)\log_c \frac{n+1}{\delta_0}} = c^{(16n^2+4n)(\log_c(n+1)-\log_c \delta_0)} = c^{(16+o(1))n^2 \log_c n} = c^{(16\log_c 2+o(1))n^2 \log_2 n}$. We see that the exponent differs from the one in the lower bound by an $O(\log n)$ factor.

6 Sharper Upper Bounds in Special Cases

For probabilistic finite automata, the case of a unary alphabet is much different from the general case. The first lower bound on the determinization complexity in the unary case was given by Freivalds [11]. Milani and Pighizzini [18] proved that the worst-case determinization blowup in the unary case is of the order of Landau's function, that is, $e^{(1+o(1))\sqrt{n \ln n}}$. Later Mereghetti et al. [17] and Bianchi et al. [6] investigated more details of the complexity of unary PFAs.

For input-driven pushdown automata, there is no unary case in the strict sense: as long as there is a pair of matching brackets, one can use them to encode any alphabet. In order to obtain a variant of the unary case, the use of brackets should be somehow restricted. The following condition of *nesting depth one* still allows encoding a binary alphabet by abusing the brackets, but this can be done only outside the brackets; if $|\Sigma_0| = 1$, then inside the brackets the string is truly unary.

▶ **Definition 16.** *A well-nested language L is called a depth-one language if for every $w \in L$ the maximal nesting depth of w is one, i.e., $L \subseteq \Sigma_0^*(\Sigma_{+1}\Sigma_0^*\Sigma_{-1}\Sigma_0^*)^*$.*

For depth-one languages, it is natural to consider the classical Myhill–Nerode relation operating on the outer level of brackets and restricted to well-nested strings.

▶ **Definition 17.** *For a language of well-nested strings L, define a relation $\sim_L$ on the set of well-nested strings by $u \sim_L u'$ if and only if, for every well-nested string v, the string uv is in L if and only if $u'v$ is in L. When the language L is clear from the context, the relation $\sim_L$ shall be denoted by $\sim$.*

Denote by $[u]$ the equivalence class of a string u under $\sim$.

Later it will be shown that if a depth-one language uses unary strings inside the brackets, then the determinization complexity is reduced. More generally, assume that the string inside each pair of brackets belongs to a regular set $S \subseteq \Sigma_0^*$.

▶ **Definition 18.** *Let $S \subseteq \Sigma_0^*$ be a regular prefix-closed language, that is, if $w_1w_2 \in S$, then $w_1 \in S$. A depth-one language L is called S-nice if, for every string $x \in L$ and for every partition $x = u{<}w{>}v$, with $w \in \Sigma_0^*$, the string w is in S.*

For an S-nice language L, consider the equivalence relation from Definition 2, defined by $w \approx w'$ if and only if, for all x, y with xy well-nested, $xwy \in L$ if and only if $xw'y \in L$. The following notation is introduced for equivalence classes restricted to elements of S.

▶ **Definition 19.** *For $u \in S$, let $[[u]]_S \subseteq S$ be the set of all strings in S equivalent to u under the relation $\approx$.*

For S-nice languages, there is an automaton of size proportional to the number of these equivalence classes (cf. Lemma 3 for the general case).

▶ **Lemma 20.** *Let L be an S-nice language, and let n_1 be the number of equivalence classes under $\sim$, and let n_2 be the number of equivalence classes under $\approx$ restricted to S. Then there exists a DIDPDA recognizing L with $n_1 + O(n_2)$ states and $n_1 \cdot |\Sigma_{+1}|$ stack symbols.*

Equivalence classes under $\sim$ become states used outside brackets. Whenever a bracket is encountered, these states are pushed onto the stack along with the bracket. Equivalence classes under $\approx$ are used as states inside brackets. The details are omitted for brevity.

Thus, in order to obtain upper bounds on the size of DIDPDA recognizing various languages of restricted form, it is sufficient to estimate the number of equivalence classes under both relations $\sim$ and $\approx$.

The first observation is that the number of Myhill–Nerode classes on the outer level of brackets can be estimated using Rabin's [23] argument for finite automata.

▶ **Lemma 21.** *Let a language L be recognized by a PIDPDA A with n states and a δ-cutpoint λ. Then L has at most $(1 + \frac{1}{\delta})^{n-1}$ equivalence classes under $\sim$.*

Sketch of a proof. Rabin's [23, Thm. 3] argument works, because it never refers to actual transitions of a probabilistic automaton, and uses only probabilities of computations over prefixes and suffixes. If these prefixes and suffixes contain any brackets, this does not affect the argument.

For well-nested u and v we introduce the vectors $p(u), r(v) \in \mathbb{R}^n$ with

$$p(u)_i = \Pr(A \text{ goes into } q_i \text{ after reading } u),$$
$$r(v)_i = \Pr(A \text{ accepts from } q_i \text{ after reading } v).$$

Then the probability that A accepts uv is equal to

$$\sum_{i=1}^{n} \Pr(A \text{ goes into } q_i \text{ after reading } u) \cdot \Pr(A \text{ accepts from } q_i \text{ after reading } v) = \langle p(u), r(v) \rangle.$$

Again, for well-nested u and u' it turns out that $\sum_{i=1}^{n} |p(u)_i - p(u')_i| < 2\delta$ implies $u \sim u'$, because for any well-nested v the following inequality holds.

$$|\Pr(A \text{ accepts } uv) - \Pr(A \text{ accepts } u'v)| = |\langle p(u), r(v)\rangle - \langle p(u'), r(v)\rangle| =$$
$$= |\langle p(u) - p(u'), r(v)\rangle| = \Big|\sum_{i=1}^{n} (p(u)_i - p(u')_i) r(v)_i\Big| \leq$$
$$\leq \sum_{i=1}^{n} |p(u)_i - p(u')_i| \cdot r(v)_i \leq \sum_{i=1}^{n} |p(u)_i - p(u')_i| < 2\delta$$

Therefore, pairwise inequivalent strings $u_1, \dots, u_m$ yield vectors $p(u_1), \dots, p(u_m)$ with $\sum_{i=1}^{n} |p(u_k)_i - p(u_l)_i| > 2\delta$, and, as Rabin showed, in this case $m \leq (1 + \frac{1}{\delta})^m$. ◀

In some special cases of languages, the number of equivalence classes under $\approx$ restricted to S is fairly small, leading to upper bounds on the size of DIDPDA. The first such case is when the probabilistic automaton behaves deterministically inside the brackets.

▶ **Theorem 22.** *Let A be a PIDPDA with n states and a δ-cutpoint λ that recognizes a depth-one language, and assume that all transitions by symbols in Σ_0 are deterministic. Then there is an equivalent DIDPDA with $O(n^n)$ states.*

Proof. By Lemma 21, the number of equivalence classes under $\sim$ for L is at most $(1+\frac{1}{\delta})^{n-1}$.

The language is S-nice with $S = \Sigma_0^*$. Recall that if $P^{w_1} = P^{w_2}$, the words w_1 and w_2 are equivalent under $\approx$. Since all matrices P^w for $w \in S = \Sigma_0^*$ are deterministic, they correspond to functions from Q to Q and, hence, the number of such matrices is not greater than n^n. Thus, there are at most n^n equivalence classes under $\approx$ restricted to S.

Finally, by Lemma 20, there is a DIDPDA with $O\big((1+\frac{1}{\delta})^{n-1} + n^n\big) = O(n^n)$ states. ◀

The theorem implies that it is impossible to achieve the $\Omega(c^{n^2})$ lower bound with a depth-one language using probabilistic transitions only for the brackets.

The special case of automata in Theorem 22 had the transition matrices inside the brackets generate a finite set of size n^n. In other special cases of automata with reduced determinization complexity, defined below, transition matrices inside the brackets generate infinite subspaces, yet the *dimension* of those subspaces is bounded. The following lemma allows a small DIDPDA to be constructed in such cases.

▶ **Lemma 23.** *Let L be an S-nice language recognized by a PIDPDA with n states. Let W be the subspace of $M_n(\mathbb{R})$ generated by $\{\, P^w \mid w \in S \,\}$. Then there is a DIDPDA with $O((1+\frac{1}{\delta})^{\max\{n-1, \dim W\}})$ states recognizing L.*

The proof of Lemma 23 relies on the following geometric property.

▶ **Lemma 24.** *Let $\xi^{(1)}, \ldots, \xi^{(m)}$ be stochastic matrices of order n, such that $||\xi^{(k)} - \xi^{(\ell)}|| \geqslant 2\delta$ for $k \neq \ell$. Assume that there exists a linear subspace $W \leqslant M_n(\mathbb{R})$, such that for every i the matrix $\xi^{(i)}$ lies in W. Then $m \leqslant (1+\frac{1}{\delta})^{\min\{n^2-n, \dim W\}}$.*

Lemma 24 is proved generally similarly to Lemma 9, but requires a more careful choice of $S(r)$; the proof is omitted due to space constraints.

Proof of Lemma 23. Indeed, the number of equivalence classes under $\approx$ is bounded by $(1+\frac{1}{\delta})^{\max\{n-1, \dim W\}}$: if we have m equivalence classes under $\approx$, then they yield m pairwise inequivalent strings $w_1, \ldots, w_m$. That, in turn, by Lemma 11, gives rise to m matrices $P^{w_1}, \ldots, P^{w_m}$ with $||P^{w_k} - P^{w_\ell}|| \geqslant 2\delta$ for $k \neq \ell$. By Lemma 24, $m \leq (1+\frac{1}{\delta})^{\max\{n-1, \dim W\}}$. Furthermore, the number of equivalence classes under $\sim$ is bounded by $(1+\frac{1}{\delta})^{n-1}$ by Lemma 21. Combining these two observations and using Lemma 20, we obtain the desired result. ◀

The first class of languages with an improved bound on the dimension of the subspace generated by transition matrices inside the brackets is the following variant of unary languages.

▶ **Theorem 25.** *If L is an S-nice language, and $S = a_1^* \cup \ldots \cup a_k^*$ for some $a_1, \ldots, a_k \in \Sigma_0^*$, then there is a DIDPDA recognizing L with $O((1+\frac{1}{\delta})^{kn})$ states.*

Proof. For every word w in S the matrix P^w is of the form $(P^{a_i})^m$ for some $m \in \mathbb{N}_0$ and $1 \leqslant i \leqslant k$. By Cayley–Hamilton theorem, for each a_i the space W_i generated by $\{\, (P^{a_i})^m \mid m \geq 0 \,\}$ has the dimension of at most n; therefore, all such matrices lie in a vector space $W = W_1 + \ldots + W_n$ such that $\dim W \leqslant \sum_{i=1}^{n} \dim W_i \leqslant kn$. It remains to apply Lemma 23. ◀

▶ **Corollary 26.** *If $S = a^*$, then there is a DIDPDA recognizing L with $O((1+\frac{1}{\delta})^n)$ states.*

In particular, the theorem shows that we would not be able to prove the lower bound from Section 5 using a "unary" depth-one language. For instance, the determinization of the automaton that reads $<a^n>$ and verifies that n belongs to some fixed subset yields at most exponential growth in the number of states. This is somewhat similar to the case of probabilistic finite automata, where the state complexity of the determinization in the unary case is also reduced.

The last special case with improved determinization is the case of automata that use strings over an alphabet $\{a, b\}$ inside the brackets, and the transitions by a commute with transitions by b, that is, $P^a P^b = P^b P^a$. In other words, such automata only count the number of as and bs inside the brackets. In this case, there is the following known result on the dimension of the subspace they generate.

▶ **Theorem 27** (Gerstenhaber, 1961)**.** *Let $A, B \in M_n(\mathbb{R})$ be a pair of commuting matrices. Then the dimension of the subalgebra generated by $\{A, B\}$ is at most n.*

▶ **Theorem 28.** *If $S = \{a, b\}^*$ and P^a and P^b commute, then there exists an equivalent DIDPDA with $O((1+\frac{1}{\delta})^n)$ states.*

Proof. If $w \in S$, then $P^w = (P^a)^{|w|_a}(P^b)^{|w|_b}$ lies in the subalgebra generated by $\{P^a, P^b\}$, whose dimension is at most n. Now we can use Lemma 23 to get the desired upper bound. ◀

7 Conclusion

It would be interesting to refine the results on the complexity of determinization for the new model by proving a lower bound on both the number of states and the number of stack symbols. Such a lower bound is known for the determinization of nondeterministic input-driven pushdown automata [20] and of their event-clock real-time extension [19]. The method employed in these papers uses strings of arbitrarily large nesting depth, and the automaton makes non-deterministic choices at each nesting level; however, if the same approach were used in our case, then the probability of error would tend to 1 as the nesting depth goes to infinity. Apparently, a new method would be necessary to prove such a bound in the probabilistic case.

Another interesting direction to pursue is improving the bound with respect to δ. Our current upper bound is polynomial in $\frac{1}{\delta}$. However, it seems possible that there is room for improvement: either a trade-off between the number of states and the probability of error, or perhaps an upper bound that does not depend on δ at all.

References

1 Rajeev Alur, Viraj Kumar, P. Madhusudan, and Mahesh Viswanathan. Congruences for visibly pushdown languages. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, Lisbon, Portugal, July 11-15, 2005, Proceedings*, volume 3580 of *Lecture Notes in Computer Science*, pages 1102–1114. Springer, 2005. doi:10.1007/11523468_89.

2 Rajeev Alur and P. Madhusudan. Visibly pushdown languages. In László Babai, editor, *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 202–211. ACM, 2004. doi:10.1145/1007352.1007390.

3 Rajeev Alur and P. Madhusudan. Adding nesting structure to words. *J. ACM*, 56(3):16:1–16:43, 2009. doi:10.1145/1516512.1516518.

4 Andris Ambainis. The complexity of probabilistic versus deterministic finite automata. In Tetsuo Asano, Yoshihide Igarashi, Hiroshi Nagamochi, Satoru Miyano, and Subhash Suri, editors, *Algorithms and Computation, 7th International Symposium, ISAAC '96, Osaka, Japan, December 16-18, 1996, Proceedings*, volume 1178 of *Lecture Notes in Computer Science*, pages 233–238. Springer, 1996. doi:10.1007/BFb0009499.

5 Nathalie Bertrand, Serge Haddad, and Engel Lefaucheux. Diagnosis in infinite-state probabilistic systems. In Josée Desharnais and Radha Jagadeesan, editors, *27th International Conference on Concurrency Theory, CONCUR 2016, August 23-26, 2016, Québec City, Canada*, volume 59 of *LIPIcs*, pages 37:1–37:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.CONCUR.2016.37.

6 Maria Paola Bianchi, Carlo Mereghetti, Beatrice Palano, and Giovanni Pighizzini. On the size of unary probabilistic and nondeterministic automata. *Fundam. Informaticae*, 112(2-3):119–135, 2011. doi:10.3233/FI-2011-583.

7 Laura Bozzelli. Alternating automata and a temporal fixpoint calculus for visibly pushdown languages. In Luís Caires and Vasco Thudichum Vasconcelos, editors, *CONCUR 2007 – Concurrency Theory, 18th International Conference, CONCUR 2007, Lisbon, Portugal, September 3-8, 2007, Proceedings*, volume 4703 of *Lecture Notes in Computer Science*, pages 476–491. Springer, 2007. doi:10.1007/978-3-540-74407-8_32.

8 Laura Bozzelli, Aniello Murano, and Adriano Peron. Event-clock nested automata. In Shmuel Tomi Klein, Carlos Martín-Vide, and Dana Shapira, editors, *Language and Automata Theory and Applications – 12th International Conference, LATA 2018, Ramat Gan, Israel, April 9-11, 2018, Proceedings*, volume 10792 of *Lecture Notes in Computer Science*, pages 80–92. Springer, 2018. doi:10.1007/978-3-319-77313-1_6.

9 Vojtech Forejt, Petr Jancar, Stefan Kiefer, and James Worrell. Game characterization of probabilistic bisimilarity, and applications to pushdown automata. *Log. Methods Comput. Sci.*, 14(4), 2018. doi:10.23638/LMCS-14(4:13)2018.

10 Rusins Freivalds. Language recognition using probabilistic turing machines in real time, and automata with a push-down store. *Probl. Inform. Transm.*, 15(4):319–323, 1979.

11 Rusins Freivalds. On the growth of the number of states in result of determinization of probabilistic finite automata. *Avtomatika i Viciskitelnaja Tehnika*, 3:39–42, 1982.

12 Rusins Freivalds. Non-constructive methods for finite probabilistic automata. *Int. J. Found. Comput. Sci.*, 19(3):565–580, 2008. doi:10.1142/S0129054108005826.

13 Juraj Hromkovic and Georg Schnitger. On probabilistic pushdown automata. *Inf. Comput.*, 208(8):982–995, 2010. doi:10.1016/j.ic.2009.11.001.

14 Martin Kutrib, Andreas Malcher, and Matthias Wendlandt. Tinput-driven pushdown, counter, and stack automata. *Fundam. Informaticae*, 155(1-2):59–88, 2017. doi:10.3233/FI-2017-1576.

15 Christof Löding, P. Madhusudan, and Olivier Serre. Visibly pushdown games. In Kamal Lodaya and Meena Mahajan, editors, *FSTTCS 2004: Foundations of Software Technology and Theoretical Computer Science, 24th International Conference, Chennai, India, December 16-18, 2004, Proceedings*, volume 3328 of *Lecture Notes in Computer Science*, pages 408–420. Springer, 2004. doi:10.1007/978-3-540-30538-5_34.

16 Kurt Mehlhorn. Pebbling moutain ranges and its application of dcfl-recognition. In J. W. de Bakker and Jan van Leeuwen, editors, *Automata, Languages and Programming, 7th Colloquium, Noordweijkerhout, The Netherlands, July 14-18, 1980, Proceedings*, volume 85 of *Lecture Notes in Computer Science*, pages 422–435. Springer, 1980. doi:10.1007/3-540-10003-2_89.

17 Carlo Mereghetti, Beatrice Palano, and Giovanni Pighizzini. Note on the succinctness of deterministic, nondeterministic, probabilistic and quantum finite automata. *RAIRO Theor. Informatics Appl.*, 35(5):477–490, 2001. doi:10.1051/ita:2001106.

18 Massimiliano Milani and Giovanni Pighizzini. Tight bounds on the simulation of unary probabilistic automata by deterministic automata. *J. Autom. Lang. Comb.*, 6(4):481–492, 2001. doi:10.25596/jalc-2001-481.

19 Mizuhito Ogawa and Alexander Okhotin. On the determinization of event-clock input-driven pushdown automata. In Alexander S. Kulikov and Sofya Raskhodnikova, editors, *Computer Science – Theory and Applications – 17th International Computer Science Symposium in Russia, CSR 2022, Virtual Event, June 29 – July 1, 2022, Proceedings*, volume 13296 of *Lecture Notes in Computer Science*, pages 256–268. Springer, 2022. doi:10.1007/978-3-031-09574-0_16.

20 Alexander Okhotin, Xiaoxue Piao, and Kai Salomaa. Descriptional complexity of input-driven pushdown automata. In Henning Bordihn, Martin Kutrib, and Bianca Truthe, editors, *Languages Alive – Essays Dedicated to Jürgen Dassow on the Occasion of His 65th Birthday*, volume 7300 of *Lecture Notes in Computer Science*, pages 186–206. Springer, 2012. doi:10.1007/978-3-642-31644-9_13.

21 Alexander Okhotin and Kai Salomaa. Descriptional complexity of unambiguous input-driven pushdown automata. *Theor. Comput. Sci.*, 566:1–11, 2015. doi:10.1016/j.tcs.2014.11.015.

22 Alexander Okhotin and Victor L. Selivanov. Input-driven pushdown automata on well-nested infinite strings. In Rahul Santhanam and Daniil Musatov, editors, *Computer Science – Theory and Applications – 16th International Computer Science Symposium in Russia, CSR 2021, Sochi, Russia, June 28 – July 2, 2021, Proceedings*, volume 12730 of *Lecture Notes in Computer Science*, pages 349–360. Springer, 2021. doi:10.1007/978-3-030-79416-3_21.

23 Michael O. Rabin. Probabilistic automata. *Inf. Control.*, 6(3):230–245, 1963. doi:10.1016/S0019-9958(63)90290-0.

24 Nguyen Van Tang and Mizuhito Ogawa. Event-clock visibly pushdown automata. In Mogens Nielsen, Antonín Kucera, Peter Bro Miltersen, Catuscia Palamidessi, Petr Tuma, and Frank D. Valencia, editors, *SOFSEM 2009: Theory and Practice of Computer Science, 35th Conference on Current Trends in Theory and Practice of Computer Science, Spindleruv Mlýn, Czech Republic, January 24-30, 2009. Proceedings*, volume 5404 of *Lecture Notes in Computer Science*, pages 558–569. Springer, 2009. doi:10.1007/978-3-540-95891-8_50.

25 Burchard von Braunmühl and Rutger Verbeek. Input driven languages are recognized in log n space. In Marek Karplnski and Jan van Leeuwen, editors, *Topics in the Theory of Computation*, volume 102 of *North-Holland Mathematics Studies*, pages 1–19. North-Holland, 1985. doi:10.1016/S0304-0208(08)73072-X.

26 Tobias Winkler, Christina Gehnen, and Joost-Pieter Katoen. Model checking temporal properties of recursive probabilistic programs. In Patricia Bouyer and Lutz Schröder, editors, *Foundations of Software Science and Computation Structures – 25th International Conference, FOSSACS 2022, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Munich, Germany, April 2-7, 2022, Proceedings*, volume 13242 of *Lecture Notes in Computer Science*, pages 449–469. Springer, 2022. doi:10.1007/978-3-030-99253-8_23.