The Online Space Complexity of Probabilistic Languages

Nathanaël Fijalkow

University of Oxford, United Kingdom

Abstract

The notion of online space complexity, introduced by Karp in 1967, quantifies the amount of space required to solve a given problem using an online algorithm.

Motivated by the seminal paper of Rabin from 1963 introducing probabilistic automata, we study the online space complexity of probabilistic languages.

We prove two results: the first is that probabilistic languages can have arbitrarily high online space complexity, and the second is that determining the online space complexity of a probabilistic language is undecidable.

Keywords: Online Complexity, Probabilistic Languages, Automata, Online Algorithms, Complexity Theory
2010 MSC: 68W27, Online algorithms

1. Introduction

1.1. Online Space Complexity

Consider the following task: given a property over finite words, we would like to construct a machine processing words in an *online* fashion checking this property. The term "online" means that the symbols forming the word are input one after the other, from left to right, in one pass.

A typical example is a machine presented with a sequence of a's and b's, which should at any point determine whether the sequence read so far contains exactly as many a's as b's. The canonical machine solving this problem uses as memory one counter taking integer values, the difference between the number of a's and the number of b's. This machine is of linear size, because after reading at most n letters it can be in $2n + 1$ different states; as we shall see, it is optimal, meaning that this problem does not have sublinear online space complexity.

The notion of *online computing* has been identified as a fundamental research question in the 80s, and has since then blossomed into several directions with various ap-

[☆] This journal version extends the conference paper with the same name, published in the proceedings of LFCS'2016 [1], as well as the results presented in [2].

proaches. We are concerned here with *complexity* questions, and in particular about the use of *space* for online algorithms.

We follow the approach of Karp [3], who represents online algorithms by infinite deterministic automata. In this setting, the size of an automaton is the function $\mathbb{N} \to \mathbb{N}$ associating to n the number of states reachable from the initial state by reading words of length at most n. A language has *online space complexity* $f : \mathbb{N} \to \mathbb{N}$ if there exists an automaton of size f recognising it.

This approach led to exciting developments, such as a series of results by Hartmanis and Shank who gave tight bounds on the complexity of checking the primality of a number written in binary [4], for online algorithms.

We investigate the online space complexity of probabilistic automata, a simple probabilistic model of computation introduced by Rabin in his seminal paper [5]. This study is motivated by the section "approximate calculation of matrix products" in this paper; in the end of this section, Rabin states a result, without proof; the aim of this paper is to substantiate this claim, *i.e.* formalising and proving the result.

Contributions and organisation of the paper. The remainder of this section is devoted to related works. The definitions are given in Section 2.

We prove two results:

- In Section 3, we exhibit a probabilistic language which does not have subexponential online space complexity. This substantiate the claim of Rabin.
- In Section 4, we prove that determining the online space complexity of a probabilistic automaton is undecidable.

1.2. Related Works

The research area concerned with *online computing* introduced different models to represent online algorithms. Unlike an *offline algorithm*, which has access to the whole input, an *online algorithm* is presented with its input in a restricted way: it has to process it letter by letter.

We discuss three frameworks that study different aspects of online computing: first the model of *dynamic algorithms*, which generalises online algorithms, second *streaming algorithms*, which take into account both time and space complexity to construct online algorithms, and third the *competitive analysis of online algorithms*, which aims at quantifying the influence of restricting to online algorithms over offline algorithms.

In the setting of *dynamic algorithms*, the input can go through a series of changes, and the challenge is to construct a data structure together with algorithms for three tasks: initialising, updating and querying the data structure. Whereas for online algorithms, the changes are only insertions, dynamic algorithms also consider deletions, and sometimes more complicated operations.

The seminal paper of Patnaik and Immerman [6] introduced the dynamic complexity class DynFO, which is the set of problems whose solutions can be maintained by

first-order formulae. This motivated the line of work called dynamic complexity, which has seen recent impressive progress, see for instance [7].

The field of *streaming algorithms* was initiated by a series of papers; Munro and Paterson [8], then Flajolet and Martin [9], followed by the foundational paper of Alon, Matias and Szegedy [10]. A streaming algorithm has both a limited available memory, much smaller than the input size, and a limited processing time per letter. The challenge there is to use these constrained resources to compute relevant information about the processed input, such as for instance statistics on frequency distributions.

The field of *competitive analysis of online algorithms*, initiated by Sleator and Tarjan [11], and by Karp [12], compares the performances between offline and online algorithms, ignoring complexity issues. In this setting, each solution is assigned a real value, assessing its quality. An offline algorithm, having access to the whole input, can select the best solution. An online algorithm, however, has to make choices ignoring part of the input that is still to be read. The question is then whether there exist online algorithms that can perform nearly as good as offline algorithms, up to a competitive ratio.

2. Definitions

2.1. Online Space Complexity Classes

We fix an *alphabet* A, which is a finite set of letters. An instance of a problem is given by a *word*, which is a finite sequence of letters, often denoted $w = a_1 a_2 \cdots a_n$, where a_i's are letters from the alphabet A, *i.e.* $a_i \in A$. We say that w has length n. We denote A^* the set of all words and $A^{\leq n}$ the set of words of length at most n.

A computational problem is given by a set of words L, called a *language*; *i.e.* $L \subseteq A^*$. The online space complexity of a language measures the size of an abstract machine able to recognise the language in an online way: the machine processes words letter by letter, and must at any point be able to determine whether the word read so far belongs to the language or not.

Following Karp [3], we use *deterministic infinite automata* to model our machines.

Definition 1 (Automaton). *An automaton is given by a (potentially infinite) set Q of states, an initial state $q_0 \in Q$, a transition function $\delta : Q \times A \to Q$ and a set of accepting states $F \subseteq Q$.*

When processing a word $w = a_1 a_2 \cdots a_n$, the automaton assumes a sequence of states $q_0 q_1 \cdots q_n$, that we call the run on w, defined inductively by $q_{i+1} = \delta(q_i, a_i)$. It is unique, since we assume the automaton to be *deterministic* and the transition function to be a total function. The word w is accepted if q_n is accepting, *i.e.* if $q_n \in \mathcal{A}$, and rejected otherwise. The language recognised by an automaton is the set of words accepted by this automaton.

Definition 2 (Size of an Automaton). *The size of an automaton is the function $s : \mathbb{N} \to \mathbb{N}$ defined by $s(n)$ being the number of different states reached by all words of length at most n.*

Note that in complexity theory, it is usual to count the size of an object by the size of its description. Here, it would be natural, instead of counting how many different states are reached, how many bits are necessary to describe these states; this amounts to consider the logarithm of the number of states. We do not use this definition as it would erase the differences between, for instance, linearly many and quadratically many states.

For two functions $f, g : \mathbb{N} \to \mathbb{N}$, we say that f is smaller than g, denoted $f \leq g$, if for all n, $f(n) \leq g(n)$.

Definition 3 (Online Space Complexity Class). *For a function $f : \mathbb{N} \to \mathbb{N}$, the class of languages $\mathrm{OSC}(f)$ consists of all languages which are recognised by an automaton of size smaller than f.*

The notation $\mathrm{OSC}(O(f))$ involves a constant multiplicative factor: the language L is in $\mathrm{OSC}(O(f))$ if there exists an automaton of size smaller than $C \cdot f$ recognising L, for some constant C.

Throughout the paper, we denote the function $f : n \mapsto f(n)$ by $f(n)$, making n the implicit variable. For instance, $\mathrm{OSC}(O(n^2))$ denotes the languages having quadratic online space complexity.

For the sake of succinctness, the acronym OSC will be used in place of online space complexity.

2.2. Remarks and Examples

We begin with a few simple remarks. The first remark is that the size of an automaton satisfies the following inequality, for all n:

$$s(n) \leq 1 + \mathrm{Card}(A) + \mathrm{Card}(A^2) + \cdots + \mathrm{Card}(A^n) = \frac{\mathrm{Card}(A)^{n+1} - 1}{\mathrm{Card}(A) - 1}.$$

It follows that the maximal complexity of a language is exponential, and the online space complexity classes are relevant for functions smaller than exponential.

We denote Reg the class of regular languages, *i.e.* those recognised by automata.

Theorem 1.

- $\mathrm{OSC}(O(1)) = \mathrm{Reg}$*, i.e. a language has constant online space complexity if, and only if, it is regular.*
- $\mathrm{OSC}\left(O(\mathrm{Card}(A)^n)\right)$ *contains all languages.*

The first item is a mere rephrasing. For the second item, consider a language L, we construct an automaton recognising L of exponential size. Its set of states is A^*, the initial state is ε and the transition function is defined by $\delta(w, a) = wa$. The set of accepting states is simply L itself.

We give three examples.

- Define $\textsc{Maj}_2 = \left\{w \in \{a,b\}^* \mid |w|_a > |w|_b\right\}$, the majority language over two letters. Here $|w|_a$ denotes the number of occurrences of the letter a in w. We construct an automaton of linear size recognising $\textsc{Maj}_2$: its set of states is $\mathbb{Z}$, the integers, the letter a acts as $+1$ and the letter b as -1, and the set of accepting states is $\mathbb{N}$, the positive integers. After reading the word w, the state is $|w|_a - |w|_b$, implying that the automaton has linear size. So $\textsc{Maj}_2 \in \mathrm{OSC}(O(n))$, and we will show in subsection 2.3 that this bound is tight.

- Define $\textsc{Eq} = \left\{w \in \{a,b,c\}^* \mid |w|_a = |w|_b = |w|_c\right\}$. We construct an automaton of quadratic size recognising $\textsc{Eq}$: its set of states is $\mathbb{Z}^2$, the letter a acts as $(+1,+1)$, the letter b as $(-1,0)$, the letter c as $(0,-1)$, and the only accepting state is $(0,0)$. After reading the word w, the state is $(|w|_a - |w|_b, |w|_a - |w|_c)$, implying that the automaton has quadratic size. So $\textsc{Eq} \in \mathrm{OSC}(O(n^2))$, and we will show in subsection 2.3 that this bound is tight.

- Define $\textsc{Squares} = \{ww \mid w \in A^*\}$. This language has maximal online space complexity; specifically, we will show in subsection 2.3 that $\textsc{Squares}$ does not have subexponential online space complexity.

2.3. The Myhill-Nerode Point of View

We present an equivalent point of view on the online space complexity, based on Myhill-Nerode equivalence relation.

Let w be a finite word, define its left quotient with respect to L by

$$w^{-1}L = \{u \mid wu \in L\}.$$

A well known result from automata theory states that for all regular languages, there exists a minimal deterministic finite automaton, called the *syntactic automaton*, whose states is the set of left quotients.

This construction extends *mutatis mutandis* when dropping the assumption that the automaton has finitely many states. The statement gives precise lower bounds on the online space complexity of the language.

Formally, consider a language L, we define the syntactic automaton of L, denoted $\mathcal{A}_L$, as follows. We define the set of states as the set of all left quotients: $\left\{w^{-1}L \mid w \in A^*\right\}$. The initial state is $\varepsilon^{-1}L$, and the transition function is defined by $\delta(w^{-1}L, a) = (wa)^{-1}L$. Finally, the set of accepting states is $\left\{w^{-1}L \mid w \in L\right\}$.

Denote s_L the size of the syntactic machine of L.

Theorem 2 (Reformulation of Myhill-Nerode Theorem [13]).

- *$\mathcal{A}_L$ recognises L, so $L \in \mathrm{OSC}(s_L)$,*
- *for all f, if $L \in \mathrm{OSC}(f)$, then $f \geq s_L$.*

Note that the implies the existence of a minimal function f such that $L \in \mathrm{OSC}(f)$; in other words $L \in \mathrm{OSC}(f)$ if, and only if, $f \geq s_L$. This was not clear a priori, because the order on functions is partial.

The first item is routinely proved. For the second item, assume towards contradiction that there exists f such that $L \in \mathrm{OSC}(f)$ and $f \not\geq s_L$, *i.e.* there exists n such that $f(n) < s_L(n)$. Consider an automaton $\mathcal{A}$ recognising L of size f. Since $f(n) < s_L(n)$, there exists two words u and v of length n such that $u^{-1}L \neq v^{-1}L$ but in $\mathcal{A}$ the words u and v lead to the same state. The left quotients $u^{-1}L \neq v^{-1}L$ being different, there exists a word w such that $uw \in L$ and $vw \notin L$, or the other way around. But since the words u and v lead to the same state and $\mathcal{A}$ is deterministic, this state must be both accepting and rejecting, contradiction.

We illustrate the use of Theorem 2 on the examples introduced in subsection 2.2.

We say that L has sub-f online space complexity if it is recognised by an automaton of sub-f size g, meaning that $g = o(f)$.

- We show that $\textsc{Maj}_2$ does not have sublinear online space complexity.

  Fix n. The words $a^{n+k}b^{n-k}$, for $0 \leq k \leq n$, have length $2n$ and all induce pairwise distinct left quotients, since $a^{n+k}b^{n-k} \cdot b^p \in \textsc{Maj}_2$ if, and only if, $p < 2k$. The result follows from Theorem 2.

- We show that Eq does not have subquadratic online space complexity.

  Fix n. The words $a^{2n-p-q}b^pc^q$, for $0 \leq p, q \leq n$, have length $2n$ and all induce pairwise distinct left quotients, since $a^{2n-p-q}b^pc^q \cdot a^{k+\ell}b^{2n-k}c^{2n-\ell} \in \textsc{Eq}$ if, and only if, $k = p$ and $\ell = q$. The result follows from Theorem 2.

- We show that Squares does not have sublexponential online space complexity.

  Fix n. All words of length n induce pairwise distinct left quotients, since for v of length n, we have that $wv \in \textsc{Squares}$ if, and only if, $v = w$. The result follows from Theorem 2.

We conclude this section by observing that the examples studied above show that online space complexity and circuit complexity are orthogonal. Indeed:

- The language Maj has linear online space complexity, but does not belong to AC^0, *i.e.* it has a rather big circuit complexity.

  Another example of a language having a small online space complexity and a big circuit complexity is Parity: it is regular, and recognised by an automaton of size 2, but does not belong to AC^0.

- The language Squares does not have subexponential online space complexity, but it has a very small circuit complexity: it is recognised by a family of circuits of linear size of constant depth.

2.4. Probabilistic Automata

Let Q be a finite set of states. A distribution over Q is a function $\delta : Q \to [0, 1]$ such that $\sum_{q \in Q} \delta(q) = 1$. We denote $\mathcal{D}(Q)$ the set of distributions over Q.

Definition 4 (Probabilistic Automaton). *A probabilistic automaton $\mathcal{A}$ is given by a finite set of states Q, a transition function $\phi : A \to (Q \to \mathcal{D}(Q))$, an initial state $q_0 \in Q$, and a set of final states $F \subseteq Q$.*

In a transition function ϕ, the quantity $\phi(a)(s,t)$ is the probability to go from the state s to the state t reading the letter a. A transition function naturally induces a morphism $\phi : A^* \to (Q \to \mathcal{D}(Q))$. We denote $\mathbb{P}_{\mathcal{A}}(s \xrightarrow{w} t)$ the probability to go from a state s to a state t reading w on the automaton $\mathcal{A}$, *i.e.* $\phi(w)(s,t)$.

The *acceptance probability* of a word $w \in A^*$ by $\mathcal{A}$ is $\sum_{t \in F} \phi(w)(q_0, t)$, which we denote $\mathbb{P}_{\mathcal{A}}(w)$.

The following threshold semantics was introduced by Rabin [5].

Definition 5 (Probabilistic Language). *Let $\mathcal{A}$ be a probabilistic automaton and x a threshold in $(0,1)$, it induces the probabilistic language*

$$L^{>x}(\mathcal{A}) = \{w \in A^* \mid \mathbb{P}_{\mathcal{A}}(w) > x\}.$$

3. Substantiating the Claim of Rabin

In the section called "approximate calculation of matrix products" in the paper introducing probabilistic automata [5], Rabin asks the following question: is it possible, given a probabilistic automaton, to construct an algorithm which reads words and compute the acceptance probability in an online fashion?

He first shows that this is possible under some restrictions on the probabilistic automaton, and concludes the section by stating that "*an example due to R. E. Stearns shows that without assumptions, a computational procedure need not exist*". The example is not given, and to the best of the author's knowledge, has never been published anywhere.

In this section, we substantiate this claim, in the framework of online space complexity. Note that the formalisation of Rabin's claim is subject to discussions, as for instance Rabin asks whether the acceptance probability can be computed up to a given precision; in our setting, the acceptance probability is not actually computed, but only compared to a fixed threshold, following Rabin's definition of probabilistic languages.

The following result shows that there exists a probabilistic automaton defining a language of maximal (exponential) online space complexity.

Theorem 3. *There exists a probabilistic automaton $\mathcal{A}$ such that $L^{>\frac{1}{2}}(\mathcal{A})$ does not have subexponential online space complexity.*

In the original paper introducing probabilistic automata, Rabin [5] gave an example of a probabilistic automaton $\mathcal{A}$ computing the binary decomposition function (over the alphabet $\{0,1\}$), denoted bin, *i.e.* $\mathbb{P}_{\mathcal{A}}(u) = \mathrm{bin}(u)$, defined by

$$\mathrm{bin}(a_1 \dots a_n) = \frac{a_1}{2^n} + \dots + \frac{a_n}{2^1}$$

(*i.e.* $0.a_n \dots a_1$ in binary). We show that adding one letter and one transition to this probabilistic automaton induces a language which does not have subexponential online space complexity.

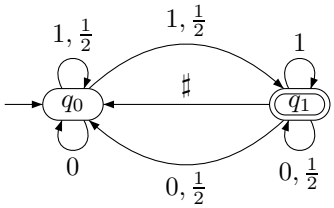

Figure 1: The initial state is marked by an ingoing arrow and the accepting state by an outgoing arrow. The first symbol over a transition is a letter (either 0, 1, or ♯). The second symbol (if given) is the probability of this transition. If there is only one symbol, then the probability of the transition is 1.

The automaton $\mathcal{A}$ is represented in Figure 1. The alphabet is $A = \{0, 1, \sharp\}$. The only difference between the automaton proposed by Rabin [5] and this one is the transition over $\sharp$ from q_1 to q_0. As observed by Rabin, a simple induction shows that for u in $\{0,1\}^*$, we have $\mathbb{P}_{\mathcal{A}}(u) = \mathrm{bin}(u)$.

Let $w \in A^*$, it decomposes uniquely into $w = u_1\sharp u_2\sharp \cdots \sharp u_k$, where $u_i \in \{0,1\}^*$. Observe that $\mathbb{P}_{\mathcal{A}}(w) = \mathrm{bin}(u_1) \cdot \mathrm{bin}(u_2) \cdots \mathrm{bin}(u_k)$.

Consider an automaton recognising $L^{>\frac{1}{2}}(\mathcal{A})$ and fix n. The binary decomposition function maps words of length n to rationals of the form $\frac{a}{2^n}$, for $0 \leq a < 2^n$. Consider two different words u and v in $\{0,1\}^*$ of length n, we show that $(u1)^{-1}L^{>\frac{1}{2}}(\mathcal{A}) \neq (v1)^{-1}L^{>\frac{1}{2}}(\mathcal{A})$.

Without loss of generality assume $\mathrm{bin}(u1) < \mathrm{bin}(v1)$; observe that $\frac{1}{2} \leq \mathrm{bin}(u1) < \mathrm{bin}(v1)$. There exists w in $\{0,1\}^*$ such that $\mathrm{bin}(u1) \cdot \mathrm{bin}(w) < \frac{1}{2}$ and $\mathrm{bin}(v1) \cdot \mathrm{bin}(w) > \frac{1}{2}$: it suffices to choose w such that $\mathrm{bin}(w)$ is in $\left(\frac{1}{2\mathrm{bin}(v1)}, \frac{1}{2\mathrm{bin}(u1)}\right)$, which exists by density of the dyadic numbers in $(0,1)$. Thus, $(u1)^{-1}L^{>\frac{1}{2}}(\mathcal{A}) \neq (v1)^{-1}L^{>\frac{1}{2}}(\mathcal{A})$, and we exhibited exponentially many words having pairwise distinct left quotients.

It follows from Theorem 2 that $L^{>\frac{1}{2}}(\mathcal{A})$ does not have subexponential online space complexity.

The property above can be strengthened: for every number x in $(0,1)$, the language $L^{>x}(\mathcal{C})$ does not have subexponential online space complexity. The proof follows the same line, we explain the differences. Fix s in $\{0,1\}^*$ such that $\mathrm{bin}(s) > x$, we show that $(us)^{-1}L^{>x}(\mathcal{A}) \neq (vs)^{-1}L^{>x}(\mathcal{A})$. To this end, one needs to choose w in $\left(\frac{x}{\mathrm{bin}(vs)}, \frac{x}{\mathrm{bin}(us)}\right)$.

We will use this stronger property in the next section.

4. Undecidability of the Classification Problems for Probabilistic Automata

We give another application for the automaton presented above. We consider the following decision problem, called the online space complexity classification problem,

parameterised by a function $f : \mathbb{N} \to \mathbb{N}$: given a probabilistic automaton $\mathcal{A}$, determine whether $L^{>\frac{1}{2}}(\mathcal{A})$ is in $\mathrm{OSC}(O(f))$.

Theorem 4. *(Undecidability of the regularity problem) The online space complexity classification problem is undecidable for probabilistic automata, for all functions subexponential f, i.e. such that there exists $C > 1$ with $f = o(C^n)$.*

Note that as a special case, we obtain the undecidability of the regularity problem: indeed, for f the constant function equal to 1, the online space complexity classification problem reduces to the regularity problem. The undecidability of the regularity problem was originally proved in [16], and we gave the following simple proof of this result in [2].

To prove Theorem 4, we construct a reduction from the emptiness problem to the online space complexity classification problem. The *emptiness problem* was considered by Rabin: given a probabilistic automaton $\mathcal{A}$, determine whether $L^{>\frac{1}{2}}(\mathcal{A})$ is non-empty, *i.e.* whether there exists a word w such that $\mathbb{P}_{\mathcal{A}}(w) > \frac{1}{2}$. It has been shown undecidable by Paz [14]; a simple undecidability proof was given by Gimbert and Oualhadj in [15].

Theorem 5 ([14]). *The emptiness problem is undecidable.*

We proceed with the proof of Theorem 4, constructing a reduction from the emptiness problem. Roughly speaking, the idea is to use the automaton given in Section 3 to "amplify" complicated behaviours. As observed at the end of the section, for all x in $(0, 1)$, the language $L^{>x}(\mathcal{C})$ does not have subexponential online space complexity.

Let $\mathcal{A}$ be a probabilistic automaton over an alphabet A. We construct a probabilistic automaton $\mathcal{B}$ such that:

$$L^{>\frac{1}{2}}(\mathcal{A}) \text{ is empty} \quad \text{if, and only if,} \quad L^{>\frac{1}{2}}(\mathcal{B}) \text{ is in } \mathrm{OSC}(O(f)).$$

The automaton $\mathcal{B}$ is over the alphabet $B = A \uplus C$ where $C = \{0, 1, \sharp\}$, and uses the automaton $\mathcal{C}$ from Section 3. It is obtained as the sequential composition of $\mathcal{A}$ and $\mathcal{C}$: it starts in $\mathcal{A}$ and from every final state of $\mathcal{A}$ moves by $\sharp$ to the initial state of $\mathcal{C}$. The initial state of $\mathcal{B}$ is the initial state of $\mathcal{A}$, the only final state of $\mathcal{B}$ is the final state of $\mathcal{C}$.

For $u \in A^*$ and $v \in C^*$, we have $\mathbb{P}_{\mathcal{B}}(u \cdot \sharp \cdot v) = \mathbb{P}_{\mathcal{A}}(u) \cdot \mathbb{P}_{\mathcal{C}}(v)$. A word which is not in $A^* \cdot \sharp \cdot C^*$ has no accepting run, so is accepted with probability 0.

- Assume that $L^{>\frac{1}{2}}(\mathcal{A})$ is empty. Thanks to the above observation we have that $L^{>\frac{1}{2}}(\mathcal{B})$ is empty, so in particular it is in $\mathrm{OSC}(O(f))$, whatever is f.
- Conversely, assume that $L^{>\frac{1}{2}}(\mathcal{A})$ is non-empty. Let u be a word such that $\mathbb{P}_{\mathcal{A}}(u) > \frac{1}{2}$. Observe that $L^{>\frac{1}{2}}(\mathcal{B}) \cap (u \cdot \sharp \cdot C^*) = u \cdot \sharp \cdot L^{>x}(\mathcal{C})$, where $x = \frac{1}{2 \cdot \mathbb{P}_{\mathcal{A}}(u)}$ is in $(0, 1)$. The language $L^{>x}(\mathcal{C})$ does not have subexponential complexity, so the same holds for $u \cdot \sharp \cdot L^{>x}(\mathcal{C})$, implying that $L^{>\frac{1}{2}}(\mathcal{B})$ is not in $\mathrm{OSC}(O(f))$.

Conclusion

We studied the online space complexity, quantifying how hard it is to solve a problem when the input is given in an online fashion, focusing on the space consumption.

We considered the online space complexity of probabilistic automata, as hinted by Rabin in [5], and showed that probabilistic automata give rise to languages of high (maximal) online space complexity. Further, we proved that determining the online space complexity of a probabilistic automaton is undecidable.

We mention some directions for future research about online space complexity.

The first is to give characterisations of the natural online space complexity classes (linear, quadratic, polynomial). Such characterisations could be in terms of logics, as it is done in descriptive complexity, or algebraic, as it is done in the automata theory. The canonical example is languages of constant online space complexity, which are exactly regular languages, defined by monadic second-order logic.

A second direction would be to extend the framework of online space complexity to quantitative queries. Indeed, we defined here the online space complexity of a language, *i.e.* of qualitative queries: a word is either inside, or outside the language.

A third intriguing question is the existence of a dichotomy for the online space complexity of probabilistic automata. Is the following conjecture true: for every probabilistic language, its online space complexity is either polynomial or exponential?

Acknowledgements

The author would like to thank Michał Skrzypczak for thoughtful suggestions and the anonymous referees for their useful comments.

References

[1] N. Fijalkow, The online space complexity of probabilistic languages, in: LFCS'2016, 2016, pp. 106–116.

[2] N. Fijalkow, M. Skrzypczak, Irregular behaviours for probabilistic automata, in: RP'2015, 2015, pp. 33–36.

[3] R. M. Karp, Some bounds on the storage requirements of sequential machines and turing machines, Journal of the ACM 14 (3).

[4] J. Hartmanis, H. Shank, Two memory bounds for the recognition of primes by automata, Mathematical Systems Theory 3 (2).

[5] M. O. Rabin, Probabilistic automata, Information and Control 6 (3) (1963) 230–245.

[6] S. Patnaik, N. Immerman, Dyn-FO: A parallel, dynamic complexity class, in: PODS'94, 1994, pp. 210–221.

[7] T. Zeume, T. Schwentick, On the quantifier-free dynamic complexity of reachability, Information and Computation 240 (2015) 108–129.

[8] J. I. Munro, M. Paterson, Selection and sorting with limited storage, Theoretical Computer Science 12 (1980) 315–323.

[9] P. Flajolet, G. N. Martin, Probabilistic counting algorithms for data base applications, Journal of Computer and System Sciences 31 (2) (1985) 182–209.

[10] N. Alon, Y. Matias, M. Szegedy, The space complexity of approximating the frequency moments, in: STOC'96, 1996, pp. 20–29.

[11] D. D. Sleator, R. E. Tarjan, Amortized efficiency of list update and paging rules, Communications of the ACM 28 (2) (1985) 202–208.

[12] R. M. Karp, On-line algorithms versus off-line algorithms: How much is it worth to know the future?, in: IFIP'92, 1992, pp. 416–429.

[13] A. Nerode, Linear automaton transformations, Proceedings of the American Mathematical Society 9 (4) (1958) 541–544.

[14] A. Paz, Introduction to probabilistic automata, Academic Press, 1971.

[15] H. Gimbert, Y. Oualhadj, Probabilistic automata on finite words: Decidable and undecidable problems, in: ICALP (2), 2010, pp. 527–538.

[16] A. Bertoni, Mathematical methods of the theory of stochastic automata, in: Mathematical Foundations of Computer Science, 1975, pp. 9–22.