The Trichotomy of Regular Property Testing

Gabriel Bathie
LaBRI, Université de Bordeaux
DIENS, Paris, France

Nathanaël Fijalkow
LaBRI, CNRS, Université de Bordeaux, France

Corto Mascle
LaBRI, Université de Bordeaux, France

Abstract

Property testing is concerned with the design of algorithms making sublinear number of queries to distinguish whether the input satisfies a given property or is far from having this property. A seminal paper of Alon, Krivelevich, Newman, and Szegedy in 2001 introduced property testing of formal languages: the goal is to determine whether an input word belongs to a given language, or is far from any word in that language. They constructed the first property testing algorithm for the class of all regular languages. This opened a line of work with improved complexity results and applications to streaming algorithms. In this work, we show a trichotomy result: the class of regular languages can be divided into three classes, each associated with a query complexity. Our analysis yields effective characterizations for all three classes using so-called minimal blocking sequences, reasoning directly and combinatorially on automata.

2012 ACM Subject Classification Theory of computation → Regular languages

Keywords and phrases property testing, regular languages

1 Introduction

Property testing was defined by Goldreich, Goldwasser, and Ron [14] in 1998: it is the study of very fast randomized approximate decision procedures on huge objects, where very fast typically means sublinear; the algorithm does not even scan the whole input. A very active branch of property testing focuses on graph properties, for instance one can test whether a given graph appears as a subgraph [3] or as an induced subgraph [4], and more generally every monotone graph property can be tested with one-sided error [6]. Other families of objects heavily studied under this algorithmic paradigm include probabilistic distributions [19, 9] combined with privacy constraints [2], numerical functions [8, 20], and programs [11, 10]. We refer to the book of Goldreich [13] for an overview of the field of property testing.

In this paper we continue the line of work initiated by Alon, Krivelevich, Newman, and Szegedy [5] which studies property testing of formal languages: given a language L (a set of finite words), the goal is to determine whether an input word u belongs to the language or is ε-far from it, where ε is the precision parameter. We assume random access to the input word: a query specifies a position in the word and asks for the letter in this position. To measure the distance of a word to a language we assume a metric over words; two natural choices include the Hamming distance (the number of positions at which two words differ) or the edit distance (the number of edits to transform one word into the other one). The seminal paper [5] showed a surprising result: all regular languages (meaning, languages recognised by deterministic finite automata) are testable with $\mathcal{O}(\log^3(\varepsilon^{-1})/\varepsilon)$ queries, where the $\mathcal{O}(\cdot)$ notation hides constants that depend on the language, but, crucially, not on the length of the input word.

A series of papers built upon this work, improving the query complexity (i.e. the number of queries). The original paper [5] identified the class of *trivial* regular languages, those for which the answer is always *yes* or always *no* for large enough n, and showed that testing membership in a non-trivial regular language requires $\Omega(1/\varepsilon)$ queries. Building upon their work, Magniez and de Rougemont [18] extended their result by giving a tester using $\mathcal{O}(\log^2(\varepsilon^{-1})/\varepsilon)$ queries for the edit distance with moves, and François et al. [12] gave a tester using $\mathcal{O}(1/\varepsilon^2)$ queries for the case of the weighted edit distance. More recently, Bathie and Starikovskaya [7] gave a tester for the edit distance using $\mathcal{O}(\log(\varepsilon^{-1})/\varepsilon)$ queries, and showed that there exists a *hard* regular language that cannot be tested with asymptotically fewer queries. However, there exist *easy* regular languages that can be tested with $\mathcal{O}(1/\varepsilon)$ queries. These results raise the following questions:

1. are there regular languages with a query complexity different from asymptotically 0, $\Theta(1/\varepsilon)$ and $\Theta(\log(\varepsilon^{-1})/\varepsilon)$?
2. is there a combinatorial and effective characterization of the languages in each class?

In this work, we answer both questions almost completely: we show a trichotomy theorem that classifies all regular languages in one of the three classes: trivial (asymptotically 0 queries[1]), easy ($\Theta(1/\varepsilon)$ queries), and hard ($\Omega(\log(\varepsilon^{-1})/\varepsilon)$ queries). In the case of languages recognised by strongly connected NFAs, we even provide a matching upper bound of $\mathcal{O}(\log(\varepsilon^{-1})/\varepsilon)$ for all languages. Our characterization of the three classes relies on the combinatorial notion of minimal blocking factors (and sequences).

We can therefore ask the meta-question: can we determine whether a given regular language is trivial, easy, or hard? Answering this question has practical motivations: determining to which class the language belongs enables choosing the appropriate most efficient property testing algorithm. We show that the meta-question is complete for the complexity class PSPACE (Turing machines working in polynomial space).

2 Overview of the paper

In this overview we assume familiarity with classical notions; all definitions can be found in Section 3. Let us start with the notion of a property tester for a language L: the goal is to determine whether an input word u belongs to the language L, or whether it is ε-far from it. We say that u of length n is *ε-far from L* with respect to a metric d over words if all words $v \in L$ satisfy $d(u, v) \geq \varepsilon n$, written $d(u, L) \geq \varepsilon n$. Throughout this work and unless explicitly stated otherwise, we will consider the case where d is the Hamming distance, defined for two words u and v as the number of positions at which they differ if they have the same length, and as $+\infty$ otherwise. In that case, $d(u, L) \geq \varepsilon n$ means that one cannot change a proportion ε of the letters in u to obtain a word in L. We assume random access to the input word: a query specifies a position in the word and asks for the letter in this position.

▶ **Definition 2.1.** *A property tester for the language L and precision ε is a randomized algorithm T that, for any input u of length n, given random access to u, satisfies the following properties:*

$$\text{if } u \in L, \text{ then } T(u) = 1, \tag{1}$$

$$\text{if } u \text{ is } \varepsilon\text{-far from } L, \text{ then } \mathbb{P}\left(T(u) = 0\right) \geq 2/3. \tag{2}$$

[1] By *asymptotically* 0 *queries*, we mean that for every small enough $\varepsilon > 0$, for large enough n, the answer is either *yes* for all words of length n or *no* for all, and only depends on n, thus the algorithm does not need to query the input word.

The query complexity of T is a function of n and ε that counts the maximum number of letters of the input that T reads over all inputs of length n and over all possible random choices. In this paper we are interested in property testers whose query complexity is independent of n, and only depends on ε.

More precisely, the definition of property testers given above is called "property testers with perfect completeness": they always accept positive instances. Because they are based on the notion of blocking factors that we will discuss below, all known testers for regular languages [5, 18, 12, 7] have perfect completeness.

We say that a tester is *non-adaptive* if the index of a query does not depend on the result of previous queries. Alternatively, a non-adaptive tester can be understood as an algorithm that first sends the index of all of its queries, receives the result of all the queries, and then returns its output.

Infinite languages

Let us make a trivial observation: if L is finite, meaning that it contains finitely many words, then it is trivial. Let N denote the maximum length of a word in L: as L is finite, N is also finite. We can test L as follows: if the input has length less than N, query all of it and check whether it is one of the finitely many words of L and answer accordingly. Otherwise, if the length of the input is greater than N, answer *no*. This tester makes no queries for $n > N$, hence L is trivial. For this reason, we only consider infinite languages L; this will make some technical statements nicer.

Easy languages

Let us consider the language $L_1 = a^*$ consisting of words containing only a's, over the alphabet $\{a, b\}$. For a word $u \in \{a, b\}^*$, the distance $d(u, L)$ is the number of b's in u. Here is a very simple property tester for L_1: given a word of length n, sample $1/\varepsilon$ letters at random and answer no if we find a b, and yes otherwise. If $u \in L_1$, it contains no b to the algorithm returns yes, and if u is ε-far from L_1, then each sample has probability at least ε to be a b, and thus we will find a b with constant probability. One can easily show that $1/\varepsilon$ is a lower bound on the number of samples to get a property tester for L_1; we say that L_1 is easy:

▶ **Definition 2.2.** *We say that L is easy if for small enough $\varepsilon > 0$, the optimal query complexity for a property tester for L is $\Theta(1/\varepsilon)$.*

Blocking factors

Extrapolating from the example L_1, let us introduce the notion of *blocking factors* (also known as killing words [17]): a word v is a blocking factor for L if it cannot appear as a factor of a word in L. For instance, b is a blocking factor for L_1. Note that bb and bbb are also blocking factors, but b is a minimal blocking factor (there are no strict factors of b that are blocking factors). Blocking factors were introduced in the original work giving a property tester for all regular languages [5]. A key insight of our work is to focus on *minimal blocking factors*. One important although simple property we will use is that if L is a regular language, then the set of minimal blocking factors of L is also a regular language.

It turns out that all property testers will be based on extensions of this very simple idea: we sample a number of positions in the word looking for blocking factors and answer no if we find a blocking factor, and yes otherwise. To be more precise, the analysis above for L_1 rests on the following property:

If u is ε-far from L_1, then it contains at least εn disjoint minimal blocking factors.

We will show later that this property can be extended to all regular languages.

Trivial languages

At this point we can revisit the class of trivial languages identified in [5]:

▶ **Definition 2.3.** *We say that L is trivial if for all small enough $\varepsilon > 0$, there exists a property tester for L that makes 0 queries for all large enough n.*

An example of a trivial language is L_2 consisting of words containing at least one a over the alphabet $\{a, b\}$. For any word u, replacing any letter by a yields a word in L_2, so $d(u, L_2) \leq 1$. A trivial property tester for L_2 simply answers yes all the time. One of our contributions in this work is a characterization of the trivial languages identified by Alon et al. [5].

▶ **Lemma 2.4.** *A regular language L is trivial if and only if it has no (minimal) blocking factors.*

Period and positional words

Let us now consider the language $L_3 = (ab)^*$ consisting of words of the form $ab \cdot ab \cdot ab \cdots ab$ over the alphabet $\{a, b\}$. Generalizing the ideas used for the analysis of L_1, a very simple property tester for L_3 goes as follows: given a word of length n, sample $1/\varepsilon$ pairs of letters from a random *even* position and answer no if we find anything else than ab, and yes otherwise. There are two new difficulties: we need to consider factors of length 2, and we want them to start at even positions. The arguments above are naturally extended to prove that this property tester has query complexity $\mathcal{O}(1/\varepsilon)$, and that this is asymptotically tight.

What this example shows is that instead of consider words we will need to consider *positional words*, which additionally encode information about the position. In the case of L_3, we need to distinguish between even and odd positions, so the word $abab$ is better represented as $(0, a)(1, b)(0, a)(1, b)$, where the first index denotes the parity of the position. More generally, we can associate to each regular language a period, and work with positional words encoding the position modulo this period. The notion of blocking factors is naturally extended to positional words, for instance $(0, a)(1, a)$ is a blocking factor, but $(1, b)(0, a)$ is not.

Almost characterizing easy languages

Generalizing the ideas presented above, one can prove the following lemma:

▶ **Lemma 2.5.** *Let L be a regular language. If there are finitely many minimal blocking factors for L, then L is easy.*

Indeed, in that case there is an upper bound ℓ on the length of minimal blocking factors, which depends only on L. We construct a property tester as follows: we sample $1/\varepsilon$ factors of length ℓ and answer no if we find a blocking factor, and yes otherwise. One can prove that this yields a property tester with query complexity $\mathcal{O}(1/\varepsilon)$. Unfortunately, this is not quite a characterization: the converse implication does not hold, let us explain why using another example.

Blocking sequences

Let us now consider the language L_4 consisting of words such that there are no c after a b over the alphabet $\{a, b, c\}$. The minimal blocking factors are of the form $ba^n c$ for $n \geq 0$, so there are infinitely many, the above argument above does not apply to this language. However, L_4 is easy: let us construct a property tester. We sample $1/\varepsilon$ letters at random and answer no if the sample contains a c after a b, and yes otherwise. To prove that this yields a property tester, we rely on the following property:

> If u is ε-far from L_4, then it can be decomposed $u = u_1 u_2$ where u_1 contains at least εn letters b and u_2 contains at least εn letters c.

What this example shows is that blocking factors are not enough: we need to consider sequences of factors, yielding the notion of blocking sequences. Intuitively, a blocking sequence for L is a sequence of (positional) words such that if the sequence appears as factors of some word u then $u \notin L$. For L_4, the minimal blocking sequence is (b, c).

Getting back to the almost characterization of easy languages sketched above, we will prove that L is easy if and only if there are finitely many minimal blocking sequences for L. The structure of the proof follows the original paper [5], considering first the case where L is recognised by a strongly connected automaton, and then extending it to the general case. Along the way, we will show that if L is recognised by a strongly connected automaton, then the characterization above holds: L is easy if and only there are finitely many minimal blocking factors for L. Introducing blocking sequences is necessary to deal with automata with more than one strongly connected component.

Hard languages

The remaining case is languages L which have infinitely many minimal blocking sequences. Let us illustrate this case on an example. We start from the parity language P consisting of words such that there is an even number of b's, over the alphabet $\{a, b\}$. If the goal would be to distinguish between $u \in P$ and $u \notin P$, any property tester would require scanning the whole input word. However, relaxing with the Hamming distance makes the question different: every word is at distance at most 1 from P by swapping at most one letter, so the language is trivial. Now, consider L_5 consisting of words such that inbetween each letter $\sharp$, there is an even number of b's, over the alphabet $\{a, b, \sharp\}$. Intuitively, L_5 encodes an arbitrary number of parity instances. Bathie and Starikovskaya [7] proved a lower bound of $\Omega(\log(\varepsilon^{-1})/\varepsilon)$ on the query complexity of (non-adaptive) property testers for L_5, matching the property testing algorithm they constructed for all regular languages.

▶ **Definition 2.6.** *We say that L is hard if for all small enough $\varepsilon > 0$, the optimal query complexity of a tester for L is $\Omega(\log(\varepsilon^{-1})/\varepsilon)$.*

Inspecting the minimal blocking sequences for L_5, we find infinitely many: this is no coincidence, we will extend Bathie and Starikovskaya's proof to show that any regular language with infinitely many minimal blocking sequences is hard.

The trichotomy theorem

Our main technical result is stated below. Recall that the case of finite languages is easy, so we focus on infinite languages.

▶ **Theorem 2.7.** *Let L be an infinite regular language, let us write $\mathsf{MBS}(L)$ for the set of minimal blocking sequences of L.*

- *L is trivial if and only if $\mathsf{MBS}(L)$ is empty;*
- *L is easy if and only if $\mathsf{MBS}(L)$ is finite and nonempty;*
- *L is hard if and only if $\mathsf{MBS}(L)$ is infinite.*

This trichotomy theorem closes a line of work on improving query complexity for property testers and identifying easier subclasses of regular languages. As mentioned above, the proof considers first the case where L is recognised by a strongly connected automaton, and then extends the results to the general case (following [5]):

- For the strongly connected case, we extend the ideas from [5] using the framework of minimal blocking factors, therereby simplifying the exposition, and obtain optimal property testers for trivial and easy languages, together with matching lower bounds. Our novel contributions here concern hard languages. First, we construct a property tester with query complexity $\Theta(\log(\varepsilon^{-1})/\varepsilon)$ for all regular languages recognised by a strongly connected automaton. This is an improvement over the similar result of Bathie and Starikovskaya [7], which works under the edit distance, while ours is designed for the Hamming distance. As the edit distance never exceeds the Hamming distance, the set of words that are ε-far with respect to the former is contained in the set of words ε-far for the latter. Therefore, an ε-tester for the Hamming distance is also an ε-tester for the edit distance, and our result supersedes and generalizes theirs in the case of strongly connected automata. Second, we prove a matching lower bound, again inspired by but strongly generalizing a result from Bathie and Starikovskaya [7], which was for a single language (L_5 discussed above), to all regular languages with infinitely many minimal blocking factors. We use Yao's minmax principle [21]: this involves constructing a hard distribution over inputs, and showing that any deterministic property testing algorithms cannot distinguish between positive and negative instances against this distribution.
- The general case follows a similar outline and builds upon the results in the strongly connected case. The notion of (minimal) blocking sequences enables smooth yet technical generalization of most of the results based on blocking factors for strongly connected automata. The main difficulty is the case of hard languages, and more precisely the lower bound. The complication here is that it is not enough to consider strongly connected components in isolation: there exists finite automata which contain a strongly connected component that induces a hard language, yet the language of the whole automaton is easy. The case where both are hard also occurs. Our proof defines a notion of "portals" which allows us to extract "crucial" connected components and show that hardness of these components imply hardness of the whole language.

The meta-question

Once the trichomoty theorem is established, the natural pending question is whether it can be made effective: the meta-question is then, given a regular language L, to determine whether it is trivial, easy, or hard. One could use this procedure to run the appropriate algorithm. On the positive front, our characterizations are indeed effective, in particular since for a regular language L, the set of minimal blocking sequences of L is another regular language, which can be effectively computed. However, we prove that the complexity of checking whether this set is empty, finite, or infinite (corresponding to the trichotomy), is PSPACE-complete.

Outline

The missing definitions are given in Section 3. We treat strongly connected automata in Section 4, and the general case in Section 5. The complexity of the meta-question is established in Section 6.

3 Preliminaries

Words and languages.

In this work, we consider a finite set Σ, the *alphabet*, whose elements are called *letters*. Words over Σ are finite sequences of letters, and Σ^* (resp. Σ^+) denotes the set of all words (resp. nonempty words) over Σ. A subset of Σ^* is called a language over Σ. The length of a word $u \in \Sigma^*$, denoted $|u|$, is the number of letters that it contains, and for $i \in [0, |u|-1]$, we use $u[i]$ to denote the i-th letter of u. Given two words $u, v \in \Sigma^*$, the *concatenation* $u \cdot v$ (more concisely denoted uv) of u and v is the word composed of the letters of u followed by the letters of v. This operation is associative, hence $(\Sigma^*, \cdot)$ is a monoid. Its unique neutral element, the empty word, is denoted γ.

Given a word $u \in \Sigma^*$ and two integers $0 \leq i, j \leq |u|-1$, define $u[i..j]$ as the word $u[i]u[i+1]\dots u[j]$ if $i \leq j$ and γ otherwise. Further, we let $u[i..j)$ denote the word $u[i..j-1]$. A word w is a *factor* (resp. *prefix*, *suffix*) of u is there exist indices i, j such that $w = u[i..j]$ (resp. with $i = 0, j = |u|-1$). We use $w \preccurlyeq u$ to denote "w is a factor of u". Furthermore, if w is a factor of u and $w \neq u$, we say that w is a *proper factor* of u.

Finite automata.

▶ **Definition 3.1** (Nondeterministic Finite automaton). *A* nondeterministic finite automaton *(NFA) $\mathcal{A}$ is a transition system defined by a tuple $(Q, \Sigma, \delta, q_0, F)$, with Q a finite set of states, Σ a finite alphabet, $\delta : Q \times \Sigma \to 2^Q$ the transition function, q_0 is the initial state and F is the set of final states.*

We say that $\mathcal{A}$ is deterministic (resp. complete) if $|\delta(q,a)| \leq 1$ (resp. ≥ 1) for all $q \in Q, a \in \Sigma$. We say that there is a transition from $p \in Q$ to $q \in Q$ labeled by $w \in \Sigma^*$, denoted $p \xrightarrow{w} q$, if there exists states $p_0 = p, p_1, \dots, p_{|w|} = q$ such that for every $i = 0, \dots, |w|-1$, $p_{i+1} \in \delta(p_i, w[i])$. In this case, we say that q is reachable from p and that p is co-reachable from q. The *language recognized by* $\mathcal{A}$, denoted $L(\mathcal{A})$, is the set of words that label a transition from the initial state to a final state, i.e.

$$L(\mathcal{A}) = \{w \in \Sigma^* \mid \exists q_f \in F : q_0 \xrightarrow{w} q_f\}.$$

We say that an NFA is *trim* if every state is reachable from the initial state and co-reachable from some final state. An NFA $\mathcal{A}$ can always be converted into a trim NFA that recognizes the same language by removing the states of $\mathcal{A}$ that are either not reachable from q_0 or not co-reachable from any final state.

Property testing.

▶ **Definition 3.2.** *Let L be a language, let u be a word of length n, let $\varepsilon > 0$ be a precision parameter and let $d : \Sigma^* \times \Sigma^* \to \mathbb{N} \cup \{+\infty\}$ be a metric. We say that the word u is* ε-far from L w.r.t. d *if $d(u, L) \geq \varepsilon n$, where*

$$d(u, L) := \inf_{v \in L} d(u, v).$$

Throughout this work and unless explicitly stated otherwise, we will consider the case where d is the Hamming distance, defined for two words u and v as the number of positions at which they differ if they have the same length, and as $+\infty$ otherwise.

Graphs and periodicity.

We now recall tools introduced by Alon et al [5] to deal with periodicity in finite automata.

The period $\lambda = \lambda(G)$ of a graph G is the greatest common divisor of the length of the cycles in G. If G is acyclic, we set $\lambda(G) = \infty$. Following the work of Alon et al [5], we will use the following property of directed graphs.

▶ **Fact 3.3** (From [5, Lemma 2.3]). *Let $G = (V, E)$ be a nonempty, strongly connected directed graph with finite period $\lambda = \lambda(G)$. Then there exists a partition $V = Q_0 \sqcup \ldots \sqcup Q_{\lambda-1}$ and a reachability constant $\rho = \rho(G)$ that does not exceed $3|V|^2$ such that:*

1. *For every $0 \leq i, j \leq \lambda - 1$ and for every $u \in Q_i, v \in Q_j$, the length of any directed path from u to v in G is equal to $(j - i) \mod \lambda$.*
2. *For every $0 \leq i, j \leq \lambda - 1$, for every $u \in Q_i, v \in Q_j$ and for every integer $r \geq \rho$, if $r = (j - i) \pmod{\lambda}$, then there exists a directed path from u to v in G of length r.*

The sets $(Q_i : i = 0, \ldots, \lambda - 1)$ are the *periodicity classes* of G. In what follows, we will slightly abuse notation and use Q_i for arbitrary non-negative integers i to mean $Q_{i \pmod{\lambda}}$ when $i \geq \lambda$.

Given a finite automaton $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$, we can naturally obtain the underlying directed graph by removing the labels from the transitions: it is the graph $G = (Q, E)$ where $E = \{(p, q) \in Q^2 \mid \exists a \in \Sigma : q \in \delta(p, a)\}$. In what follows, we naturally extend the notions of period[2], reachability constant and periodicity classes to finite automata through this graph G. The numbering of the periodicity classes is defined up to a shift mod λ: we say that Q_0 is the class that contains the initial state q_0. Similarly, we say that a finite automaton is strongly connected if the underlying graph is strongly connected.

A *strongly connected component* S of an automaton $\mathcal{A}$ is a maximal subset of states such that every state of S is reachable from every other one. Its *periodicity* is the periodicity of the subgraph induced by $\mathcal{A}$ over S.

Positional words and positional languages.

To motivate the following definitions, let us recall the example discussed in the overview. Consider a simple deterministic automaton with two states for the language $(ab)^*$: there is a transition labeled by a starting from one state but not from the other. The parity of the position of the factor ab in a word carries an important piece of information: if the position is odd, then we know that the word containing the factor is not in $(ab)^*$. Furthermore, while b appears at position 1 in ab, if this ab appears at an odd position in u then b appears at an even position in u. This leads to the definition of *positional words*.

▶ **Definition 3.4** (Positional words). *Let p be a positive integer. A p-*positional word *is a word over the alphabet $\mathbb{Z}/p\mathbb{Z} \times \Sigma$ of the form $(n \pmod{p}, a_0)((n+1) \pmod{p}, a_1) \cdots ((n+\ell) \pmod{p}, a_\ell)$. If $u = a_0 \cdots a_\ell$, we write $(n : u)$ to denote this word.*

[2] Note that in this context, an *aperiodic automaton* means an automaton with an aperiodic underlying graph, which is not the same thing as a counter-free automaton, which are sometimes called aperiodic automata.

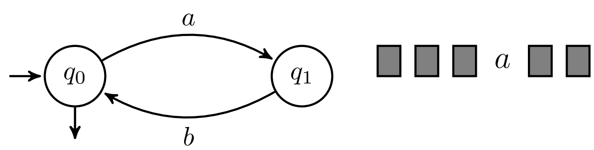

Figure 1 An automaton recognising the language $(ab)^*$. A witness that a word is not in this language is an a on an odd position or a b on an even position.

With this definition, starting with the 2-positional word $(0 : u)$, the factor ab at an odd position in u is $(1, a)(0, b)$, and the positional factor corresponding to the b is $(0, b)$. In this case, even when taking factors of factors of u, we still retain the (congruence classes of the) indices in the original word.

Any strongly connected finite automaton $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ can naturally be extended into an automaton over $\lambda(\mathcal{A})$-positional words as follows. Let $Q_0, \ldots, Q_{\lambda-1}$ be the partition of the states of $\mathcal{A}$ given by Fact 3.3, where $\lambda = \lambda(\mathcal{A})$ is the periodicity of $\mathcal{A}$. The *positional extension of* $\mathcal{A}$ is the finite automaton $\widehat{\mathcal{A}}$ defined by:

$$\widehat{\mathcal{A}} = (Q, \mathbb{Z}/\lambda\mathbb{Z} \times \Sigma, \delta', q_0, F) \qquad \text{where } \delta'(q, (i, a)) = \begin{cases} \delta(q, a) & \text{if } q \in Q_i, \\ \emptyset & \text{otherwise.} \end{cases}$$

By fact Fact 3.3, any transition from a state of Q_i goes to a state in Q_{i+1}, hence $\widehat{\mathcal{A}}$ recognized well-formed λ-positional words. We call the language recognized by $\widehat{\mathcal{A}}$ the *positional language of* $\mathcal{A}$, and denote it $\mathcal{TL}(\mathcal{A})$. This definition is motivated by the following property:

▶ **Property 3.5.** *For any word $u \in \Sigma^*$, we have $u \in \mathcal{L}(\mathcal{A})$ if and only if $(0 : u) \in \mathcal{TL}(\mathcal{A})$.*

For the reasons that we exposed earlier, positional words make it easier to manipulate factors with positional information, hence we phrase our property testing results in terms of positional languages. Notice that a property tester for $\mathcal{TL}(\mathcal{A})$ immediately gives a property tester for $\mathcal{L}(\mathcal{A})$, as one can simulate queries to $(0 : u)$ with queries to u by simply pairing the index of the query modulo $\lambda(\mathcal{A})$ with its result.

4 Strongly Connected NFAs

We first study the case of strongly connected NFAs, which are NFAs such that for any pair of states $p, q \in Q$, there exists a word w such that $p \xrightarrow{w} q$. We show that the query complexity of the language of such an NFA $\mathcal{A}$ can be characterized by the cardinality of the set of *minimal blocking factors* of $\mathcal{A}$, which are factor-minimal $\lambda(\mathcal{A})$-positional words that witness the fact that a word does not belong to $\mathcal{TL}(\mathcal{A})$. In this section, we consider a fixed NFA $\mathcal{A}$ and simply use "positional words" to refer to λ-positional words, where $\lambda = \lambda(\mathcal{A})$ is the period of $\mathcal{A}$.

▶ **Definition 4.1** (Blocking factors). *Let $\mathcal{A}$ be a strongly connected NFA. A positional word τ is a blocking factor of $\mathcal{A}$ if for any other positional word μ we have $\tau \preccurlyeq \mu \Rightarrow \mu \notin \mathcal{TL}(\mathcal{A})$.*

Further, we say that τ is a minimal blocking factor of $\mathcal{A}$ if no proper factor of τ is blocking a blocking factor of $\mathcal{A}$. We use $\mathsf{MBF}(\mathcal{A})$ to denote the set of all minimal blocking words of $\mathcal{A}$.

Intuitively and in terms of automata, $(i : u)$ is blocking for $\mathcal{A}$ if it does not label any transition in $\mathcal{A}$ labeled by u starting from a state of Q_i. This property is formally established later in Lemma 4.6.

The main result of this section is the following:

▶ **Theorem 4.2.** *Let L be an infinite language recognised by a strongly connected NFA $\mathcal{A}$.*
1. *L is hard if and only if $\mathsf{MBF}(\mathcal{A})$ is infinite.*
2. *L is easy if and only if $\mathsf{MBF}(\mathcal{A})$ is finite and nonempty.*
3. *L is trivial if and only if $\mathsf{MBF}(\mathcal{A})$ is empty.*

Our approach

The definition of blocking factors gives a simple but powerful framework to design property testers for $L(\mathcal{A})$: using random sampling, attempt to find a blocking factor in $(0 : u)$; if one is found, reject u, otherwise accept u. If $u \in L(\mathcal{A})$, then $(0 : u)$ does not contain a blocking factor, and we always accept u. The other case is where insight is required: one needs to find a sampling strategy that had a good probability of finding a blocking factor in $(0 : u)$ for any u ε-far from $L(\mathcal{A})$. A central tool for building such a sampling strategy is Lemma 4.8, which shows that any word ε-far from $L(\mathcal{A})$ contains many blocking factors. This approach, introduced by Alon et al. [5], is used by all existing property testing algorithms for regular languages.

This section is organized as follows. First, in Section 4.1, we give a few tools that help us deal with positional words and blocking factors in strongly connected NFA. Next, in Section 4.3 we tackle the case of trivial and easy languages (i.e. items Item 2 and Item 3 of Theorem 4.2). In Section 4.4, we prove that there exists an ε tester using $\mathcal{O}(\log(\varepsilon^{-1})/\varepsilon)$ queries for any language $\mathcal{L}(\mathcal{A})$. Finally, in Section 4.5, we show that any language not in the "easy" class requires $\Omega(\log(\varepsilon^{-1})/\varepsilon)$ queries, thereby proving that there is no intermediate query complexity class between easy and hard, and completing the trichotomy.

4.1 Positional words, blocking factors and strongly connected NFAs

Before diving into the proof of Theorem 4.2, we establish a few properties of positional words that will help us ensuring that we are creating well-formed positional words. In Section 4.2, we highlight the connection between property testing and blocking factors in strongly connected NFAs.

We start with the following facts, which are consequences of Fact 3.3.

▶ **Fact 4.3.** *Let n be a nonnegative integer, let w be a word of length n. If for some states $p \in Q_i, q \in Q_j$ of $\mathcal{A}$ we have $p \xrightarrow{w} q$, then the indices i, j satisfy the equation*

$$j - i = |w| \pmod{\lambda}$$

▶ **Fact 4.4.** *Let $\tau = (i : u)$ and $\mu = (j : v)$ be positional words. If $\tau \preccurlyeq \mu$, then there exists positional words η, η' with $|\eta| = i - j \pmod{\lambda}$ such that $\mu = \eta\tau\eta'$. In particular, this implies that there exists words w, w' with $|w| = i - j \pmod{\lambda}$ such that $v = wuw'$.*

The next property shows that chaining positional words in the automaton $\widehat{\mathcal{A}}$ results in well-formed positional words, in the sense that its letters are numbered by consecutive numbers modulo λ.

▶ **Property 4.5.** *Let p, q, r be states of $\widehat{\mathcal{A}}$ and let τ, μ be two positional words such that $p \xrightarrow{\tau} q$ and $q \xrightarrow{\mu} r$. Then $\tau\mu$ is a well-formed positional word, i.e. there exists a word w and an integer $i \in \mathbb{Z}/\lambda\mathbb{Z}$ such that $\tau\mu = (i : w)$.*

Proof. Let i, j be the respective indices of the periodicity classes of p and q, i.e. we have $p \in Q_i$ and $q \in Q_j$. Then there exist words u, v such that $\tau = (i : u)$ and $\mu = (j : v)$. Furthermore, by Fact 3.3, the length of any path from p to q is equal to $j - i \pmod{\lambda}$, hence the last letter of τ is $(j-1, a)$ for some $a \in \Sigma$ and the words can be chained correctly, i.e. $\tau\mu = (i : uv)$. ◀

These properties allows us to formalize the intuition we gave earlier about blocking factors.

▶ **Lemma 4.6.** *A positional word $\tau = (i : u)$ is a blocking factor for $\mathcal{A}$ iff for every states $p \in Q_i, q \in Q$, we have $p \overset{u}{\not\to} q$.*

Proof. We first show that if there exists states $p \in Q_i, q \in Q$ such that $p \xrightarrow{u} q$, then τ is not blocking, i.e. there exists $\mu \in \mathcal{TL}(\mathcal{A})$ such that $\tau \preccurlyeq \mu$. As $\mathcal{A}$ is strongly connected, there exist positional words η, η' such that $q_0 \xrightarrow{\eta} p$ and $q \xrightarrow{\eta'} q_f$ for some $q_f \in F$. By Property 4.5, the positional word $\mu = \eta\tau\eta'$ is well formed. Furthermore, it labels a transition from q_0 to q_f, hence it is in $\mathcal{TL}(\mathcal{A})$, and τ is not blocking.

For the converse, assume that τ is non-blocking: we show that there exists two states $p \in Q_i, q \in Q$ such that $p \xrightarrow{u} q$. As τ is non-blocking, there exists a positional word $\mu = (0 : w)$ such that $\tau \preccurlyeq \mu$ and there exists a final state r such that $q_0 \xrightarrow{\mu} r$, and equivalently, $q_0 \xrightarrow{w} r$. By Fact 4.4, since $\tau \preccurlyeq \mu$, there exists words v, v' such that $w = vuv'$ and the length of v is equal to i modulo λ. In particular, the path $q_0 \xrightarrow{w} r$ can be decomposed into $q_0 \xrightarrow{v} p \xrightarrow{u} q \xrightarrow{w} r$: in particular, we have $p \xrightarrow{u} q$. It only remains to show that p is in Q_i: this follows by Fact 4.3 since $|v| = i \pmod{\lambda}$. ◀

Finally, the Hamming distance between u and $\mathcal{L}(\mathcal{A})$ is the same as the distance between $(0 : u)$ and $\mathcal{TL}(\mathcal{A})$.

▷ Claim 4.7. For any word $u \in \Sigma^*$, we have $d(u, \mathcal{L}(\mathcal{A})) = d((0 : u), \mathcal{TL}(\mathcal{A}))$.

Proof. The $\leq$ part is straightforward. For the reverse inequality, if suffices to see that in any minimal substitution sequence from $(0 : u)$ to a positional word in $\mathcal{TL}(\mathcal{A})$, no operation changes only an index in an (index, letter) pair. ◁

This allows us to interchangeably use the statements "u is ε-far from $\mathcal{L}(\mathcal{A})$" and "$(0 : u)$ is ε-far from $\mathcal{TL}(\mathcal{A})$".

4.2 Strongly connected NFAs and blocking factors

Alon et al. [5, Lemma 2.6] first noticed that if a word u is ε-far from $L(\mathcal{A})$, then it contains $\Omega(\varepsilon n)$ short factors that witness the fact that u is not in $L(\mathcal{A})$. We start by translating the lemma of Alon et al. on "short witnesses" to the framework of blocking factors. More precisely, we show that if u is ε-far from $L(\mathcal{A})$, then $(0 : u)$ contains many disjoint blocking factors (Lemma 4.8).

▶ **Lemma 4.8.** *Let $\varepsilon > 0$, let u be a word of length $n \geq 6m^2/\varepsilon$ and assume that $L(\mathcal{A})$ contains at least one word of length n. If $\tau = (0 : u)$ is ε-far from $\mathcal{TL}(\mathcal{A})$, then τ contains at least $\varepsilon n/(6m^2)$ disjoint blocking factors.*

Proof. We build a set $\mathcal{P}$ of disjoint blocking factors of τ as follows: we process u from left to right, starting at index $i_1 = \rho$. Next, at iteration t, set j_t to be the smallest integer greater than or equal to i_t and smaller than $n - \rho$ such that $\tau[i_t..j_t]$ is a blocking factor. If there is no such integer, we stop the process. Otherwise, we add $\tau[i_t..j_t + \rho - 1])$ to the set $\mathcal{P}$, and iterate starting from the index $i_{t+1} = j_t + \rho$.

Let k denote the size of $\mathcal{P}$. We will show that we can substitute at most $3(k+1)m^2$ positions in τ to obtain a word in $\mathcal{TL}(\mathcal{A})$. (See Figure 2 for an illustration of this construction.) Using the assumption that τ is ε-far from $\mathcal{TL}(\mathcal{A})$ (which follows from Claim 4.7) will give us the desired bound on k.

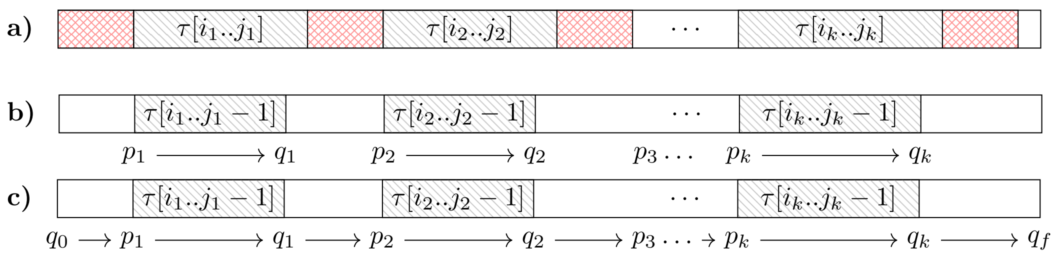

Figure 2 **a)** The decomposition process returns k factors $\tau[i_1, j_t], \dots, \tau[i_k, j_k]$ (represented as diagonally hatched in gray regions), separated together and with the start of the text by padding regions of $\rho - 1$ letters (red crosshatched regions). **b)** After removing the last letter, each previously blocking factor now labels a transition between some pair of states p_t, q_t. **c)** We use the padding regions to bridge between consecutive factors as well as the start and end of the word.

For every t, we chose j_t to be minimal so that $\tau[i_t..j_t]$ is blocking, hence $\tau[i_t..j_t - 1]$ is not blocking, and therefore $\tau[i_t..j_t - 1]$ labels a run from some $p_t \in Q_{i_t}$ to a $q_t \in Q_{j_t}$. Therefore, using the strong connectivity of $\mathcal{A}$ and Fact 3.3, we can edit the last ρ letters of the block $\tau[i_t..j_t + \rho - 1]$ to obtain a non-blocking factor that labels a transition from p_t to p_{t+1}. Using the ρ letters at the start and the end of the word, we add transitions from an initial state to p_1 and from q_k to a final state: the assumption that $\mathcal{L}(\mathcal{A})$ contains a word of length n ensures that Q_n contains a final state, hence this is always possible. The resulting word is in $\mathcal{TL}(\mathcal{A})$ and was obtained from τ using $(k+1)\rho \leq 3(k+1)m^2$ substitutions. As τ is ε-far from $\mathcal{TL}(\mathcal{A})$, we obtain the following bound on k:

$$\begin{aligned} 3(k+1)m^2 \geq \varepsilon n &\Rightarrow k + 1 \geq \frac{\varepsilon n}{3m^2} \\ &\Rightarrow k \geq \frac{\varepsilon n}{3m^2} - 1 \\ &\Rightarrow k \geq \frac{\varepsilon n}{6m^2} \end{aligned}$$

The last implication uses the assumption that $n \geq 6m^2/\varepsilon$. ◀

Lemma 4.8 allows us to handle three cases of Theorem 4.2, namely we use it to construct a tester with $\mathcal{O}(\log(\varepsilon^{-1})/\varepsilon)$ queries for any regular language, to construct a tester with $\mathcal{O}(1/\varepsilon)$ queries for regular languages with a finite number of blocking factors and to show the triviality of languages with no blocking factors.

4.3 The finite case

Using the framework described in the previous subsection, we show that when $\mathsf{MBF}(\mathcal{A})$ is finite then $\mathcal{L}(\mathcal{A})$ can the be tested with $\mathcal{O}(1/\varepsilon)$ queries, and furthermore if $\mathsf{MBF}(\mathcal{A})$ is empty, then $\mathcal{L}(\mathcal{A})$ is trivial.

Automata with no blocking factors.

First, observe that if $\mathsf{MBF}(\mathcal{A})$ is empty, there are no blocking factors, and no word can contain a blocking factor. Hence, the decomposition procedure used in the proof of Lemma 4.8 terminates with $k = 0$, which shows that, if $\mathcal{L}(\mathcal{A}) \cap \Sigma^n$ is nonempty, then any word of length n is at distance at most $3m^2$ of $\mathcal{L}(\mathcal{A})$. Therefore, for any $\varepsilon > 0$ and for $n \geq 3m^2/\varepsilon$, no word of length n is ε-far from $\mathcal{L}(\mathcal{A})$, and the tester that always accepts without queries is correct.

A tester for the finite-but-nonempty case.

To design a property tester with $\mathcal{O}(1/\varepsilon)$ queries, recall that, from Lemma 4.8, if u is ε-far from $L(\mathcal{A})$, then $(0 : u)$ contains many disjoint blocking factors. We then extract from each of these blocking factor a minimum blocking factor: because $\mathsf{MBF}(\mathcal{A})$ is finite, the length of each of these minimal factors is bounded by a constant C independent of u, hence a constant number of queries is enough to read one such factor. Finally, we show in Lemma 4.9 that sampling $\mathcal{O}(1/\varepsilon)$ factors is enough; the result follows.

▶ **Lemma 4.9.** *Let $\mathcal{A}$ be a trim strongly connected NFA. If $\mathsf{MBF}(\mathcal{A})$ is finite, then the language $L = L(\mathcal{A})$ can be tested with $\mathcal{O}(1/\varepsilon)$ queries.*

Proof. If $\mathsf{MBF}(\mathcal{A})$ if finite, then there exists a constant C such that every minimal blocking factor of $\mathcal{A}$ has length at most C.

Let m denote the number of states of $\mathcal{A}$. Given a word u of length n, we first check the following:

- If $n < 6m^2/\varepsilon$, read all of u, run the automaton $\mathcal{A}$ on u and accept if and only if $\mathcal{A}$ accepts.
- If $L(\mathcal{A})$ does not contain words of length n, reject. This can be checked efficiently using a simple dynamic programming algorithm.

The above procedure uses at most $\mathcal{O}(1/\varepsilon)$ queries, and if both checks fail, then u satisfies the hypotheses of Lemma 4.8. We then use the following procedure:

- sample independently $K = 6m^2 \ln(3)/\varepsilon$ random factors of length C in $(0 : u)$. To sample a factor, choose an index i uniformly at random in $\{1, \ldots, n\}$, and return $(i : u[i..i + C))$.
- rejects if at least one of these factors is blocking for $\mathcal{A}$.

We show that this algorithm is an ε-tester for L.

First, if $u \in L$, then no factor of $(0 : u)$ is blocking, and the algorithm accepts with probability 1.

Now, assume that u is ε-far from L. By Lemma 4.8, $(0 : u)$ contains at least $N = \varepsilon n/(6m^2)$ disjoint blocking factors. Each of these blocking factors induces at least one minimal blocking factor, i.e. $(0 : u)$ contains at least N disjoint minimal blocking factors. Each of these factors has length at most C, therefore the probability that the sampling procedure returns a factor that contains one of them is at least $N/n = \varepsilon/(6m^2)$. By repeating independently $K = 6m^2 \ln(3)/\varepsilon$ times, the probability of *not* finding any of the blocking factors is at most $(1 - N/n)^K \leq e^{-KN/n} = e^{-\ln 3} = 1/3$, therefore the algorithm rejects u with probability at least $2/3$ and satisfies Definition 2.1.

This tester uses $6Cm^2/\varepsilon = \mathcal{O}(1/\varepsilon)$ queries. ◀

Lower bound in the nonempty case.

It remains to show that if $\mathsf{MBF}(\mathcal{A})$ is nonempty, then testing $\mathcal{L}(\mathcal{A})$ requires $\Omega(1/\varepsilon)$ queries. Alon et al. [5] showed that "non-trivial" regular languages require $\Omega(1/\varepsilon)$ queries, using a notion of triviality defined differently from ours. They define non-trivial languages as follows:

▶ **Definition 4.10** ([5, Definition 3.1]). *A language L is non-trivial if there exists a constant $\varepsilon_0 > 0$, so that for infinitely many values of n the set $L \cap \Sigma^n$ is nonempty, and there exists a word $w \in \Sigma^n$ so that $d(w, L) \geq \varepsilon_0 n$.*

Their lower bound is the following:

▶ **Fact 4.11** ([5, Proposition 2]). *Let L be a non-trivial (in the sense of Alon et. al) regular language. Then for all sufficiently small $\varepsilon > 0$, any ε-tester for L requires $\Omega(1/\varepsilon)$ queries.*

To prove the lower bound in item 2) of Theorem 4.2, we show that if a language is non-trivial in our sense, i.e. $\mathsf{MBF}(\mathcal{A})$ is nonempty, then it is non-trivial in the sense of Alon et al.: we then get our lower bound by applying theirs.

▶ **Lemma 4.12.** *Let $\mathcal{A}$ be a strongly connected NFA such that $\mathsf{MBF}(\mathcal{A})$ is nonempty and denote $L = \mathcal{L}(\mathcal{A})$. Then there exists a constant $\varepsilon_0 > 0$ such that for infinitely many values of n the set $L \cap \Sigma^n$ is nonempty and there exists a word $w \in \Sigma^n$ so that $d(w, L) \geq \varepsilon_0 n$.*

Proof. As $\mathcal{A}$ is strongly connected, L is infinite, hence there are infinitely many integers n such that $L \cap \Sigma^n$ is nonempty. We show that there exists a constant ε_0 such that for large enough n such that $L \cap \Sigma^n$ is nonempty, there is a word of length n that is ε_0-far from L.

Since $\mathsf{MBF}(L)$ is nonempty, it contains at least one blocking factor, which is of the form $(i : u)$ for some $i \in \mathbb{Z}/\lambda\mathbb{Z}$. Let C denote the smallest multiple of λ greater than the length of u, let x denote an arbitrary word of length C with u as a prefix, and let $\varepsilon_0 = 1/(2C)$. We proceed to show that for any sufficiently large $n \geq 2(C + \lambda)$ such that $L \cap \Sigma^n$ is nonempty, there exists a word $w \in \Sigma^n$ such that $d(w, L) \geq \varepsilon_0 n$. We construct the word w by replacing a portion of v with disjoint copies of x, where x is an arbitrary word of length C that has u as a prefix. More precisely, we define w as $w = v[..i]x^k v[i + k \cdot C + 1..]$ where $k = \lceil \varepsilon_0 n \rceil$ disjoint copies of x. This word has length n as $i + k \cdot C \leq \lambda + C\lceil \varepsilon_0 n \rceil \leq n$.

We now claim that $d(w, L) \geq k \geq \varepsilon_0 n$. First, notice that as C is a multiple of λ, all k copies of x (and therefore of u) in w start at position equal to i modulo λ. Therefore, any such occurrence of u induces an occurrence of $(i : u)$ in $(0 : w)$. Next, consider a word w' obtained by performing less than k substitutions on w. Some copy of u in w' will be untouched, hence $(i : u) \preccurlyeq (0 : w')$, and therefore $w' \notin L$. Overall, we have

$$d(w, L) = d((0 : w), \mathcal{TL}(\mathcal{A})) \geq k \geq \varepsilon_0 n.$$

We have shown that there exists ε_0 such that for infinitely many n, $L \cap \Sigma^n$ is nonempty and there exists a word $w \in \Sigma^n$ so that $d(w, L) \geq \varepsilon_0 n$, hence L is non-trivial in the sense of Alon et al, and their lower bound applies. ◀

4.4 An efficient generic property tester for regular languages.

In this section, we show that for any strongly connected NFA $\mathcal{A}$, there exists ε-property tester for $L(\mathcal{A})$ that uses $\mathcal{O}(\log(\varepsilon^{-1})/\varepsilon)$ queries.

▶ **Theorem 4.13.** *Let $\mathcal{A}$ be a strongly connected NFA. For any $\varepsilon > 0$, there exists an ε-property tester for $L(\mathcal{A})$ that uses $\mathcal{O}(\log(\varepsilon^{-1})/\varepsilon)$ queries.*

Note that this result is an improvement over the similar result of Bathie and Starikovskaya [7]: while both testers use the same number of queries, theirs works under the edit distance, while ours is designed for the Hamming distance. As the edit distance never exceeds the Hamming distance, the set of words that are ε-far with respect to the former is contained in the set of words ε-far for the latter. Therefore, an ε-tester for the Hamming distance is also an ε-tester for the edit distance, and our result supersedes and generalizes theirs.

The algorithm for Theorem 4.13 is given in Algorithm 1. The procedure is fairly simple: the algorithm samples at random factors of various lengths in u, and rejects if and only if at least one of these factors is blocking. On the other hand, the correctness of the tester is far from trivial. The lengths and the number of factors of each lengths are chosen so that the number of queries is $\mathcal{O}(\log(\varepsilon^{-1})/\varepsilon)$ and the probability of finding a blocking factor is maximized, regardless of their repartition in u.

Algorithm 1 Generic ε-property tester using $\mathcal{O}(\log(\varepsilon^{-1})/\varepsilon)$ queries

```
function SAMPLE(u, ℓ)
    i ← UNIFORM(0, n − 1)
    l ← max(i − ℓ, 0), r ← min(i + ℓ, n − 1)
    η ← (l : u[l..r])
    return v
function TESTER(u, ε)
    β ← 12m²/ε
    if L(A) ∩ Σⁿ = ∅ then
        Reject
    else if n < β then
        Query all of u and run A on it
        Accept if and only if A accepts
    else
        F ← ∅
        T ← ⌈log(β)⌉
        for t = 0 to T do
            ℓ_t ← 2^t, r_t ← ⌈2 ln(3)β/ℓ_t⌉
            for i = 1 to r_t do
                F ← F ∪ {SAMPLE(u, ℓ_t)}
        Reject if and only if F contains a factor blocking for A.
```

We now turn to proving these properties formally.

▷ Claim 4.14. The tester given in Algorithm 1 makes $\mathcal{O}(\log(\varepsilon^{-1})/\varepsilon)$ queries to u.

Proof. If $n \le \beta$, then the tester makes $|u| \le \beta = \mathcal{O}(1/\varepsilon)$ queries, and the claim holds. The Sample function with parameter ℓ makes at most 2ℓ queries to u. Therefore, if $n \ge \beta$, the tester it makes at most $\ell_t \cdot r_t = \mathcal{O}(\beta)$ queries for every $t = 0, \dots, T$, which adds up to $\mathcal{O}(T \cdot \beta) = \mathcal{O}(\log(\varepsilon^{-1})/\varepsilon)$ queries. ◀

Next, we show an extension of Lemma 4.8 that shows that if u is ε-far from $L(\mathcal{A})$, then $(0 : u)$ contains $\Omega(\varepsilon n)$ blocking factors *of length* $\mathcal{O}(1/\varepsilon)$.

▶ **Lemma 4.15.** *Let $\varepsilon > 0$, let u be a word of length $n \ge 6m^2/\varepsilon$ and assume that $L(\mathcal{A})$ contains at least one word of length n. If u is ε-far from $L(\mathcal{A})$, then the positional word $(0 : u)$ contains at least $\varepsilon n/(12m^2)$ disjoint blocking factors of length at most $12m^2/\varepsilon$.*

Proof. Let $u, \mathcal{A}$ be a word and an automaton satisfying the above hypotheses. By Lemma 4.8, $(0 : u)$ contains at least $\varepsilon n/(6m^2)$ *disjoint* blocking factors. As these factors are disjoint, at most half of them (that is, $\varepsilon n/(12m^2)$ of them) can have length greater than $12m^2/\varepsilon$, as the sum of their lengths cannot exceed n. ◀

For the correctness analysis, we assume that u is ε-far from $L(\mathcal{A})$, and show that Algorithm 1 finds a blocking factor of $(0:u)$ with probability at least $2/3$.

▶ **Lemma 4.16.** *In the last* `Else` *block, if u is ε-far from $L(\mathcal{A})$, then Algorithm 1 rejects with probability at least $2/3$.*

Proof. Assume that u is ε-far from $L(\mathcal{A})$. As we are in the last `Else` block of Algorithm 1, $L(\mathcal{A}) \cap \Sigma^n$ is not empty (i.e. $L(\mathcal{A})$ contains a word of length n) and $n \geq \beta$, therefore the conditions of Lemma 4.15 are satisfied. Let $\mathcal{B}$ denote the set of minimal blocking factors in $(0:u)$ given by Lemma 4.15: we have $\mathcal{B} \geq n/\beta$. We conceptually divide the blocking factors in $\mathcal{B}$ into different categories depending on their length: for $t = 0, \ldots, T$, let B_t denote the subset of $\mathcal{B}$ of blocking factors of length at most $\ell_t = 2^t$. We then carefully analyze the probability that randomly sampled factors of length $2\ell_t$ contains a blocking factor from B_t, and show that over all t, at least one blocking factor is found with probability at least $2/3$.

▷ Claim 4.17. If in a call to Sample, the value i is such that there exists indices $l, r, l \leq i \leq r$, such that $(0:u)[l,r]$ is a blocking factor of $\mathcal{A}$ of length at most ℓ, then the factor η returned by the function is blocking for $\mathcal{A}$.

As the factors given by Lemma 4.15 are disjoint, the probability p_t that the factor returned by Sample is blocking is lower bounded by

$$p_t \geq \frac{1}{n} \sum_{\tau \in B_t} |\tau|$$

The Sample function is called $r_t = 2\ln(3)\beta/\ell_t$ times independently for each t, hence the probability p that the algorithm samples a blocking factor satisfies the following:

$$\begin{aligned}
(1-p) = \prod_{t=0}^{T}(1-p_t)^{r_t} &\leq \exp\left(-\sum_{t=0}^{T} p_t r_t\right) \\
&\leq \exp\left(-\frac{2\ln(3)\beta}{n}\sum_{t=0}^{T}\frac{1}{\ell_t}\sum_{\tau\in B_t}|\tau|\right) \\
&= \exp\left(-\frac{2\ln(3)\beta}{n}\sum_{\tau\in\mathcal{B}}|\tau|\sum_{t=\lceil\log|\tau|\rceil}^{T} 2^{-t}\right) \\
&\leq \exp\left(-\frac{2\ln(3)\beta}{n}\sum_{\tau\in\mathcal{B}}|\tau|\cdot 2^{-\lceil\log|\tau|\rceil}\right) \\
&\leq \exp\left(-\frac{2\ln(3)\beta}{n}\sum_{\tau\in\mathcal{B}}|\tau|\frac{1}{2|\tau|}\right) \\
&= \exp\left(-\frac{2\ln(3)\beta}{n}\cdot\frac{|\mathcal{B}|}{2}\right) \\
&\leq \exp\left(-\frac{2\ln(3)\beta}{n}\cdot\frac{n}{2\beta}\right) \\
&\leq \exp\left(-\ln(3)\right) = 1/3
\end{aligned}$$

It follows that $p \geq 2/3$, and Algorithm 1 satisfies Definition 2.1. ◀

4.5 Lower bound when there are infinitely many minimal blocking words

We now show that languages with infinitely many blocking factors are hard, i.e. any tester for such a language requires $\Omega(\log(\varepsilon^{-1})/\varepsilon)$ queries.

▶ **Theorem 4.18.** *Let $\mathcal{A}$ be a trim strongly connected automaton. If MBF($\mathcal{A}$) is infinite, then there exists a constant ε_0 such that for any $\varepsilon < \varepsilon_0$, any ε-property tester for $L = \mathcal{L}(\mathcal{A})$ uses $\Omega(\log(\varepsilon^{-1})/\varepsilon)$ queries.*

Our proof of this result will look familiar to readers acquainted with the lower bound of Bathie and Starikovskaya [7, Theorem 15]: our proof extends theirs to any language with arbitrarily long minimal blocking words. One difference is that our lower bound applies to ε-testers for the Hamming distance, instead of the edit distance. This is a weakening assumption as the edit distance never exceeds the Hamming distance, but it appears to be needed in the proof of Lemma 4.23.

Our proof is based on (a consequence of) Yao's minmax principle, which we recall here.

▶ **Fact 4.19** (From Yao's Minmax Principle [21]). *Let $f : \mathbb{R} \to \mathbb{N}$ be a nondecreasing function. Let $\mathcal{T}$ denote the set of all algorithms using less than $f(\varepsilon)$ queries, and let $\mathcal{T}_D$ denote the subset of deterministic algorithms. Let $\mathcal{D}$ be a probability distribution over Σ^*. Then, we have*

$$\inf_{T\in\mathcal{T}} \sup_{x\in\Sigma^*} \mathbb{P}_T(T \text{ errs on } x) \geq \inf_{T\in\mathcal{T}_D} \mathbb{P}_{x\sim\mathcal{D}}(T \text{ errs on } x).$$

Therefore, to show that any randomized algorithm with less than $\log(\varepsilon^{-1})/\varepsilon$ queries errs with large probability, it suffices to exhibit a probability distribution over inputs such that any *deterministic* tester errs with large probability on this distribution.

We will construct a hard distribution using long minimal blocking factors, and show that with large probability, any deterministic algorithm using less than $\log(\varepsilon^{-1})/\varepsilon$ queries has the same query results for many pairs of positive and ε-far instances. As the tester is deterministic, it must answer the same on all these pairs, and therefore make an error with large probability.

Our proof of Theorem 4.18 goes through the following steps:

1. first, show that with high probability, an input u sampled w.r.t. $\mathcal{D}$ is either in or ε-far from L (Lemma 4.23),
2. show that with high probability, any deterministic tester that makes fewer than $c \cdot \log(\varepsilon^{-1})/\varepsilon$ queries (for a suitable constant c) cannot distinguish whether the instance u is positive or ε-far,
3. combine the above to prove Theorem 4.18 via Fact 4.19.

4.5.1 Constructing a hard distribution

Let $\varepsilon > 0$ be sufficiently small and let n be a large enough integer. In what follows, m denotes the number of states of $\mathcal{A}$. To construct the hard distribution $\mathcal{D}$, we will use an infinite family of blocking factors that share a common structure, given by the following lemma.

▶ **Lemma 4.20.** *If MBF($\mathcal{A}$) is infinite, then there exist positional words ϕ, ν_+, ν_-, χ such that:*

1. *the words ν_+ and ν_- have the same length,*
2. *there exists a constant $S = 2^{\mathcal{O}(m)}$ such that $|\phi|, |\nu_+|, |\nu_-|, |\chi| \leq S$,*

3. *there exists an index $i_* \in \mathbb{Z}/\lambda\mathbb{Z}$ and a state $q_* \in Q_{i_*}$ such that for every integer $r \geq 1$, $\tau_{-,r} = \phi(\nu_-)^r z$ is blocking for $\mathcal{A}$, and for every $s < r$, we have*

$$q_* \xrightarrow{\tau_{+,r,s}} q_* \text{ where } \tau_{+,r,s} = \phi(\nu_-)^j \nu_+ (\nu_-)^{r-1-s}\chi.$$

In particular, $\tau_{+,r,s}$ is not blocking for $\mathcal{A}$.

The crucial property here is that $\tau_{-,r}$ and $\tau_{+,r,s}$ are very similar: they have the same length, differ in at most S letters, yet one of them is blocking and the other is not. The proof of this lemma is deferred to Appendix A.

We now use the words $\tau_{-,r}$ and $\tau_{+,r,s}$ and the constant S to describe how to sample an input $\mu = (0 : u)$ of length n w.r.t. $\mathcal{D}$.

Let π be a uniformly random bit. If $\pi = 1$, we will construct a positive instance $\mu \in \mathcal{TL}(\mathcal{A})$, and otherwise the instance will be ε-far from $\mathcal{TL}(\mathcal{A})$ with high probability. We divide the interval $[1..n]$ into $k = \varepsilon n$ intervals of length $\ell = 1/\varepsilon$, plus small initial and final segments μ_i and μ_f of length $\mathcal{O}(\rho)$ to be specified later. For the sake of simplicity, we assume that k and ℓ are integers and that λ divides ℓ. For $j = 1, \ldots, k$, let a_j, b_j denote the endpoints of the j-th interval. For each interval, we sample independently at random a variable τ_j with the following distribution:

$$\tau_j = \begin{cases} t, & \text{w.p. } p_t = 3 \cdot 2^t S\varepsilon / \log((S\varepsilon)^{-1}) \text{ for } t = 1, 2, \ldots, \log((S\varepsilon)^{-1}), \\ 0, & \text{w.p. } p_0 = 1 - \sum_{t=1}^{\log((S\varepsilon)^{-1})} p_t. \end{cases} \tag{3}$$

The event $\tau_j > 0$ means that the j-th interval is filled with with $N \approx 2^{-\tau_j}/\varepsilon$ "special" factors. When $\pi = 0$, these "special" factors will be minimal blocking factors $\tau_{-,r}$ for $r = 2^{\tau_j}$, whereas when $\pi = 1$, they will instead be similar non-blocking factors $\tau_{+,r,s}$ for a uniformly random s: they will be hard to distinguish with few queries. On the other hand, the event $\tau_j = 0$ means that the j-th interval contains no specific information. More precisely, we choose a positional word η_* of length ℓ such that $q_* \xrightarrow{w_*} q_*$: by Fact 3.3, this is possible as $\ell = 0 \pmod{\lambda}$. Then, if $\tau_j = 0$, we set $\mu[a_j..b_j] = \eta_*$, regardless of the value of π.

Formally, if $\tau_j > 0$, let $r = 2^{\tau_j}$, $N = 2^{-\tau_j}/(S\varepsilon)$ and let η be a word of length $\ell - N \cdot |\tau_{-,r}|$ such that $q_* \xrightarrow{\eta} q_*$: such a word exists as λ divides ℓ and $|\tau_{-,r}|$. We construct the j-th interval as follows:

- if $\pi = 0$, we set $\mu[a_j..b_j] = (\tau_{-,r})^N \eta$,
- if $\pi = 1$, we select $s \in [0..r-1]$ uniformly at random, and set $\mu[a_j..b_j] = (\tau_{+,r,s})^N \eta$.

Finally, the initial and final fragments μ_i and μ_f of μ are chosen to be the shortest words that label a transition from q_0 to q_* and q_* to a final state, respectively.

4.5.2 Properties of the distribution $\mathcal{D}$

We now conclude the proof of Theorem 4.2 by studying properties of the distribution $\mathcal{D}$.

▶ **Observation 4.21.** *If ε is small enough, $\mathcal{D}$ is well-defined, i.e. for every t between 0 and $\log((S\varepsilon)^{-1})$, we have $0 \leq p_t \leq 1$.*

▶ **Observation 4.22.** *If $\pi = 1$, then $\mu \in \mathcal{TL}(\mathcal{A})$.*

Next, we show that when $\pi = 0$, the resulting instance is ε-far from L with high probability.

▶ **Lemma 4.23.** *Conditioned on $\pi = 0$, the probability of the event $\mathcal{F} = \{\mu$ is ε-far from $\mathcal{TL}(\mathcal{A})\}$ goes to 1 as n goes to infinity.*

Proof. When $\pi = 0$, the procedure for sampling μ puts blocking factors of the form $(i_* : x)$ at positions equal to $i_* \mod \lambda$. Any word containing such a factor at such a position is not in $\mathcal{TL}(\mathcal{A})$, therefore any sequence of substitutions that transforms μ into a word of $\mathcal{TL}(\mathcal{A})$ must make at least one substitution in every such factor. Consequently, the distance between μ and $\mathcal{TL}(\mathcal{A})$ is at least the number of blocking factors in μ. To prove the lemma, we show that this number is at least εn with high probability, by showing that it is larger than εn by a constant factor in expectation and using a concentration argument.

Let B_j denote the number of blocking factors in the j-th interval: it is equal to $2^{-\tau_j}/(S\varepsilon)$ when $\tau_j > 0$ and to 0 otherwise.

▷ Claim 4.24. Let $B = \sum_{j=1}^{k} B_j$, and let $E = \mathbb{E}[B]$. We have $E \geq 2\varepsilon n$.

Claim proof. By direct calculation:

$$
\begin{aligned}
E &= \sum_{j=1}^{k} \mathbb{E}[B_j] && \text{by linearity}\\
&= \sum_{j=1}^{k} \sum_{t=1}^{\log(S/\varepsilon)} 2^{-t}/(S\varepsilon) \cdot p_t && \text{def. of expectation}\\
&= \sum_{j=1}^{k} \sum_{t=1}^{\log(S/\varepsilon)} 2^{-t}/(S\varepsilon) \cdot 3 \cdot 2^t \varepsilon S/\log(S/\varepsilon) && \text{def. of } p_t\\
&= \sum_{j=1}^{k} \sum_{t=1}^{\log(S/\varepsilon)} 3/\log(S/\varepsilon)\\
&= 3k \geq 2\varepsilon n
\end{aligned}
$$

◀

We will now show that $\mathbb{P}(B < \varepsilon n)$ goes to 0 as n goes to infinity. By Claim 4.24, we have $B < \varepsilon n \Rightarrow E - B \geq \varepsilon n$, and therefore $\mathbb{P}(B < \varepsilon n) \leq \mathbb{P}(E - B \geq \varepsilon n)$. The random variable B is the sum of k independent random variables, each taking values between 0 and $1/(S\varepsilon)$. Therefore, by Hoeffding's Inequality (Lemma B.1), we have

$$
\begin{aligned}
\mathbb{P}(E - B < \varepsilon n) &\leq \exp\left(-\frac{2\varepsilon^2 n^2}{k/(S\varepsilon)^2}\right)\\
&\leq \exp\left(-\frac{2S^2\varepsilon^4 n^2}{\varepsilon n}\right) \quad \text{as } k \leq \varepsilon n\\
&\leq \exp\left(-2S^2\varepsilon^3 n\right)
\end{aligned}
$$

This probability goes to 0 as n goes to infinity, which concludes the proof. ◀

▶ **Corollary 4.25.** *For large enough n, we have $\mathbb{P}(\mathcal{F}) \geq 5/12$.*

Intuitively, our distribution is hard to test because positive and negative instance are very similar. Therefore, a tester with few queries will likely not be able to tell them apart: the perfect completeness constraint forces the tester to accept in that case. Below, we prove this last part formally.

▶ **Lemma 4.26.** *Let T be a deterministic tester with perfect completeness (i.e. one sided error, always accepts $\tau \in \mathcal{TL}(\mathcal{A})$) and let q_j denote the number of queries that it makes in the j-th interval. Conditioned on the event $\mathcal{M} = \{\forall j \text{ s.t. } \tau_j > 0, q_j < 2^{\tau_j}\}$, the probability that T accepts u is 1.*

Proof. We show that if there exists a τ with non-zero probability w.r.t. $\mathcal{D}$ under $\mathcal{M}$ that T rejects, then there exists a word $\tau' \in \mathcal{TL}(\mathcal{A})$ that T rejects that also has non-zero probability, contradicting the fact that T has perfect completeness.

Let τ be the word rejected by T: as T has perfect completeness, hence $\tau \notin \mathcal{TL}(\mathcal{A})$, and there must be at least one interval with $\tau_j > 0$. Consider every interval j such that $\tau_j > 0$: it is of the form $(\tau_{-,r})^N v$ where $r = 2^{\tau_j}$ and $\tau_{-,r} = \phi(\nu_-)^r \chi$. Therefore, if $q_j < 2^{\tau_j}$, then there is a copy of ν_- that has not been queried by T across all copies of $\tau_{-,r}$. Consider the word τ' obtained by replacing this copy of ν_- with ν_+ in all N copies of $\tau_{-,r}$ in the block. The result block is of the form $(\tau_{+,r,s})^N v$ for some $s < r$, and by construction it is not blocking. Applying this operation to all blocks results in a word τ' that is in $\mathcal{TL}(\mathcal{A})$. Furthermore, τ' has non-zero probability under $\mathcal{D}$ conditioned on $\mathcal{M}$: it can be obtained by flipping the random bit π and choosing the right index s in every block. ◀

Next, we show that if a tester makes few queries, then the event $\mathcal{M}$ has large probability.

▶ **Lemma 4.27.** *Let T be a deterministic tester, let q_j denote the number of queries that it makes in the j-th interval, and assume that T makes at most $\frac{1}{72} \cdot \log(\varepsilon^{-1})/\varepsilon$ queries, i.e. $\sum_j q_j \leq \frac{1}{72} \cdot \log(\varepsilon^{-1})/\varepsilon$. The probability of the event $\mathcal{M} = \{\forall j \text{ s.t. } \tau_j > 0, q_j < 2^{\tau_j}\}$ is at least $11/12$.*

Proof. We show that the probability of $\overline{\mathcal{M}}$, the complement of $\mathcal{M}$, is at most $1/12$. We have:

$$
\begin{aligned}
\mathbb{P}\left(\overline{\mathcal{M}}\right) &= \mathbb{P}\left(\exists j : \tau_j > 0 \wedge q_j \geq 2^{\tau_j}\right) \\
&\leq \sum_j \mathbb{P}\left(\tau_j > 0 \wedge q_j \geq 2^{\tau_j}\right) && \text{by union bound} \\
&\leq \sum_j \sum_{t=1}^{\lfloor \log q_j \rfloor} p_t \\
&= \sum_j \sum_{t=1}^{\lfloor \log q_j \rfloor} \frac{3 \cdot 2^t \varepsilon}{\log(S/\varepsilon)} && \text{by def. of } p_t \\
&\leq \frac{3\varepsilon}{\log(S/\varepsilon)} \sum_j \sum_{t=1}^{\lfloor \log q_j \rfloor} 2^t \\
&\leq \frac{3\varepsilon}{\log(S/\varepsilon)} \sum_j 2q_j \\
&= \frac{3\varepsilon}{\log(S/\varepsilon)} \cdot \frac{2}{72} \cdot \frac{\log(1/\varepsilon)}{\varepsilon} \\
&\leq 1/12
\end{aligned}
$$

◀

We are now ready to prove Theorem 4.2.

Proof of Theorem 4.2. We want to show that any non-adaptive tester with perfect completeness for $L(\mathcal{A})$ requires at least $\frac{1}{72} \cdot \log(\varepsilon^{-1})/\varepsilon$ queries, by showing that any tester with fewer queries errs with probability at least $1/3$. We use Yao's minmax principle (Fact 4.19), and show that any **deterministic** non-adaptive algorithm T with perfect completeness that makes less than $\frac{1}{72} \cdot \log(\varepsilon^{-1})/\varepsilon$ queries errs on u when $(0 : u) \sim \mathcal{D}$ with probability at least $1/3$.

Consider such an algorithm T. The probability that T makes an error on u is lower-bounded by the probability that u is ε-far from $L(\mathcal{A})$ and T accepts, which in turn is larger than the probability of $\mathcal{M}\cap\mathcal{F}$. By Corollary 4.25, we have $\mathbb{P}(\mathcal{F}) \geq 5/12$, and by Lemma 4.27, $\mathbb{P}(\mathcal{M})$ is at least 11/12. Therefore, we have

$$\mathbb{P}(T \text{ errs}) \geq \mathbb{P}(\mathcal{M}\cap\mathcal{F}) \geq 1 - 7/12 - 1/12 = 1/3.$$

This concludes the proof of Theorem 4.2. ◀

5 The Case of General NFAs

In this section we extend the previous characterisation to all finite automata, proving our main theorem, stated as Theorem 2.7 in the overview section. To do so, we generalise the notion of blocking factor: we introduce *blocking sequences*, which are sequences of factors that witness the fact that we cannot take any path through the strongly connected components of the automaton. We define a suitable partial order on blocking sequences, which extends the factor relation on words to those sequences. The classification between trivial, easy and hard of a language can be characterised by the set of minimal blocking sequences of an automaton recognising it. This is expressed by the following theorem, where $\mathsf{MBS}(\mathcal{A})$ stands for the set of minimal blocking sequences of $\mathcal{A}$.

The statement we will prove is the following:

▶ **Theorem 5.1.** *Let L be an infinite language recognised by the trim NFA $\mathcal{A}$. The complexity of testing L is characterized by $\mathsf{MBS}(\mathcal{A})$ as follows:*

1. *L is hard to test if and only if $\mathsf{MBS}(\mathcal{A})$ is infinite.*
2. *L is easy to test if and only if $\mathsf{MBS}(\mathcal{A})$ is finite and nonempty.*
3. *L is trivial if and only if $\mathsf{MBS}(\mathcal{A})$ is empty.*

Recall that we only consider infinite languages in this classification.

> This section uses the knowledge package, to help the reader keep track of the various notions. Some *important terms* are coloured in red when we define them. Occurrences of those important terms are coloured in blue. The reader can click on those (or just hover over them on some PDF readers) to see the definition.

The rest of this section is dedicated to the proof of Theorem 5.1. Before we get into the proof, let us go through some examples, which illustrate some of the main difficulties. In all that follows we will abbreviate "strongly connected component" as SCC. We call an SCC trivial if it is just a single state with no self-loop.

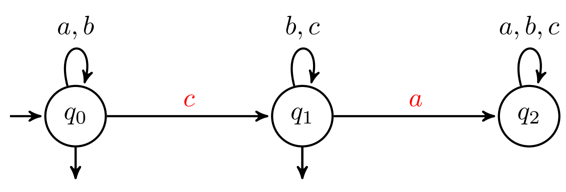

Figure 3 An automaton recognising the language $(a+b)^*(b+c)^*$.

▶ **Example 5.2.** Observe the automaton in Figure 3. It has two SCCs, plus a sink state. The set of minimal blocking factors of its language is infinite: it is the set cb^*a. Yet, it is easy: Given a word w, it suffices to sample $O(1/\varepsilon)$ positions at random and reject if we see a c appearing before an a. Clearly if the word is in the language, every c must be after every a, thus we accept. On the other hand, suppose the word is ε-far from the language. Let u be the maximal prefix of w containing less than $\varepsilon|w|/2$ occurrences of c. If $u = w$ then we can turn every c in w into an a to make it accepted, and thus $d(w, L) < \varepsilon|w|/2$, a contradiction. Hence we can write w as ucv. If v contains less than $\varepsilon|w|/2$ occurrences of a then $d(w, L) < \varepsilon|w|$, again a contradiction.

Otherwise, u contains $\varepsilon|w|/2$ occurrences of c and v contains $\varepsilon|w|/2$ occurrences of a. Then the probability that when picking $\varepsilon|w|$ letters at random we sample one of the c in u and one of the a in v is lower-bounded by a positive constant. In conclusion, we reject with constant probability when the word is ε-far from the language.

The crucial point in the following proof is the use of blocking *sequences* instead of blocking *factors*. A blocking sequence is a list of factors that are blocking for SCCs of the automaton, so that seeing this sequence as disjoint factors of a word guarantees that it is rejected. Blocking sequences come with a natural notion of minimality, which lets us characterise languages that are easy as those that admit finitely many minimal blocking sequences.

In the example above, a (unique) minimal blocking sequence is (c, a).

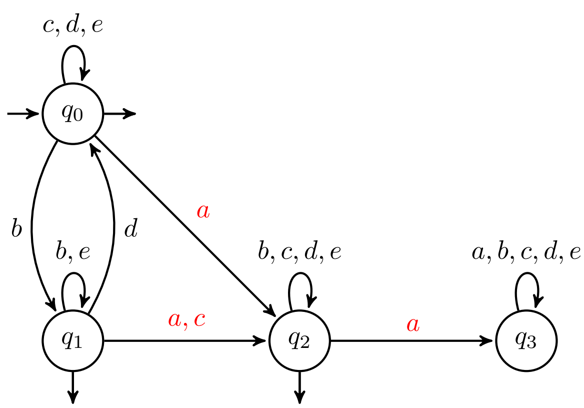

Figure 4 An automaton recognising the language $[\epsilon + ((c + d + e)^*b(b + e)^*d)^*a](b + c + d + e)^*$.

▶ **Example 5.3.** In Figure 4 we display an automaton with two SCCs and a sink state. The first SCC has blocking factors $be^*c + a$ and the second one just a. This automaton is easy: intuitively, a word that is ε-far from this language has to contain many a, as otherwise we can make it accepted by deleting all a, thanks to the second SCC. As a is also a blocking factor of the first SCC, we only need to look for two as in the word.

The family of unbounded blocking factors of the first SCC is made irrelevant by the fact that a word far from the language must contain many a anyway.

We fix an NFA $\mathcal{A} = (Q, \Sigma, \delta, q_{init}, q_f)$. Once again note that it has a single final state q_f. Let $\mathscr{S}$ be its set of SCCs. We define the partial order relation $\leq_{\mathcal{A}}$ on $\mathscr{S}$ as: $S \leq_{\mathcal{A}} T$ if and only if T is reachable from S. We write $<_{\mathcal{A}}$ for its strict part $\leq_{\mathcal{A}} \setminus \geq_{\mathcal{A}}$.

We define p as the least common multiple of the lengths of all simple cycles of $\mathcal{A}$. Given a number $k \in \{0, \ldots, p-1\}$, we say that a state t is k-reachable from a state s if there is a path from s to t of length k modulo p. In what follows, we use "positional words" for p-positional words with this value of p.

▶ Remark 5.4. In the rest of this section we will not try to optimise the constants in the formulas. They will, in fact, become quite large in some of the proofs. We make this choice to make the proofs more readable, although some of them are already technical.

For instance, the choice of p as the lcm of the lengths of simple cycles is not optimal: we could use, for instance, the lcm of the periodicities of the SCCs.

▶ **Definition 5.5.** *A portal is a 4-tuple $s, x \rightsquigarrow t, y \in (Q \times \{0, \ldots, p-1\})^2$, such that s and t are in the same SCC. It describes the first and last states visited by a path in an SCC, and the times at which it first and lasts visits that SCC (modulo p).*

The *positional language* of a portal is the set

$$\mathcal{PL}(s, x \rightsquigarrow t, y) = \{(x : w) \mid t \in \delta(s, w) \wedge x + |w| = y \pmod{p}\}.$$

Portals were already defined in [5], in a slightly different way. Our definition will allow us to express blocking sequences more naturally.

▶ **Definition 5.6.** *A positional word $(n : u)$ is blocking for a portal $s, x \rightsquigarrow t, y$ if it is not a factor of any word of $\mathcal{PL}(s, x \rightsquigarrow t, y)$. In other words, there is no path that starts in s and ends in t, of length $y - x$ modulo p, which reads u after $n - x$ steps modulo p.*

▶ Remark 5.7. There is an NFA with $\leq p|\mathcal{A}|$ states recognising $\mathcal{PL}(s, x \rightsquigarrow t, y)$: it simply simulates the SCC of s while keeping track of the number of letters read, plus x, modulo p. Its set of states is thus a subset of $\{0, \ldots, p-1\} \times Q$.

It is strongly connected: say we read a word u from (s, x) and reach (s', x'). There is a path from s' to s in $\mathcal{A}$, labelled by a word v. Hence we can reach (s, x) from (s', x') by reading $v(uv)^{p-1}$.

Its periodicity is p. Hence we can use all results we obtained on strongly connected NFAs on portals, with $p|\mathcal{A}|$ as the number of states and p as the periodicity.

▶ **Definition 5.8.** *An SCC-path π of $\mathcal{A}$ is a sequence of portals linked by transitions*

$$\pi = s_0, x_0 \rightsquigarrow t_0, y_0 \xrightarrow{a_1} s_1, x_1 \rightsquigarrow t_1, y_1 \cdots \xrightarrow{a_k} s_k, x_k \rightsquigarrow t_k, y_k,$$

such that for all $i \in \{1, \ldots, k\}$, $x_i = y_{i-1} + 1 \pmod{p}$, $s_i \in \delta(t_{i-1}, a_i)$, and $t_{i-1} <_{\mathcal{A}} s_i$.

It is a description of the states and times at which a path through the automaton enters and leaves the SCCs.

The language $\mathcal{L}(\pi)$ of an SCC-path $\pi = s_0, x_0 \rightsquigarrow t_0, y_0 \xrightarrow{a_1} \cdots s_k, x_k \rightsquigarrow t_k, y_k$ is the set

$$\mathcal{L}(\pi) = \mathcal{L}(s_0, x_0 \rightsquigarrow t_0, y_0) a_1 \mathcal{L}(s_1, x_1 \rightsquigarrow t_1, y_1) a_2 \cdots \mathcal{L}(s_k, x_k \rightsquigarrow t_k, y_k)$$

We say that π is *accepting* if $x_0 = 0$, $s_0 = q_{init}$, $t_k = q_f$ and $\mathcal{L}(\pi)$ is non-empty.

▶ **Fact 5.9.**

$$\mathcal{L}(\mathcal{A}) = \bigcup_{\pi \text{ accepting}} \mathcal{L}(\pi).$$

Proof. Let $w = b_1 \cdots b_\ell \in \mathcal{L}(\mathcal{A})$. There exists $\rho = q_0 \xrightarrow{b_1} q_1 \cdots \xrightarrow{b_\ell} q_\ell$ an accepting run in $\mathcal{A}$.

Let $i_1 < \ldots < i_k$ be the sequence of indices such that $\{i_1, \ldots, i_k\} = \{i \in \{1, \ldots, m\} \mid q_{i-1} <_{\mathcal{A}} q_i\}$. We also define $i_0 = 0$ and $i_{k+1} = \ell + 1$. In other words, those are the indices at which ρ enters a new SCC.

We define the SCC-path $\pi(\rho)$ as follows:

$$\pi(\rho) = q_0, 0 \rightsquigarrow q_{i_1-1}, y_0 \xrightarrow{a_{i_1}} q_{i_1}, x_1 \rightsquigarrow q_{i_2-1}, y_1 \cdots \xrightarrow{a_{i_k}} q_{i_k}, x_k \rightsquigarrow q_\ell, y_k$$

with $x_j = m + i_j \pmod p$ and $y_j = m + i_{j+1} - 1 \pmod p$ for all $j \in \{0, \ldots, k\}$. Clearly $w \in \mathcal{L}(\pi(\rho))$ and $\pi(\rho)$ is an accepting SCC-path.

Conversely, let $\pi = s_0, x_0 \rightsquigarrow t_0, y_0 \xrightarrow{a_1} \cdots s_k, x_k \rightsquigarrow t_k, y_k$ be an accepting SCC-path in $\mathcal{A}$ and let $w \in \mathcal{L}(\pi)$.

For all $j \in \{0, \ldots, k\}$, there is a word w_j labelling a path from s_j to t_j, such that $w = w_0 a_1 \cdots w_k$. By gluing those paths and the transitions $t_{j-1} \xrightarrow{a_j} s_j$, we obtain an accepting run for w in $\mathcal{A}$. ◀

Decomposing $\mathcal{A}$ as a union of SCC-paths allows us to use them as an intermediate step. We define blocking sequences for SCC-paths before defining them on automata.

▶ **Definition 5.10.** *We say that a sequence $((n_1 : u_1), \ldots, (n_\ell : u_\ell))$ of positional factors is blocking for an SCC-path $\pi = s_0, x_0 \rightsquigarrow t_0, y_0 \xrightarrow{a_1} \cdots s_k, x_k \rightsquigarrow t_k, y_k$ if there is a sequence of indices $i_0 \leq i_1 \leq \cdots \leq i_k$ such that $(n_{i_j} : u_{i_j})$ is blocking for $s_j, x_j \rightsquigarrow t_j, y_j$, for all j.*

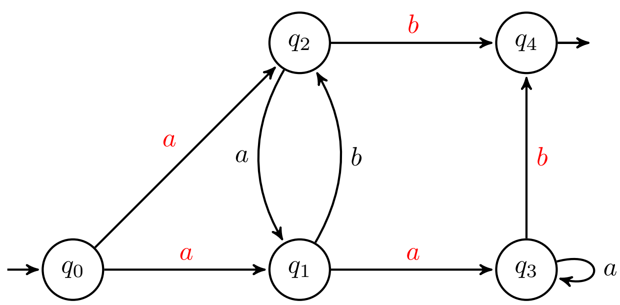

Figure 5 Automaton used for Example 5.11.

▶ **Example 5.11.** Take a look at the automaton displayed in Figure 5. It has four SCCs, including two trivial ones $\{q_0\}$ and $\{q_4\}$. The lcm of the lengths of its simple cycles is $p = 2$.

It has six accepting SCC-paths:

- $q_0, 0 \rightsquigarrow q_0, 0 \xrightarrow{a} q_1, 1 \rightsquigarrow q_1, 1 \xrightarrow{a} q_3, 0 \rightsquigarrow q_3, 0 \xrightarrow{b} q_4, 1 \rightsquigarrow q_4, 1$
- $q_0, 0 \rightsquigarrow q_0, 0 \xrightarrow{a} q_1, 1 \rightsquigarrow q_1, 1 \xrightarrow{a} q_3, 0 \rightsquigarrow q_3, 1 \xrightarrow{b} q_4, 0 \rightsquigarrow q_4, 0$
- $q_0, 0 \rightsquigarrow q_0, 0 \xrightarrow{a} q_2, 1 \rightsquigarrow q_1, 0 \xrightarrow{a} q_3, 1 \rightsquigarrow q_3, 0 \xrightarrow{b} q_4, 1 \rightsquigarrow q_4, 1$
- $q_0, 0 \rightsquigarrow q_0, 0 \xrightarrow{a} q_2, 1 \rightsquigarrow q_1, 0 \xrightarrow{a} q_3, 1 \rightsquigarrow q_3, 1 \xrightarrow{b} q_4, 0 \rightsquigarrow q_4, 0$
- $q_0, 0 \rightsquigarrow q_0, 0 \xrightarrow{a} q_1, 1 \rightsquigarrow q_2, 0 \xrightarrow{b} q_4, 1 \rightsquigarrow q_4, 1$
- $q_0, 0 \rightsquigarrow q_0, 0 \xrightarrow{a} q_2, 1 \rightsquigarrow q_2, 1 \xrightarrow{b} q_4, 0 \rightsquigarrow q_4, 0$

The language of the first SCC-path is $a(ba)^*a(a^2)^*b$. A blocking sequence for this SCC-path is $(0 : aa), (0 : b)$, which is in fact blocking for all those SCC-paths. Another one is $(1 : ab)$.

On the other hand, $(0 : ab)$ is not blocking for this path, as $(0 : ab)$ is not a blocking factor for the portal $q_1, 1 \rightsquigarrow q_1, 1$. It is, however, a blocking sequence for the third, fourth and last SCC-paths.

This is because if we enter the SCC $\{q_1, q_2\}$ through q_1, a factor ab can only appear after an even number of steps, while if we enter through q_2, it can only appear after an odd number of steps.

▶ **Example 5.12.** The automaton $\mathcal{A}$ displayed in Figure 6 only has cycles of length 1, hence $p = 1$. They are totally ordered by $\leq_{\mathcal{A}}$.

Observe that the sequence $((0 : a), (0 : b))$ is a blocking sequence for the SCC-path $\pi = q_0, 0 \rightsquigarrow q_0, 0 \xrightarrow{a} q_1, 0 \rightsquigarrow q_1, 0 \xrightarrow{a} q_2, 0 \rightsquigarrow q_2, 0$. Indeed, a is blocking for the first two portals, and b for the third. We can verify Lemma 5.15 here: If a word contains $|Q| = 4$ disjoint sequences $((0 : a), (0 : b))$, then in particular it must contain factors a, a and b in that order.

Even two blocking sequences would be enough here, but note that containing one blocking sequence is not enough: the word aba contains $((a : 0), (b : 0))$, yet it is in the language of π.

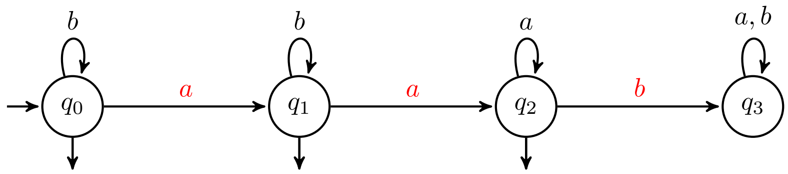

Figure 6 An automaton recognising the language $b^* + b^*ab^*a^*$.

In order to smoothen the proofs of the following results, let us start with two technical lemmas expressing two basic properties of the Hamming distance with respect to $\mathcal{A}$. The first one states that, for all SCC-path π and ℓ large enough, whether $\mathcal{L}(\pi)$ contains a word of length ℓ only depends on the value $\ell \pmod p$.

▶ **Lemma 5.13.** *Let $\pi = s_0, x_0 \rightsquigarrow t_0, y_0 \xrightarrow{a_1} \cdots s_k, x_k \rightsquigarrow t_k, y_k$ be an SCC-path and $r \in \{0, \ldots, p-1\}$. If there exists a word $w \in \mathcal{L}(\pi)$ with $|w| = r \pmod p$ and $|w| \geq |\mathcal{A}|$ then for all $\ell \in \mathbb{N}$ such that $\ell = r \pmod p$ and $\ell \geq p|\mathcal{A}| + 3|\mathcal{A}|^3$ there exists $w' \in \mathcal{L}(\pi)$ with $|w'| = \ell$.*

Proof. Suppose there exists $w \in \mathcal{L}(\pi)$ with $|w| = r \pmod p$ and $|w| \geq |\mathcal{A}|$. Then we can decompose it as $w = w_0 a_1 \cdots w_k$ with $w_i \in \mathcal{L}(s_i, x_i \rightsquigarrow t_i, y_i)$ for all i. For each $i \in \{0, \ldots, k\}$, let S_i be the SCC of s_i, and p_i the periodicity S_i. For all i such that the S_i is not trivial, by Fact 3.3, there is a word v_i' labelling a path from s_i to t_i of length ℓ_i with $\ell_i = m_{i+1} - m_i \pmod{p_i}$ and $\ell_i \leq 3|\mathcal{A}|^2$. If $i = k$, we set $m_{k+1} = r$. Since the SCC of s_i is not trivial, there is a simple cycle from s_i to itself. Let c_i be its length and u_i the word it reads. Since p_i divides p, we know that $m_{i+1} - m_i - \ell_i = r_i p_i \pmod p$ for some $r_i \in \{0, \ldots, p/p_i - 1\}$. The word $w_i' = u_i^{r_i} v_i'$ labels a path from s_i to t_i, of length $\ell_i + r_i p_i = m_{i+1} - m_i \pmod p$. Furthermore we have $|w_i'| \leq p + 3|\mathcal{A}|^2$. If s_i is in a trivial SCC, then w_i is the empty word γ. In that case we set $w_i' = \gamma$. We set $w' = w_1' a_1 w_2' \cdots a_{k-1} w_k'$. We have $w' \in \mathcal{L}(\pi)$, $|w'| \leq p|\mathcal{A}| + 3|\mathcal{A}|^3$ and $|w'| = r \pmod p$.

Since $w \in \mathcal{L}(\pi)$ and $|w| \geq |\mathcal{A}|$, the run reading w has to go through a cycle, hence there must be an i such that S_i is non-trivial. Let u be a word labelling a simple cycle from s_i to itself. Since $|u|$ divides p, for any $\ell \in \mathbb{N}$ such that $\ell = r \pmod{}$ and $\ell \geq p|\mathcal{A}| + 3|\mathcal{A}|^3$ we can find a word of length ℓ in $\mathcal{L}(\pi)$ by adding this cycle enough times in the run of w' constructed before. ◀

Our second technical lemma expresses that adding p letters to a word can only increase the distance by p.

▶ **Lemma 5.14.** *Let $s, x \rightsquigarrow t, y$ be a portal such that the SCC of s and t is non-trivial, and w a word such that $d(w, \mathcal{L}(s, x \rightsquigarrow t, y)) < +\infty$. Let $u \in \Sigma^p$. Then we have $d(wu, \mathcal{L}(s, x \rightsquigarrow t, y)) \leq d(w, \mathcal{L}(s, x \rightsquigarrow t, y)) + p$.*

Proof. As $d(w, \mathcal{L}(s, x \rightsquigarrow t, y)) < +\infty$, there exists $w' \in \mathcal{L}(s, x \rightsquigarrow t, y)$ such that $d(w, w') = d(w, \mathcal{L}(s, x \rightsquigarrow t, y))$. Thus there is a path of length $y - x \pmod p$ from s to t reading w'. As the SCC of t is non-trivial, there is a cycle from t to itself. Let v be a word labelling a simple cycle from t to itself. By definition of p, $|v|$ divides p, thus there exists k such that $k|v| = p$. In consequence, the word $w'v^k$ is in $\mathcal{L}(s, x \rightsquigarrow t, y)$. Furthermore, since $d(w, w') = d(w, \mathcal{L}(s, x \rightsquigarrow t, y))$, we have $d(wu, \mathcal{L}(s, x \rightsquigarrow t, y)) \leq d(wu, w'v^k) \leq d(w, \mathcal{L}(s, x \rightsquigarrow t, y)) + p$. ◀

We say that blocking sequences of a word are disjoint if they appear on disjoint sets of positions.

▶ **Lemma 5.15.** *If $(0 : w)$ contains $|Q|$ disjoint blocking sequences for an SCC-path $\pi = s_0, x_0 \rightsquigarrow t_0, y_0 \xrightarrow{a_1} \cdots s_k, x_k \rightsquigarrow t_k, y_k$, then $w \notin \mathcal{L}(\pi)$.*

Proof. We prove a slightly stronger statement by induction on k:

If $(m : w)$ contains k disjoint blocking sequences for an SCC-path $\pi = s_0, x_0 \rightsquigarrow t_0, y_0 \xrightarrow{a_1} \cdots s_k, x_k \rightsquigarrow t_k, y_k$, with $x_0 = m$, then no word of $\mathcal{L}(\pi)$ has w as a suffix.

The base case is trivial as the empty SCC-path has an empty language.

Now let $k > 0$ and suppose this proposition holds for $k - 1$. Consider an SCC-path $\pi = s_0, x_0 \rightsquigarrow t_0, y_0 \xrightarrow{a_1} \cdots s_k, x_k \rightsquigarrow t_k, y_k$ and disjoint blocking sequences $\sigma_1 \ldots, \sigma_k$. Let $(m : w) = (m : w_-)(m_v : v)(m_+ : w_+)$ with $(m_v : v)$ the first factor from one of the blocking sequences that is blocking for (m_1, s_1, t_1). Let σ_i be the blocking sequence in which it appears.

Since the blocking sequences are disjoint, for every blocking sequence other than σ, its part appearing in w_+ must be a blocking sequence for π, and thus also for $s_1, x_1 \rightsquigarrow t_1, y_1 \xrightarrow{a_2} \cdots s_k, x_k \rightsquigarrow t_k, y_k$. Hence w_+ contains $k - 1$ disjoint blocking sequences for $s_1, x_1 \rightsquigarrow t_1, y_1 \xrightarrow{a_2} \cdots s_k, x_k \rightsquigarrow t_k, y_k$. By induction hypothesis, no word of $\mathcal{L}(s_1, x_1 \rightsquigarrow t_1, y_1 \xrightarrow{a_2} \cdots s_k, x_k \rightsquigarrow t_k, y_k)$ has w_+ as a suffix. Let u be a word having w as a suffix. Suppose by contradiction that $u \in \mathcal{L}(\pi)$. Then $u = u_- a_1 u_+$ with $u_- \in \mathcal{L}((m_1, s_1, t_1))$ and $u_+ \in \mathcal{L}((m_2, s_2, t_2), \ldots, (m_k, s_k, t_k))$. Further, since w_+ is a suffix of w which is a suffix of u, we have $u = u_p w_+$ for some prefix u_p. Since w_+ cannot be a suffix of u_+, u_p must be a prefix of u_-, meaning that $(m_v : v)$ must appear as a factor of $(m : u_-)$. As $(m_v : v)$ is a blocking factor for (m_1, s_1, t_1), this contradicts the fact that u_- should be read entirely in the SCC of s_1. As a result, $u \notin \mathcal{L}(\pi)$.

This concludes our induction. ◀

The following lemma expresses a sort of converse implication: if a word is far from the language then it contains many blocking sequences. Let $B = p|\mathcal{A}| + 3|\mathcal{A}|^2$.

In the following results we will often use terms like "$(x : w)$ contains at least N_0 blocking factors for $s_0, x_0 \rightsquigarrow t_0, y_0$, ..., N_k blocking factors for $s_k, x_k \rightsquigarrow t_k, y_k$, in that order, all disjoint". This means that we can cut the word $(x : w)$ in k parts $(x : w) = (x_0 : w_0) \cdots (x_k : w_k)$, where for all i we have N_i disjoint blocking factors for $s_i, x_i \rightsquigarrow t_i, y_i$.

▶ **Lemma 5.16.** *Let $\pi = s_0, x_0 \rightsquigarrow t_0, y_0 \xrightarrow{a_1} \cdots s_k, x_k \rightsquigarrow t_k, y_k$ be an SCC-path. If $|w| \geq \max\left(\frac{6p^2|\mathcal{A}|^2}{\varepsilon}, (k+2)(B+p), \frac{(2k+4)p}{\varepsilon}\right)$ and $+\infty > d(w, \mathcal{L}(\pi)) \geq \varepsilon|w|$ then $(x_0 : w)$ contains at least $\frac{\varepsilon|w|}{12p^2|\mathcal{A}|^2(k+2)}$ blocking factors for $s_0, x_0 \rightsquigarrow t_0, y_0$, ..., $\frac{\varepsilon|w|}{12p^2|\mathcal{A}|^2(k+2)}$ blocking factors for $s_k, x_k \rightsquigarrow t_k, y_k$, in that order, all disjoint.*

Proof. We prove this by induction on k using Lemma 4.8. For $k = 0$ we can directly apply Lemma 4.8, in light of Remark 5.7.

Let $k > 0$, suppose the lemma holds for $k - 1$. Since $+\infty > d(w, \mathcal{L}(\pi))$, there is a word of length $|w|$ in $\mathcal{L}(\pi)$, hence we must have $|w| = y_k - x_0 \pmod p$.

Our goal is now to cut w in two parts with an intermediate letter, $w = w_- a w_+$, so that:

- $d(w_-, \mathcal{L}(s_0, x_0 \rightsquigarrow t_0, y_0)) \geq \frac{\varepsilon|w|}{2k+4}$, and we can apply Lemma 4.8
- $d(w_+, \mathcal{L}(s_1, x_1 \rightsquigarrow t_1, y_1 \xrightarrow{a_2} \cdots s_k, x_k \rightsquigarrow t_k, y_k)) \geq \frac{(k+1)\varepsilon|w|}{k+2}$ and we can apply the induction hypothesis

To do so, we use an intermediate value argument: We show that w has a short prefix which is very close to $\mathcal{L}(s_0, x_0 \rightsquigarrow t_0, y_0)$, and a large prefix which is far from it.

Then, we use Lemma 5.14, which says that extending a prefix with p letters can only change the distance to $\mathcal{L}(s_0, x_0 \rightsquigarrow t_0, y_0)$ by p. We then argue that there is an intermediate prefix w_- which is (roughly) at distance $\frac{\varepsilon|w|}{2k+4}$ from $\mathcal{L}(s_0, x_0 \rightsquigarrow t_0, y_0)$. We split w into $w_- a w_+$, with a a single letter. As $d(w, \mathcal{L}(\pi)) \geq \varepsilon|w|$, we infer that w_+ must be at distance at least $\varepsilon|w| - \frac{\varepsilon|w|}{2k+4}$ from $\mathcal{L}(s_1, x_1 \rightsquigarrow t_1, y_1 \xrightarrow{a_2} \cdots s_k, x_k \rightsquigarrow t_k, y_k)$, which suffices to conclude.

Let us now detail the proof. We define $\pi_+ = s_1, x_1 \rightsquigarrow t_1, y_1 \xrightarrow{a_2} \cdots s_k, x_k \rightsquigarrow t_k, y_k$.

▷ Claim 5.17. There is a prefix w' of w such that $|w'| \leq B + p$, $|w'| = y_0 - x_0 \pmod p$ and $d(w', \mathcal{L}(s_0, x_0 \rightsquigarrow t_0, y_0)) \leq B + p$.

Proof. Let w' be the prefix of w such that $|w'| = y_0 - x_0 \pmod p$ and $p|\mathcal{A}| + 3|\mathcal{A}|^2 \leq |w'| < B + p$. It exists as $|w| \geq (k+2)(B+p) \geq B+p$. By Lemma 5.13, $d(w_p, \mathcal{L}(s_0, x_0 \rightsquigarrow t_0, y_0)) < +\infty$ and therefore $d(w', \mathcal{L}(s_0, x_0 \rightsquigarrow t_0, y_0)) \leq |w_p| \leq B + p$. ◁

▷ Claim 5.18. There is a prefix w'' of w such that $|w''| > B + p$, $|w''| = y_0 - x_0 \pmod p$ and $d(w'', \mathcal{L}(s_0, x_0 \rightsquigarrow t_0, y_0)) \geq \varepsilon|w| - B - p - 1$.

Proof. Let $w = w'' a w_s$ such that $p|\mathcal{A}| + 3|\mathcal{A}|^2 \leq |w_s| \leq B + p$ and $|w_s| = y_k - x_1 \pmod p$. This decomposition exists as $|w| \geq (k+2)(B+p) \geq B+p$. We have $|w''| = y_0 - x_0 \pmod p$. Furthermore, as $B \leq |w_+|$, by Lemma 5.13, $d(w_+, \mathcal{L}(\pi_+)) < +\infty$. As a consequence, $d(w_+, \mathcal{L}(\pi_+) \leq |w_+| \leq B + p)$.

As $d(w_+, \mathcal{L}(\pi_+)) \leq |w_+| \leq B+p$ and $+\infty > d(w, \mathcal{L}(\pi)) \geq \varepsilon n$, we must have $d(w_-, \mathcal{L}(s_0, x_0 \rightsquigarrow t_0, y_0)) \geq \varepsilon n - B - p - 1$. ◁

▷ Claim 5.19. There exist words w_-, w_+ and a letter a such that $w = w_- a w_+$ and $d(w_-, \mathcal{L}(s_0, x_0 \rightsquigarrow t_0, y_0)) \geq \frac{\varepsilon|w|}{2k+4}$ and $d(w_+, \mathcal{L}(\pi_+)) \geq \frac{(k+1)\varepsilon|w|}{k+2}$.

Proof. By the two previous claim, w has a prefix w' of length $\geq B$ at distance $\leq B + p$ from $\mathcal{L}(s_0, x_0 \rightsquigarrow t_0, y_0)$, and a longer prefix w'' at distance $\geq \varepsilon|w| - B - p - 1$ from $\mathcal{L}(s_0, x_0 \rightsquigarrow t_0, y_0)$. Furthermore, as $|w| \geq 2\frac{B+p+1}{\varepsilon}$, we have $\varepsilon|w| - B - p - 1 \geq \frac{\varepsilon|w|}{k+2}$.

In consequence, there must exist w_- a prefix of w and u a word of length p such that $d(w_-, \mathcal{L}(s_0, x_0 \rightsquigarrow t_0, y_0)) < \frac{\varepsilon|w|}{k+2} \leq d(w_- u, \mathcal{L}(s_0, x_0 \rightsquigarrow t_0, y_0))$.

By Lemma 5.14, we have $d(w_-, \mathcal{L}(s_0, x_0 \rightsquigarrow t_0, y_0)) \geq \frac{\varepsilon|w|}{k+2} - p \geq \frac{\varepsilon|w|}{2k+4}$. As $|w| \geq \frac{(2k+4)p}{\varepsilon}$, we have $\frac{\varepsilon|w|}{k+2} - p \geq \frac{\varepsilon|w|}{2k+4}$ and thus $d(w_-, \mathcal{L}(s_0, x_0 \rightsquigarrow t_0, y_0)) \geq \frac{\varepsilon|w|}{2k+4}$.

On the other hand, as $d(w_-, \mathcal{L}(s_0, x_0 \rightsquigarrow t_0, y_0)) < \frac{\varepsilon|w|}{k+2}$ and $+\infty > d(w, \mathcal{L}(\pi)) \geq \varepsilon|w|$, we must have $d(w_+, \mathcal{L}(\pi_+)) \geq \frac{(k+1)\varepsilon|w|}{k+2}$. ◁

By the claim above and Lemma 4.8 we have that w_- contains at least $\frac{\varepsilon|w|}{12p^2|\mathcal{A}|^2(k+2)}$ blocking factors for $s_0, x_0 \rightsquigarrow t_0, y_0$. On the other hand, by induction hypothesis, w_+ contains at least $\frac{\varepsilon|w|}{12p^2|\mathcal{A}|^2(k+2)}$ blocking factors for $s_1, x_1 \rightsquigarrow t_1, y_1$, ..., $\frac{\varepsilon|w|}{12p^2|\mathcal{A}|^2(k+2)}$ blocking factors for $s_k, x_k \rightsquigarrow t_k, y_k$, in that order, all disjoint.

By combining the two, we obtain the lemma. ◀

A *blocking sequence* for $\mathcal{A}$ is a sequence $((n_1 : u_1), \ldots, (n_\ell : u_\ell))$ that is blocking for all SCC-paths of $\mathcal{A}$. As an example, observe that the sequences $(0 : ab), (1 : ab)$ and $(0 : aa), (0 : b)$ are both blocking for the automaton displayed in Figure 5 (see Example 5.11).

The goal of the next two lemmas is to show that we can reduce property testing of $\mathcal{L}(\mathcal{A})$ to a search for blocking sequences in the word:

- If we find a few blocking sequences in a word then we can answer no as it is not in the language (Lemma 5.20).
- A word that is far from the language contains many blocking sequences (Lemma 5.21). Hence if we do not find blocking sequences in the word then it is unlikely to be far from the language.

▶ **Lemma 5.20.** *If w contains $|\mathcal{A}|$ disjoint blocking sequences for $\mathcal{A}$ then $w \notin \mathcal{L}(\mathcal{A})$.*

Proof. Let π be an accepting SCC-path through $\mathcal{A}$. By definition a blocking sequence for $\mathcal{A}$ is a blocking sequence for π. As w contains $|Q|$ disjoint blocking sequences for $\mathcal{A}$, it contains $|Q|$ disjoint blocking sequences for π, hence $w \notin \mathcal{L}(\pi)$ by Lemma 5.15.

As a result, w is not in the language of any accepting SCC-path of $\mathcal{A}$, and thus not in $\mathcal{L}(\mathcal{A})$. ◀

Before going into the next proof, we start by observing that an SCC-path has at most $|\mathcal{A}|$ terms, and thus there are at most $(|\mathcal{A}|^2 p^2 |\Sigma| + 1)^{|\mathcal{A}|}$ SCC-paths in $\mathcal{A}$.

Let $C = (|\mathcal{A}|^2 p^2 |\Sigma| + 1)^{|\mathcal{A}|}$.

▶ **Lemma 5.21.** *If $+\infty > d(w, \mathcal{L}(\mathcal{A})) \geq \varepsilon |w|$ and $|w| \geq \max\left(\frac{6p^2}{\varepsilon}, (k+2)(B+p), \frac{(2k+4)p}{\varepsilon}\right)$ then w contains $\frac{\varepsilon|w|}{12C|\mathcal{A}|^3 p^2}$ disjoint blocking sequences for $\mathcal{A}$.*

Proof. For each accepting SCC-path π, as $\mathcal{L}(\pi) \subseteq \mathcal{L}(\mathcal{A})$, $d(w, \mathcal{L}(\pi)) \geq d(w, \mathcal{L}(\mathcal{A}))$. Thus, $\mathcal{A}$ must have $\frac{\varepsilon|w|}{12|\mathcal{A}|^2 p^2}$ disjoint blocking sequences for π, by Lemma 5.16. It remains to prove that $\frac{\varepsilon|w|}{12|\mathcal{A}|^2 p^2}$ disjoint blocking sequences for each π implies $\frac{\varepsilon|w|}{12C|\mathcal{A}|^3 p^2}$ disjoint blocking sequences for $\mathcal{A}$. Given a set of SCC-paths Π, we define $||\Pi||$ as the sum of the lengths of its elements. We say that a sequence is blocking for Π if it is blocking for all its elements.

We now prove the following statement by induction on $||\Pi||$: Let Π be a set of SCC-paths through $\mathcal{A}$, and let w be a word with $\frac{\varepsilon|w|}{12|\mathcal{A}|^2 p^2}$ disjoint blocking sequences for each $\pi \in \Pi$. Then w contains $\frac{\varepsilon|w|}{12||\Pi|||\mathcal{A}|^2 p^2}$ disjoint blocking sequences for Π.

The base case is immediate as w contains arbitrarily many disjoint occurrences of the empty word, which is a blocking sequence for $\emptyset$.

Let $w = w_- w_+$ where w_- is the minimal prefix of w containing $\frac{\varepsilon|w|}{12||\Pi|||\mathcal{A}|^2 p^2}$ disjoint blocking factors for the first element of some $\pi \in \Pi$. That is, $\pi = s_0, x_0 \rightsquigarrow t_0, y_0 \xrightarrow{a_1} \cdots s_k, x_k \rightsquigarrow t_k, y_k$ and w_- contains $\frac{\varepsilon|w|}{12||\Pi|||\mathcal{A}|^2 p^2}$ disjoint blocking factors for $s_0, x_0 \rightsquigarrow t_0, y_0$.

Then, by minimality of w_-, w_+ must have $\frac{(||\Pi||-1)\varepsilon|w|}{12||\Pi|||\mathcal{A}|^2 p^2}$ many disjoint blocking sequences for $\pi' = s_1, x_1 \rightsquigarrow t_1, y_1 \xrightarrow{a_1} \cdots s_k, x_k \rightsquigarrow t_k, y_k$ and for each $\pi'' \neq \pi$. We can then apply the induction hypothesis on w_+, with $\varepsilon' = \frac{(||\Pi||-1)\varepsilon}{||\Pi||}$ and $\Pi' = \Pi \setminus \{\pi\} \cup \{\pi'\}$: it must contain $\frac{\varepsilon'|w|}{12||\Pi'|||\mathcal{A}|^2 p^2} = \frac{\varepsilon|w|}{12||\Pi|||\mathcal{A}|^2 p^2}$ disjoint blocking sequences for Π'.

Appending a blocking factor for $s_0, x_0 \rightsquigarrow t_0, y_0$ in front of any of those blocking sequences for Π' yields a blocking sequence for Π. In consequence, we can form $\frac{\varepsilon|w|}{12||\Pi|||\mathcal{A}|^2 p^2}$ disjoint blocking sequences for Π by matching the $\frac{\varepsilon|w|}{12||\Pi|||\mathcal{A}|^2 p^2}$ blocking factors for $s_0, x_0 \rightsquigarrow t_0, y_0$ in w_- with the $\frac{\varepsilon|w|}{12||\Pi|||\mathcal{A}|^2 p^2}$ blocking sequences for Π' in w_+.

This concludes our induction. To obtain the lemma, we simply apply this property with Π the set of accepting SCC-paths of $\mathcal{A}$. ◀

We define a partial order $\trianglelefteq$ on sequences of positional factors. It is an extension of the factor order on blocking factors. It will let us define minimal blocking sequences, with which we characterise hard languages.

▶ **Definition 5.22.** *We have* $(n_1 : u_1), \ldots, (n_k : u_k) \trianglelefteq (n'_1 : u'_1), \ldots, (n'_\ell : u'_\ell)$ *when there exists a sequence of indices* $i_1 \leq i_2 \leq \ldots \leq i_k$ *such that* $(n_{i_j} : u_{i_j})$ *is a factor of* $(n'_j : u'_j)$ *for all* j*.*

A blocking sequence $(n_1 : u_1), \ldots, (n_k : u_k)$ *for* $\mathcal{A}$ *is minimal if it is minimal for* $\trianglelefteq$ *among blocking sequences of* $\mathcal{A}$*.*

▶ Remark 5.23. If $\sigma \trianglelefteq \sigma'$ and σ is a blocking sequence for an SCC-path π then σ' is also a blocking sequence for π.

The *left effect* of a sequence σ on an SCC-path $\pi = s_0, x_0 \rightsquigarrow t_0, y_0 \xrightarrow{a_1} \cdots s_k, x_k \rightsquigarrow t_k, y_k$ is the maximal index i such that the sequence is blocking for $s_0, x_0 \rightsquigarrow t_0, y_0 \xrightarrow{a_1} \cdots s_i, x_i \rightsquigarrow t_i, y_i$ (-1 if there is no such i). It is written $(\sigma \gg \pi)$. Similarly, the *right effect* of a sequence on π is the minimal index i such that the sequence is blocking for $(m_i, s_i), \ldots, (m_k, s_k)$ ($k+1$ if there is no such i). It is written $(\pi \ll \sigma)$.

▶ Remark 5.24. A sequence σ is blocking for an SCC-path $\pi = s_0, x_0 \rightsquigarrow t_0, y_0 \xrightarrow{a_1} \cdots s_k, x_k \rightsquigarrow t_k, y_k$ if and only if $(\sigma \gg \pi) = k$ if and only if $(\pi \ll \sigma) = 0$.

Also, given two sequences σ^l, σ^r, the sequence $\sigma^l \sigma^r$ is blocking for π if and only if $(\sigma^l \gg \pi) \geq (\pi \ll \sigma) - 1$.

We make the remark that minimal blocking sequences have a bounded number of terms. This is because if we build the sequence from left to right by adding terms one by one, the minimality implies that at each step the left effect on some SCC-path should increase. As the number and lengths of SCC-paths are bounded, so is the number of terms in the sequence.

▶ **Lemma 5.25.** *A minimal blocking sequence for* $\mathcal{A}$ *has at most* $|Q|(p|Q|)^{2|Q|}$ *terms.*

Proof. The number of SCC-paths in $\mathcal{A}$ is bounded by $(p|Q|)^{2|Q|}$, as each path has at most $|Q|$ portals and there are at most $p^2|Q|^2$ portals. Let $\sigma = (n_1 : u_1), \ldots, (n_\ell : u_\ell)$ be a minimal blocking sequence for $\mathcal{A}$. For all $i \in \{1, \ldots, \ell\}$ we write σ_i for $(n_1 : u_1), \ldots, (n_i : u_i)$.

For all $i \in \{1, \ldots, \ell - 1\}$ and SCC-path π, we have $(\sigma_i \gg \pi) \leq (\sigma_{i+1} \gg \pi)$. Furthermore, for all i there must exist π such that $(\sigma_i \gg \pi) < (\sigma_{i+1} \gg \pi)$: Otherwise we could remove $(n_{i+1} : u_{i+1})$ and the sequence would still be blocking for all SCC-paths of $\mathcal{A}$, contradicting the minimality of σ.

As there are at most $(p|Q|)^{2|Q|}$ SCC-paths, each of length at most $|Q|$, ℓ must be at most $|Q|(p|Q|)^{2|Q|}$. ◀

We now have all the tools to present the proof that languages recognised by automata with bounded minimal blocking sequences are exactly easy languages. Let us start with the easier direction.

▶ **Lemma 5.26.** *If* $\mathcal{A}$ *has finitely many minimal blocking sequences, then it is easy.*

Proof. As the length of minimal blocking sequences of $\mathcal{A}$ is bounded, so is the number of minimal blocking sequences. Let K be the bound on the length and P the bound on the number of minimal blocking sequences.

Let us first sketch the proof before detailing the formulas. We infer from the fact that there are boundedly many blocking sequences that if a word w is ε-far from the language of $\mathcal{A}$ then it must contain $O(\varepsilon|w|)$ many times the same minimal sequence σ.

Since each positional word in this sequence has length at most K, by sampling $O(\frac{1}{\varepsilon})$ factors of length K uniformly at random, we can show a positive constant lower bound on the probability to find σ. We can repeat this step to obtain a probability $> 1/2$ to find $|\mathcal{A}|$ times the sequence σ. This proves that $w \notin \mathcal{L}(\mathcal{A})$ by Lemma 5.20.

We now develop the formal proof, starting with a proof that a word that is ε-far from $\mathcal{L}(\mathcal{A})$ must contain many times some minimal blocking sequence σ. The next claim shows that having many sequences $\trianglelefteq$ -greater than a sequence σ implies having many occurrences of σ.

▷ Claim 5.27. Let $\sigma = (n_1 : u_1) \cdots (n_k : u_k)$ be a blocking sequence for $\mathcal{A}$ and let $M \in \mathbb{N}$. If a positional word $(m : w)$ contains M disjoint blocking sequences for $\mathcal{A}$ that are all greater or equal to σ for $\trianglelefteq$, then $(m : w)$ contains at least $\frac{M}{k}$ occurrences of $(n_1 : u_1)$, ..., $\frac{M}{k}$ occurrences of $(n_k : u_k)$, in that order, all disjoint.

Proof. We proceed by induction on k. If $k = 0$ the claim is immediate.

Let $k > 0$, suppose the claim holds for sequences of length $k-1$. Suppose $(m : w)$ contains M disjoint blocking sequences for $\mathcal{A}$ that are all greater or equal to σ for $\trianglelefteq$. We can assume without loss of generality that all those sequences have $(n_1 : u_1)$ as a factor of their first element: If a sequence $\sigma' = (n'_1 : u'_1) \cdots (n'_l : u'_l)$ is such that $\sigma \trianglelefteq \sigma'$ and $(n_1 : u_1)$ is not a factor of $(n'_1 : u'_1)$, then we must have $\sigma \trianglelefteq (n'_2 : u'_2) \cdots (n'_l : u'_l)$. Hence we can shorten the sequences of timed words until they all have $(n_1 : u_1)$ as a factor of their first term.

Let $(m : w_1)$ be the smallest prefix of w containing $\frac{M}{k}$ occurrences of $(n_1 : u_1)$. Let $(m' : w')$ be such that $(m : w) = (m : w_1)(m' : w')$. As $(m : w)$ contains M disjoint blocking sequences for $\mathcal{A}$ which all have $(n_1 : u_1)$ as a factor of their first term, we can find at least $M - \frac{M}{k}$ of them in $(m' : w')$. As they are all greater or equal to σ, they are also greater or equal to $(n_2 : u_2), \ldots, (n_k : u_k)$. By induction hypothesis, $(m' : w')$ contains at least $\frac{1}{k-1}(M - \frac{M}{k}) = \frac{M}{k}$ disjoint occurrences of $(n_2 : u_2)$, ..., $(n_k : u_k)$, in that order, all disjoint.

As a result, $(m : w)$ contains at least $\frac{M}{k}$ occurrences of σ. ◁

We can move on to the next step, which is to show that a word that is ε-far from $\mathcal{L}(\mathcal{A})$ contains many occurrences of some minimal blocking sequence σ.

Let $D = 12C|\mathcal{A}|^4(p|\mathcal{A}|)^{2|\mathcal{A}|}p^2P$.

▷ Claim 5.28. If $+\infty > d(w, \mathcal{L}(\mathcal{A})) \geq \varepsilon|w|$ and $|w| \geq \max\left(\frac{6p^2|\mathcal{A}|^2}{\varepsilon}, (k+2)(B+p), \frac{(2k+4)p}{\varepsilon}\right)$ then there exists a minimal blocking sequence $\sigma = (n_1 : u_1) \cdots (n_k : u_k)$ for $\mathcal{A}$ such that w contains $\frac{\varepsilon|w|}{D}$ occurrences of $(n_1 : u_1)$, ..., $\frac{\varepsilon|w|}{D}$ occurrences of $(n_k : u_k)$, in that order, all disjoint.

Proof. We start by applying Lemma 5.21. We obtain that w contains $\frac{\varepsilon|w|}{12C|\mathcal{A}|^3p^2}$ disjoint blocking sequences for $\mathcal{A}$.

Each one of those sequences if greater or equal to a minimal blocking sequence of $\mathcal{A}$ for $\trianglelefteq$. As a result, there exist σ a minimal blocking sequence and $\frac{\varepsilon|w|}{12C|\mathcal{A}|^3p^2P}$ disjoint blocking sequences in w that are all greater or equal to σ for $\trianglelefteq$. Furthermore, by Lemma 5.25, σ has at most $|\mathcal{A}|(p|\mathcal{A}|)^{2|\mathcal{A}|}$ terms.

We can then apply Claim 5.27 and obtain that w contains $\frac{\varepsilon|w|}{D}$ disjoint occurrences of each term of $(n_1 : u_1)$, ..., $(n_k : u_k)$, in that order, all disjoint. ◁

Given a word of length n, we start by checking that $\mathcal{A}$ accepts a word of length n. If not, we reject.

If $|w| \leq \max\left(\frac{6p^2|\mathcal{A}|^2}{\varepsilon}, (k+2)(B+p), \frac{(2k+4)p}{\varepsilon}\right)$ then we read w entirely, and accept iff it is in $\mathcal{L}(\mathcal{A})$.

Otherwise, for each minimal blocking sequence σ, we sample uniformly at random $\frac{D}{\varepsilon}$ intervals of length K in w. We reject if we find $|\mathcal{A}|$ disjoint occurrences of σ. If we have gone through every minimal blocking sequence without rejecting, we accept.

If the word is in $\mathcal{L}(\mathcal{A})$, then by Lemma 5.15 it cannot contain $|Q|$ disjoint blocking sequences, hence the algorithm will accept.

If the word is ε-far from $\mathcal{L}(\mathcal{A})$ (but within a finite distance), then by Claim 5.28 there exists a minimal blocking sequence $\sigma = (n_1 : u_1)\cdots(n_k : u_k)$ for $\mathcal{A}$ such that w contains $\frac{\varepsilon|w|}{D}$ occurrences of $(n_1 : u_1)$, ..., $\frac{\varepsilon|w|}{D}$ occurrences of $(n_k : u_k)$, in that order, all disjoint. Recall that by Lemma 5.25, σ has at most $T = |\mathcal{A}|(p|\mathcal{A}|)^{2|\mathcal{A}|}$ terms, hence $k \leq T$. By sampling $O(\frac{1}{\varepsilon})$ factors of length K at random, we have a constant positive lower bound on the probability of finding $|Q|$ of those occurrences of $(n_i : u_i)$, for any i. From this we infer that by sampling $O(\frac{1}{\varepsilon})$ factors of length K at random, we have a constant positive lower bound on the probability of finding $|\mathcal{A}|$ occurrences of $(n_i : u_i)$ for each i, and thus $|\mathcal{A}|$ occurrences of σ.

We can iterate this procedure a constant number of times to obtain a procedure using $O(\frac{1}{\varepsilon})$ samples that accepts every word in the language and rejects with probability $> 1/2$ words that are ε-far from the language. ◀

In order to prove a lower bound on the number of samples necessary to test a language with infinitely many minimal blocking sequences, we proceed as follows. We exhibit a portal with infinitely many minimal blocking factors $s, x \rightsquigarrow t, y$ and "isolate it" by constructing two sequences of timed factors σ^l and σ^r such that for all $(n' : u')$, $\sigma^l(n' : u')\sigma^r$ is blocking for $\mathcal{A}$ if and only if $(n' : u')$ is blocking for $s, x \rightsquigarrow t, y$. Then we reduce the problem of testing the language of this portal to the problem of testing the language of $\mathcal{A}$.

For the next proof we define a partial order on portals: $s, x \rightsquigarrow t, y \preceq s', x' \rightsquigarrow t', y'$ if all blocking factors of $s', x' \rightsquigarrow t', y'$ are also blocking factors of $s, x \rightsquigarrow t, y$. We write $\succeq$ for the reverse relation, $\simeq$ for the equivalence relation $\preceq \cap \succeq$ and $\not\simeq$ for the complement relation of $\simeq$.

Additionally, given an SCC-path $\pi = s_0, x_0 \rightsquigarrow t_0, y_0 \xrightarrow{a_1} \cdots s_k, x_k \rightsquigarrow t_k, y_k$ and two sequences of positional words σ^l, σ^r, we say that the portal $s_i, x_i \rightsquigarrow t_i, y_i$ *survives* (σ^l, σ^r) if $(\sigma^l \gg \pi) < i < (\pi \ll \sigma^r)$.

▶ **Definition 5.29.** *Let $s, x \rightsquigarrow t, y$ be a portal and σ^l and σ^r sequences of positional words. We define four properties that those objects may have:*

P1 *$\sigma^l\sigma^r$ is not blocking for $\mathcal{A}$*

P2 *$s, x \rightsquigarrow t, y$ has infinitely many minimal blocking factors*

P3 *for all accepting SCC-path π in $\mathcal{A}$, every portal in π which survives (σ^l, σ^r) is $\simeq$ -equivalent to $s, x \rightsquigarrow t, y$.*

▶ **Lemma 5.30.** *If $\mathcal{A}$ has infinitely many minimal blocking sequences, then there exist a portal $s, x \rightsquigarrow t, y$ and sequences σ^l and σ^r satisfying properties P1, P2 and P3.*

Proof. If $\mathcal{A}$ has infinitely many minimal blocking sequences, let $(\sigma_j)_{j\in\mathbb{N}}$ be a family of minimal blocking sequences such that the sum of the lengths of the elements of σ_j is at least j for all j.

By Lemma 5.25, a minimal blocking sequence has a bounded number of elements. We can thus extract from this sequence another one $(\sigma'_j)_{j\in\mathbb{N}}$ such that each σ'_j contains a factor of length at least j.

For each j let i_j be the index in σ'_j of a factor of length at least j, and l_j and r_j respectively the left effect of the $i_j - 1$ first factors and the right effect of the $k_j - i_j$ last ones, with

k_j the length of σ'_j. As those objects are taken from bounded sets, we can obtain a third sequence $(\bar{\sigma}_j)_{j\in\mathbb{N}}$ and α and K such that the ith element of each $\bar{\sigma}_j$ has length at least j and the set of components for which it is blocking is K.

For all j let (n_j, u_j) be the ith element of $\bar{\sigma}_j$. Define $\sigma^l = (n^l_1 : u^l_1), \dots, (n^l_k : u^l_k)$ and $\sigma^r = (n^r_1 : u^r_1), \dots, (n^r_\ell : u^r_\ell)$ so that $\bar{\sigma}_1 = \sigma^l(n_1, u_1)\sigma^r$. For all j, $\sigma^l(n_j : u_j)\sigma^r$ is a minimal blocking sequence.

We call *surviving portals* the portals that survive (σ^l, σ^r) in at least one SCC-path.

▷ Claim 5.31. There exists a surviving portal with infinitely many minimal blocking factors that is minimal for $\preceq$ among surviving portals.

Proof. Suppose by contradiction that all $\preceq$-minimal surviving portals have finitely many minimal blocking factors.

For all j, $(n_j : u_j)$ must be blocking for all surviving portals (otherwise $\overline{\sigma}_j$ would not be blocking for $\mathcal{A}$). Hence $(n_j : u_j)$ contains a blocking factor for each $\preceq$-minimal surviving portal. As those factors are bounded while $(n_j : u_j)$ can get arbitrarily large, there exists j such that $(n_j : u_j)$ can be split into two non-empty parts $(n_j : u_j^-)(n_j^+ : u_j^+)$ so that each $\preceq$-minimal surviving portal has a minimal blocking factor in either $(n_j : u_j^-)$ or $(n_j^+ : u_j^+)$. As a consequence, every surviving portal has a blocking factor in either $(n_j : u_j^-)$ or $(n_j^+ : u_j^+)$.

Let P be the number of portals of $\mathcal{A}$. We obtain that $\sigma^l[(n_j : u_j^-)(n_j^+ : u_j^+)]^P\sigma^r$ is a blocking sequence for $\mathcal{A}$, contradicting the minimality of $\sigma^l(n_j : u_j)\sigma^r$ for $\trianglelefteq$. In conclusion, there is a $\preceq$-minimal surviving portal with infinitely many minimal blocking factors. ◁

Let $s, x \rightsquigarrow t, y$ be a $\preceq$-minimal surviving portal with infinitely many minimal blocking factors: It satisfies P2.

The following claim shows that there is a pair of sequences (σ^l, σ^r) such that properties P1 and P3 are satisfied.

▷ Claim 5.32. There exist σ^l, σ^r such that $\sigma^l\sigma^r$ is not a blocking sequence for $\mathcal{A}$, and for all accepting SCC-path π in $\mathcal{A}$, every surviving portal in π is $\simeq$-equivalent to $s, x \rightsquigarrow t, y$.

Proof. We start from the sequences σ^l, σ^r defined before and extend them so that they have the desired property.

For each $s', x' \rightsquigarrow t', y' \not\simeq s, x \rightsquigarrow t, y$, since $s, x \rightsquigarrow t, y$ is $\preceq$-minimal we can pick a positional word $(n : u)_{s',x' \rightsquigarrow t',y'}$ that is blocking for $s', x' \rightsquigarrow t', y'$ but not for $s, x \rightsquigarrow t, y$.

We extend σ^l and σ^r as follows. While there is a surviving portal $s', x' \rightsquigarrow t', y'$ that is not $\simeq$-equivalent to $s, x \rightsquigarrow t, y$:

- We pick an SCC-path π such that $s', x' \rightsquigarrow t', y'$ survives in π.
- Let $i_\ell = (\sigma^l \gg \pi)$ and $i_r = (\pi \ll \sigma^r)$
- If for all $i \in \{i_\ell + 1, \dots, i_r - 1\}$, $s_i, x_i \rightsquigarrow t_i, y_i \not\simeq s, x \rightsquigarrow t, y$ then we append at the end of σ^l the sequence $(n : u)_{s_{i_\ell+1}, x_{i_\ell+1} \rightsquigarrow t_{i_\ell+1}, y_{i_\ell+1}}, \dots, (n : u)_{s_{i_r-1}, x_{i_r-1} \rightsquigarrow t_{i_r-1}, y_{i_r-1}}$. The sequence $\sigma^l\sigma^r$ is now blocking for π. On the other hand, since we did not add any blocking factor for $s, x \rightsquigarrow t, y$, there must still be a surviving portal that is $\simeq$-equivalent to it.
- If there is an $i \in \{i_\ell + 1, \dots, i_r - 1\}$ such that $s_i, x_i \rightsquigarrow t_i, y_i \simeq s, x \rightsquigarrow t, y$ then let c be the maximal index in $\{i_\ell + 1, \dots, i\}$ such that (m_c, s_c, t_c) is not equivalent to $s, x \rightsquigarrow t, y$ for $\simeq$, or i_ℓ if there is no such index. Symmetrically, let d the minimal index in $\{i, \dots, i_r - 1\}$ such that $(m_d, s_d, t_d) \not\simeq s, x \rightsquigarrow t, y$, or i_r if there is no such index. We append at the end of σ^l the sequence $(n : u)_{s_{i_\ell+1}, x_{i_\ell+1} \rightsquigarrow t_{i_\ell+1}, y_{i_\ell+1}}, \dots, (n : u)_{m_c, s_c, t_c}$. We append at the

beginning of σ^r the sequence $(n:u)_{s_d,x_d \rightsquigarrow t_d,y_d}, \dots, (n:u)_{s_{i_r-1},x_{i_r-1} \rightsquigarrow t_{i_r-1},y_{i_r-1}}$. Now all surviving portals in π are $\simeq$-equivalent to $s,x \rightsquigarrow t,y$, and $s_i,x_i \rightsquigarrow t_i,y_i$ still survives.

We iterate this step until all surviving portals are $\simeq$-equivalent to $s,x \rightsquigarrow t,y$. We made sure that at least one portal was still surviving after each step, hence in the end the sequence $\sigma^l\sigma^r$ is not blocking for $\mathcal{A}$. ◁

◀

▶ **Lemma 5.33.** *Let $\pi = s_0,x_0 \rightsquigarrow t_0,y_0 \xrightarrow{a_1} \cdots s_\ell,x_\ell \rightsquigarrow t_\ell,y_\ell$ be an accepting SCC-path, and let $i \in \{0,\dots,\ell\}$. Let $\sigma^l = (n^l_1 : u^l_1),\dots,(n^l_k : u^l_k)$ a sequence such that $(\sigma^l \gg \pi) < i$ and $N \in \mathbb{N}$.*

Then there is a word w^l of length at most $(3|\mathcal{A}|^3 + |\mathcal{A}|)(k+1) + N(2p^2+p)k|\mathcal{A}| + pN\sum_{i=1}^k |u^l_i|$ such that $|w^l| = x_i - x_0 \pmod p$, there is a run reading w^l from s_0 to s_i in $\mathcal{A}$, and $(x_0 : w)$ contains N times $(n^l_1 : u^l_1)$, ..., N times $(n^l_k : u^l_k)$ as disjoint factors, in that order.

Proof. We define w^l by induction on k. As π is accepting, by definition $\mathcal{L}(\pi) \neq \emptyset$, and thus for all $j \in \{0,\dots,\ell\}$ there exists a word of length $y_j - x_j \pmod p$ labelling a path from s_j to t_j. By Fact 3.3, there is such a word v_j of length at most $3|\mathcal{A}|^2$. As a result, for all $z \in \{0,\dots,\ell\}$ we can form a word $w_z = v_0a_1v_1\cdots a_z$, of length at most $3|\mathcal{A}|^3 + |\mathcal{A}|$, labelling a path of length $x_z \pmod p$ from q_{init} to s_z in $\mathcal{A}$. If $k = 0$, we can simply set $w^l = w_i$.

Let $k > 0$, suppose the lemma holds for $k-1$. Let $j = ((n_1 : u^l_1) \gg \pi)$. As $((n_1 : u^l_1) \gg \pi) \leq (\sigma^l \gg \pi) < i$, we have $j < i$. By definition, $(n_1 : u^l_1)$ is not blocking for $s_{j+1},x_{j+1} \rightsquigarrow t_{j+1},y_{j+1}$. As a consequence, there is a word v_j labelling a path from s_j to t_j such that $(x_j : v_j)$ has $(n_1 : u^l_1)$ as a factor. We can remove cycles of length $0 \pmod p$ in that path, before and after reading $(x_j : v_j)$, so we can assume that $|v_j| \leq |u^l_1| + 2p|\mathcal{A}|$. As s_j and t_j are in the same SCC, we can extend v_j into a word v'_j of length $\leq |v_j| + |\mathcal{A}| \leq |u^l_1| + (2p+1)|\mathcal{A}|$ that labels a cycle from s_j to itself.

Let $\sigma' = (n^l_2 : u^l_2),\dots,(n^l_k : u^l_k)$ and $\pi' = s_{j+1},x_{j+1} \rightsquigarrow t_{j+1},y_{j+1} \xrightarrow{a_{j+2}} \cdots s_\ell,x_\ell \rightsquigarrow t_\ell,y_\ell$. By definition, we have $(\sigma' \gg \pi') = (\sigma^l \gg \pi) < i$. By induction hypothesis, there is a word w' of length $\leq (3|\mathcal{A}|^3 + |\mathcal{A}|)k + N(2p^2+p)(k-1)|\mathcal{A}| + pN\sum_{i=1}^{k-1}|u_i|$ such that $|w'| = x_i - x_j \pmod p$, there is a run reading w' from s_j to s_i in $\mathcal{A}$, and $(x_j : w)$ contains N times $(n^l_2 : u^l_2)$, ..., N times $(n^l_k : u^l_k)$ as disjoint factors, in that order.

We set $w^l = w_j(v'_j)^{pN}w'$. This word has length $x_i \pmod p$, and at most $|w_j| + pN|v'_j| + |w'| \leq 3|\mathcal{A}|^3 + |\mathcal{A}| + pN(|u^l_1| + (2p+1)|\mathcal{A}|) + |w'| \leq (3|\mathcal{A}|^3 + |\mathcal{A}|)(k+1) + N(2p^2+p)k|\mathcal{A}| + pN\sum_{i=1}^k |u^l_i|$. It labels a path from s_0 to s_i, and contains N times $(n^l_1 : u^l_1)$, ..., N times $(n^l_k : u^l_k)$ as disjoint factors, in that order. ◀

▶ **Lemma 5.34.** *Let $\pi = s_0,x_0 \rightsquigarrow t_0,y_0 \xrightarrow{a_1} \cdots s_\ell,x_\ell \rightsquigarrow t_\ell,y_\ell$ be an accepting SCC-path, and let $i \in \{0,\dots,\ell\}$. Let $\sigma^r = (n^r_1 : u^r_1),\dots,(n^r_k : u^r_k)$ a sequence such that $(\pi \ll \sigma^r) > i$ and $N \in \mathbb{N}$.*

Then there is a word w^r of length at most $(3|\mathcal{A}|^3 + |\mathcal{A}|)(k+1) + N(2p^2+p)k|\mathcal{A}| + pN\sum_{i=1}^k |u^r_i|$ such that $|w^r| = y_\ell - y_i \pmod p$, there is a run reading w^r from t_i to t_ℓ in $\mathcal{A}$, and $(y_i : w^r)$ contains N times $(n^r_1 : u^r_1)$, ..., N times $(n^r_k : u^r_k)$ as disjoint factors, in that order.

Proof. By a symmetric proof to the one of the previous lemma. ◀

Given a sequence σ, define $||\sigma||$ as the sum of the lengths of the terms of σ.

▶ **Lemma 5.35.** *If there exist $s, x \rightsquigarrow t, y$ and σ^l, σ^r satisfying properties P1, P2 and P3 then $\mathcal{L}(\mathcal{A})$ is hard.*

Proof. A direct consequence of properties P1 and P3 is that for all $(n' : u')$, $\sigma^l(n' : u')\sigma^r$ is blocking for $\mathcal{A}$ if, and only if $(n' : u')$ is blocking for $s, x \rightsquigarrow t, y$.

The proof goes as follows: we show that we can turn an algorithm testing $\mathcal{L}(\mathcal{A})$ with $f(\varepsilon)$ samples into an algorithm testing $\mathcal{L}(s, x \rightsquigarrow t, y)$ with $f(\varepsilon/X)$ samples with X a constant. We then apply Theorem 4.18 to obtain the lower bound.

Consider an algorithm testing $\mathcal{L}(\mathcal{A})$ with $f(\varepsilon)$ samples for some function f. We describe an algorithm for testing $\mathcal{L}(s, x \rightsquigarrow t, y)$. Say we are given a threshold ε and a word v of length n. First of all we can apply Lemmas 5.33 and 5.34 to compute two words w^l and w^r of length at most $E + \varepsilon n F$ for some constants E and F such that we can read w^l from q_{init} to s and w^r from t to q_f and w_l contains each element of σ^l at least εn times and w_r contains each element of σ^r at least εn times. Let $w = w^l v w^r$. Suppose $|v| \geq \frac{6p^2|\mathcal{A}|^2}{\varepsilon}$ and $d(v, \mathcal{L}(s, x \rightsquigarrow t, y)) < +\infty$.

- If $v \in \mathcal{L}(\mathcal{A})$ then clearly $w \in \mathcal{L}(\mathcal{A})$.
- If $d(v, \mathcal{L}(s, x \rightsquigarrow t, y)) \geq \varepsilon n$ then by Lemma 4.8 (in light of Remark 5.7), $(x : v)$ contains at least $\frac{\varepsilon n}{6p^2|\mathcal{A}|^2}$ blocking factors for $s, x \rightsquigarrow t, y$. Then we have that w contains at least $\frac{\varepsilon n}{6p^2|\mathcal{A}|^2}$ disjoint blocking sequences for $\mathcal{A}$. As a result, $d(w, \mathcal{L}(\mathcal{A})) \geq \frac{\varepsilon n}{6p^2|\mathcal{A}|^2}$. We divide this by the length of w, which is at most $2E + 2F\varepsilon n + n$. We obtain that $d(w, \mathcal{L}(\mathcal{A})) \geq \frac{\varepsilon}{X}|w|$ for some constant X.

Let us now describe the algorithm for testing $\mathcal{L}(s, x \rightsquigarrow t, y)$.

- If $\mathcal{L}(s, x \rightsquigarrow t, y) \cap \Sigma^n = \emptyset$ then we reject.
- If $|v| < \frac{6p^2|\mathcal{A}|^2}{\varepsilon}$ then we read v entirely and check that it is in $\mathcal{L}(s, x \rightsquigarrow t, y)$.
- If $v \in \mathcal{L}(s, x \rightsquigarrow t, y)$ then we apply our algorithm for testing $\mathcal{L}(\mathcal{A})$ on $w = w^l v w^r$ with parameter $\varepsilon' = \frac{\varepsilon}{X}$.

The number of samples used on v is at most the number of samples needed on w, hence $f(\varepsilon/X)$. We obtain a procedure to test $\mathcal{L}(s, x \rightsquigarrow t, y)$ using $f(\varepsilon/X)$ samples.

By Theorem 4.18, $f(\varepsilon/X) = \Omega(log(\varepsilon^{-1})/\varepsilon)$, hence $f(\varepsilon) = \Omega(log(\varepsilon^{-1})/\varepsilon)$. This concludes our proof. ◀

▶ **Proposition 5.36.** *If $\mathcal{A}$ has infinitely many minimal blocking sequences, then $\mathcal{L}(\mathcal{A})$ is hard.*

Proof. We combine Lemmas 5.30 and 5.35. ◀

5.1 Trivial languages

We now characterise trivial languages, as defined in [5]. The definition given there is that a language is trivial if for all threshold $\varepsilon > 0$, above a certain length N, every word is at distance $\leq \varepsilon|w|$ or $+\infty$ from the language. Hence, on words of length more than N, we do not need to sample any letter: we just check if the language contains a word of length $|w|$. If not, we answer no. If yes, then we know that w is ε-close to the language and we can answer yes.

We present here some other characterisations of this set of languages. They are exactly the languages such that there is a bound B such that every word is at distance either $\leq B$ or $+\infty$ from the language.

They are also the languages that are either finite or described by an automaton with a blocking sequence.

▶ **Example 5.37.** A representative example of trivial language is $L_1 = a^*ba^*$, the set of words containing a b over $\{a, b\}$.

Given any word w, it is at distance at most 1 from L_1: it suffices to make the first letter a b to obtain a word of L_1.

In consequence, all words of length at least $\frac{1}{\varepsilon}$ are ε-close to the language, which allows us to simply answer yes without sampling anything. For words of length $< \frac{1}{\varepsilon}$, we simply read the word in full and check if it is in the language.

Now consider the language $L_2 = L_1 \cap (\{a, b\}^2)^*$. It is still trivial, but now we have to take into account the parity of the length of the input word: If $|w|$ is odd then $d(w, L_2) = +\infty$ and we can answer no. If $|w|$ is even then $d(w, L_2) \leq 1$ and we can answer yes as soon as $|w| \geq \frac{1}{\varepsilon}$.

▶ **Lemma 5.38.** *Let $\mathcal{A}$ a trim NFA. The following are equivalent:*

1. *There exists $\varepsilon_0 > 0$, such that for infinitely many n there exist words in $\mathcal{L}(\mathcal{A}) \cap \Sigma^n$ and there exists $w \in \Sigma^n$ such that $d(w, \mathcal{L}(\mathcal{A})) \geq \varepsilon_0 n$*
2. *There exists a family of words $(w_i)_{i\in\mathbb{N}}$ such that for all i, $i \leq d(w_i, \mathcal{L}(\mathcal{A})) < +\infty$*
3. *$\mathcal{L}(\mathcal{A})$ is infinite and $\mathcal{A}$ admits a blocking sequence.*
4. *$\mathcal{L}(\mathcal{A})$ is infinite and every portal appearing in an accepting SCC-path in $\mathcal{A}$ has a blocking factor.*

Proof.
- $1 \Rightarrow 2$ is immediate.
- $2 \Rightarrow 3$: For all i, $i \leq d(w_i, \mathcal{L}(\mathcal{A})) < +\infty$ implies that $|w_i| \geq i$ and that there exists a word $u_i \in \mathcal{L}(\mathcal{A})$ of length $|w_i|$.
  It remains to prove that $\mathcal{A}$ has a blocking sequence. We use Lemma 5.21. Fix an arbitrary ε, for instance $\varepsilon = 1/2$. Let i be such that $i \geq \max\left(\frac{6p^2}{\varepsilon}, (k+2)(B+p), \frac{(2k+4)p}{\varepsilon}\right)$ and $i > \frac{12C|\mathcal{A}|p^2}{\varepsilon}$.
  Then as $i \leq d(w_i, \mathcal{L}(\mathcal{A})) < +\infty$, we can apply Lemma 5.21 and obtain that w_i contains $\frac{\varepsilon|w_i|}{12C|\mathcal{A}|p^2} > 1$ blocking sequences for $\mathcal{A}$. In particular, $\mathcal{A}$ has a blocking sequence.
- $3 \Rightarrow 1$: Let $\sigma = (n_1 : u_1), \ldots, (n_k : u_k)$ be a blocking sequence for $\mathcal{A}$. As $\mathcal{A}$ is infinite, there exists an SCC-path π in $\mathcal{A}$ and $w \in \mathcal{L}(\pi)$ with $|w| \geq |\mathcal{A}|$. By Lemma 5.13, for all $\ell \geq p|\mathcal{A}| + 3|\mathcal{A}|^3$ such that $\ell = |w| \pmod p$ there exists $w' \in \mathcal{L}(\pi)$ with $|w'| = \ell$.
  For all $i \in \{1, \ldots, k\}$ we define v_i as a word of length $\leq u_i + 2p$ such that $(0 : v_i)$ has $(n_i : u_i)$ as a factor. For all $N \in \mathbb{N}$, we can then define the word $w_N = v_1^N \cdots v_k^N a^{|w|}$ with a an arbitrary letter. As it is of length $|w| \pmod p$, there is a word of the same length in $\mathcal{L}(\mathcal{A})$. On the other hand, it contains N disjoint occurrences of σ, which is a blocking sequence for $\mathcal{A}$. Let $\varepsilon_0 = \frac{1}{|u_1|+|u_2|+\cdots+|u_k|+2kp+|w|}$. We have $\varepsilon_0|w_N| \leq N \leq d(w_N, \mathcal{L}(\mathcal{A}))$.
- $3 \Rightarrow 4$: If $\mathcal{A}$ has a blocking sequence, then every portal in $\mathcal{A}$ appearing in an accepting SCC-path has to have a blocking factor in that sequence.
- $4 \Rightarrow 3$: If $\mathcal{L}(\mathcal{A})$ is infinite and every portal appearing in an accepting SCC-path in $\mathcal{A}$ has a blocking factor, then we can construct a blocking sequence for $\mathcal{A}$ as follows. Let P be the number of those portals in $\mathcal{A}$. Let σ be a sequence containing a blocking factor for each of those portals. The sequence σ^P is blocking for $\mathcal{A}$.

◀

This concludes the proof of Theorem 5.1.

6 Hardness of classifying

In the previous sections, we have shown that testing some regular languages (*easy* ones) that requires fewer queries than testing others (*hard* ones). Therefore, given the task of testing a word for membership in $\mathcal{L}(\mathcal{A})$, it is natural to first try to determine if the language of $\mathcal{A}$ is easy, and if this is the case, run the appropriate ε-tester, that uses fewer queries. In this section, we investigate the computational complexity of checking which class of the trichotomy the language of a given automaton belongs to. We formalize this question as the following decision problems:

We show that, unfortunately, our combinatorial characterization based on minimal blocking sequences does not lead to efficient algorithms: both problems are PSPACE-complete.

▶ **Theorem 6.1.** *The triviality and easiness problems are both PSPACE-complete, even for strongly connected NFAs.*

In the following section we show the PSPACE upper bounds on both problems (Propositions 6.8 and 6.9).

6.1 A PSPACE upper-bound on classifying automata

Let us first provide another characterisation of hard automata.

▶ **Lemma 6.2.** *Let $\pi = s_0, x_0 \rightsquigarrow t_0, y_0 \xrightarrow{a_1} \cdots s_\ell, x_\ell \rightsquigarrow t_\ell, y_\ell$ be an SCC-path, i an index, Π a set of SCC-paths and $(\sigma_{\pi'})_{\pi' \in \Pi}$ a family of sequences of positional words such that $(\sigma_{\pi'} \gg \pi') < i$ for all π'.*

There exists a sequence of positional words σ such that:

- $(\sigma \gg \pi) < i$
- $(\sigma_{\pi'} \gg \pi') \leq (\sigma \gg \pi')$ *for all* $\pi' \in \Pi$.

Proof. We prove this by induction on the sum of the lengths of the elements of Π. If Π is empty then we can set σ as the empty sequence.

If not, let π_{min} be such that the first term of $\sigma_{\pi_{min}}$ has the least left effect on π. Let $\sigma_{\pi_{min}} = (n_1 : u_1), \ldots, (n_k : u_k)$ and $\pi_{min} = s'_0, x'_0 \rightsquigarrow t'_0, y'_0 \xrightarrow{a_1} \cdots s'_\ell, x'_\ell \rightsquigarrow t'_\ell, y'_\ell$. Let $j = ((n_1 : u_1) \gg \pi_{min})$ and $r = ((n_1 : u_1) \gg \pi)$.

Let $\pi' = s'_{j+1}, x'_{j+1} \rightsquigarrow t'_{j+1}, y'_{j+1} \xrightarrow{a_1} \cdots s'_\ell, x'_\ell \rightsquigarrow t'_\ell, y'_\ell$. Define $\Pi' = \Pi \setminus \{\pi_{min}\} \cup \{\pi'\}$ if $j < \ell$ and $\Pi' = \Pi \setminus \{\pi_{min}\}$ otherwise. In the first case the sequence associated with π' is $\sigma_{\pi'} = (n_2 : u_2), \ldots, (n_k : u_k)$.

▷ Claim 6.3. For all $\overline{\pi} \in \Pi \setminus \{\pi_{min}\}$, we have $(\sigma_{\overline{\pi}} \gg \pi) = r + (\sigma_{\overline{\pi}} \gg s_{r+1}, x_{r+1} \rightsquigarrow t_{r+1}, y_{r+1} \xrightarrow{a_{r+2}} \cdots s_k, x_k \rightsquigarrow t_k, y_k)$

Proof. Since the first term of $\sigma_{\pi'}$ was the one with the least left effect on π, the first term of every other sequence has a left effect at least r on it.

Let $\overline{\pi} \in \Pi \setminus \{\pi_{min}\}$, let $\sigma_{\overline{\pi}} = (\overline{n}_1 : \overline{u}_1), \ldots, (\overline{n}_m : \overline{u}_m)$. Let $z = ((\overline{n}_1 : \overline{u}_1) \gg \pi)$. This means $(\overline{n}_1 : \overline{u}_1)$ is not a blocking factor for $s_{z+1}, x_{z+1} \rightsquigarrow t_{z+1}, y_{z+1}$.

We have $(\sigma_{\overline{\pi}} \gg \pi) = z + ((\overline{n}_2 : \overline{u}_2), \ldots, (\overline{n}_m : \overline{u}_m) \gg s_{z+1}, x_{z+1} \rightsquigarrow t_{z+1}, y_{z+1})$ and $(\sigma_{\overline{\pi}} \gg s_{r+1}, x_{r+1} \rightsquigarrow t_{r+1}, y_{r+1} \xrightarrow{a_{r+2}\cdots}) = z - r + ((\overline{n}_2 : \overline{u}_2), \ldots, (\overline{n}_m : \overline{u}_m) \gg s_{z+1}, x_{z+1} \rightsquigarrow t_{z+1}, y_{z+1}) = (\sigma_{\overline{\pi}} \gg \pi) - r$.

◁

As a consequence of this claim, we have that $(\sigma_{\overline{\pi}} \gg s_{r+1}, x_{r+1} \rightsquigarrow t_{r+1}, y_{r+1} \xrightarrow{a_{r+2}} \cdots s_k, x_k \rightsquigarrow t_k, y_k) < i - r$ for all $\overline{\pi} \in \Pi \setminus \{\pi'\}$.

By induction hypothesis, we obtain a sequence σ' such that

- $(\overline{\sigma} \gg s_{r+1}, x_{r+1} \rightsquigarrow t_{r+1}, y_{r+1} \xrightarrow{a_1} \cdots s_\ell, x_\ell \rightsquigarrow t_\ell, y_\ell) < i - r$
- $(\sigma_{\pi'} \gg \pi') \leq (\sigma' \gg \pi')$ for all $\pi' \in \Pi'$.

The sequence $(n_1 : u_1), \sigma'$ satisfies both conditions of the lemma. ◀

▶ **Lemma 6.4.** *An automaton $\mathcal{A}$ is hard if and only if there exists an accepting SCC-path π containing a portal $s, x \rightsquigarrow t, y$ such that:*

- *$s, x \rightsquigarrow t, y$ has infinitely many minimal blocking factors.*
- *For all accepting SCC-path π' there exist sequences σ^l, σ^r such that:*
  - *$s, x \rightsquigarrow t, y$ survives (σ^l, σ^r) in π*
  - *All portals surviving (σ^l, σ^r) in π' are $\simeq$ -equivalent to $s, x \rightsquigarrow t, y$*

Proof. Let us start with the left-to-right direction. If $\mathcal{A}$ is hard then by Lemma 5.26 it has infinitely many minimal blocking sequences. Then by Lemma 5.30 we have a portal $s, x \rightsquigarrow t, y$ and sequences σ^l, σ^r satisfying properties P1, P2 and P3.

By P1, $\sigma^l\sigma^r$ is not blocking for $\mathcal{A}$, thus there exists an SCC-path $\pi = s_0, x_0 \rightsquigarrow t_0, y_0 \xrightarrow{a_1} \cdots s_k, x_k \rightsquigarrow t_k, y_k$ and an index i such that $(\sigma^l \gg \pi) < i < (\pi \ll \sigma^r)$.

As a consequence, we have $s_i, x_i \rightsquigarrow t_i, y_i \simeq s, x \rightsquigarrow t, y$, by P3. We can assume without loss of generality that $s_i, x_i \rightsquigarrow t_i, y_i = s, x \rightsquigarrow t, y$. As a result, for all accepting SCC-path π' we have that $s, x \rightsquigarrow t, y$ survives (σ^l, σ^r) in π and all portals surviving (σ^l, σ^r) in π' are $\simeq$ -equivalent to $s, x \rightsquigarrow t, y$ (we use the same pair (σ^l, σ^r) for all π').

Let us now prove the other direction. Suppose we have π and $s, x \rightsquigarrow t, y$ satisfying the conditions of the lemma. We only need to construct two sequences σ^l, σ^r such that properties P1 and P3 are satisfied. The result follows by Lemma 5.35.

let Π be the set of accepting SCC-paths in $\mathcal{A}$. Consider families of sequences $(\sigma^l_{\pi'})_{\pi' \in \Pi}$ and $(\sigma^r_{\pi'})_{\pi' \in \Pi}$ such that for all $\pi' \in \Pi$:

- $s, x \rightsquigarrow t, y$ survives $(\sigma^l_{\pi'}, \sigma^r_{\pi'})$ in π
- All portals surviving $(\sigma^l_{\pi'}, \sigma^r_{\pi'})$ in π' are $\simeq$ -equivalent to $s, x \rightsquigarrow t, y$

Let i be the index of $s, x \rightsquigarrow t, y$ in π. By Lemma 6.2 we can build a sequence σ^l such that

- $(\sigma^l \gg \pi) < i$
- $(\sigma^l_{\pi'} \gg \pi') \leq (\sigma^l \gg \pi')$ for all $\pi' \in \Pi$.

Using a symmetric argument, we build a sequence σ^r such that

- $i < (\pi \ll \sigma^r)$
- $(\pi' \ll \sigma^r_{\pi'}) \geq (\pi' \ll \sigma^r)$ for all $\pi' \in \Pi$.

As a consequence, for all accepting SCC-path $\pi' \in \Pi$, all portals surviving (σ^l, σ^r) in π' are $\simeq$ -equivalent to $s, x \rightsquigarrow t, y$. Furthermore, $s, x \rightsquigarrow t, y$ survives (σ^l, σ^r) in π.

We have shown that $s, x \rightsquigarrow t, y$ and (σ^l, σ^r) satisfy properties P1 and P3. P2 is immediate by assumption. We simply apply Lemma 5.35 to obtain the result. ◀

Next, we establish that the items listed in the previous lemma can all be checked in polynomial space in $|\mathcal{A}|$.

▶ **Lemma 6.5.** *Given a portal $s, x \rightsquigarrow t, y$, we can check whether it has infinitely many minimal blocking factors in polynomial space in $|\mathcal{A}|$.*

Proof. We start by defining a deterministic automaton $\mathcal{B}$ recognising the set of positional words that are factors of $\mathcal{PL}(s, x \rightsquigarrow t, y)$.

For each $i \in \{0, \dots, p-1\}$ let Q_i be the set of states in the SCC of s that can be reached in $i - x$ steps from s. It is easily computable using the partition of the states given by Fact 3.3.

Let $\mathcal{A}_i$ be $\mathcal{A}$ where the initial states are Q_i and every state in the SCC of s is final. It recognises words that can be read from Q_i in $\mathcal{A}$ without leaving the SCC.

Then, we define $\mathcal{B}_i$ as the automaton obtained by determinising $\mathcal{A}_i$. It has size at most $2^{|\mathcal{A}|}$. From $\mathcal{B}_i$ we easily obtain an automaton $\mathcal{B}'_i$ of size $p|\mathcal{B}_i|$ recognising the set of positional words $\{(i : w) \mid w \in \mathcal{L}(\mathcal{B}_i)\}$: we simply keep track in the states of the number of letters read, plus i, modulo p.

Lastly, we define $\mathcal{B}$ as follows: We take all automata $\mathcal{B}_i$ and merge their initial states into one. Observe that $\mathcal{B}$ is deterministic as all $\mathcal{B}_i$ are, and for all letter (j, a) there is at most one transition from the initial state reading (j, a), which goes to a state of $\mathcal{B}_j$. This automaton is of at most exponential size in $|\mathcal{A}|$. It recognises the set of positional words that are factors of $\mathcal{PL}(s, x \rightsquigarrow t, y)$.

We can complement it to obtain an automaton $\overline{\mathcal{B}}$ recognising the complement language, i.e., the set of positional words that are not factors of $\mathcal{PL}(s, x \rightsquigarrow t, y)$. We have $|\overline{\mathcal{B}}| \leq |\mathcal{B}| + 1$.

A positional word $(n : w)$ is a *minimal blocking factor* of $s, x \rightsquigarrow t, y$ if and only if it is not a factor of $\mathcal{PL}(s, x \rightsquigarrow t, y)$ while removing its first or its last letter makes it a factor of $\mathcal{PL}(s, x \rightsquigarrow t, y)$.

The set of blocking factors can thus be recognised by an automaton of size $|\overline{\mathcal{B}}|^3$, which runs $\overline{\mathcal{B}}$ on the input word, while running $\mathcal{B}$ from the second to the last letter and from the first to the second to last letter. The automaton accepts if all three runs are accepting. It is of exponential size in $|\mathcal{A}|$.

We simply need to check if this automaton has an infinite language, which is the case if and only if it has a cycle reachable from the initial state and from which a final state is reachable. This can be checked by exploring the state space of the automaton, in non-deterministic polynomial space (in $|\mathcal{A}|$), and applying Savitch's theorem. ◀

▶ **Lemma 6.6.** *Given two SCC-paths π and π', one can check in PSPACE whether there is a sequence σ that is blocking for π and not π'.*

Proof.

▷ Claim 6.7. There is a sequence σ that is blocking for $\pi = s_0, x_0 \rightsquigarrow t_0, y_0 \xrightarrow{a_1} \cdots s_k, x_k \rightsquigarrow t_k, y_k$ and not $\pi' = s'_0, x'_0 \rightsquigarrow t'_0, y'_0 \xrightarrow{a'_1} \cdots s'_\ell, x'_\ell \rightsquigarrow t'_\ell, y'_\ell$ if and only if either:

- there is a positional word $(n : w)$ that is a blocking factor for $s_0, x_0 \rightsquigarrow t_0, y_0$ and not $s'_0, x'_0 \rightsquigarrow t'_0, y'_0$ and there is a sequence σ' that is blocking for $s_1, x_1 \rightsquigarrow t_1, y_1 \xrightarrow{a_2} \cdots s_k, x_k \rightsquigarrow t_k, y_k$ and not π',
- or there is a positional word $(n : w)$ that is a blocking factor for $s_0, x_0 \rightsquigarrow t_0, y_0$ and $s'_0, x'_0 \rightsquigarrow t'_0, y'_0$ and there is a sequence σ' that is blocking for $s_1, x_1 \rightsquigarrow t_1, y_1 \xrightarrow{a_2} \cdots s_k, x_k \rightsquigarrow t_k, y_k$ and not $s'_1, x'_1 \rightsquigarrow t'_1, y'_1 \xrightarrow{a'_2} \cdots s'_\ell, x'_\ell \rightsquigarrow t'_\ell, y'_\ell$.

Proof. The right-to-left direction is clear (just take $\sigma = (n : w), \sigma'$ in both cases).

For the left-to-right direction, consider a sequence σ that is blocking for π and not π', of minimal length. Let σ_+ and $(n : w)$ be such that $\sigma = (n : w)\sigma_+$.

- If $(n : w)$ is not blocking for $s_0, x_0 \rightsquigarrow t_0, y_0$ then σ_+ is blocking for π and not π', contradicting the minimality of σ.
- If $(n : w)$ is blocking for $s_0, x_0 \rightsquigarrow t_0, y_0$ and not $s'_0, x'_0 \rightsquigarrow t'_0, y'_0$ then we set $\sigma' = \sigma$. We know that σ is not blocking for π'. On the other hand, as σ is blocking for π, it is also blocking for $s_1, x_1 \rightsquigarrow t_1, y_1 \xrightarrow{a_2} \cdots s_k, x_k \rightsquigarrow t_k, y_k$.
- If $(n : w)$ is blocking for both $s_0, x_0 \rightsquigarrow t_0, y_0$ and $s'_0, x'_0 \rightsquigarrow t'_0, y'_0$ then we set $\sigma' = \sigma$. As σ is blocking for π, it is also blocking for $s_1, x_1 \rightsquigarrow t_1, y_1 \xrightarrow{a_2} \cdots s_k, x_k \rightsquigarrow t_k, y_k$. On the other hand, if σ was blocking for $s'_1, x'_1 \rightsquigarrow t'_1, y'_1 \xrightarrow{a'_2} \cdots s'_\ell, x'_\ell \rightsquigarrow t'_\ell, y'_\ell$, then it would

also be blocking for π', a contradiction. Hence σ is not blocking for $s'_1, x'_1 \rightsquigarrow t'_1, y'_1 \xrightarrow{a'_2} \cdots s'_\ell, x'_\ell \rightsquigarrow t'_\ell, y'_\ell$

◁

The claim above lets us define a recursive algorithm.

- First check if there is a positional word $(n : w)$ that is blocking for $s_0, x_0 \rightsquigarrow t_0, y_0$ and not $s'_0, x'_0 \rightsquigarrow t'_0, y'_0$. If it is the case, make a recursive call to check if there is a sequence σ' that is blocking for $s_1, x_1 \rightsquigarrow t_1, y_1 \xrightarrow{a_2} \cdots s_k, x_k \rightsquigarrow t_k, y_k$ and not π'. If it is the case, answer yes.
- Then check if there is a positional word $(n : w)$ that is a blocking factor for $s_0, x_0 \rightsquigarrow t_0, y_0$ and $s'_0, x'_0 \rightsquigarrow t'_0, y'_0$. If so, make a recursive call to check if there is a sequence σ' that is blocking for $s_1, x_1 \rightsquigarrow t_1, y_1 \xrightarrow{a_2} \cdots s_k, x_k \rightsquigarrow t_k, y_k$ and not $s'_1, x'_1 \rightsquigarrow t'_1, y'_1 \xrightarrow{a'_2} \cdots s'_\ell, x'_\ell \rightsquigarrow t'_\ell, y'_\ell$. If it is the case, answer yes.

If both items fail, answer no.

The existence of those positional words can be checked in polynomial space using the automaton $\mathcal{B}$ constructed in the proof of Lemma 6.5. The depth of the recursive calls is at most the sum of the lengths of π and π', which is bounded by $2|\mathcal{A}|$. In consequence, this algorithm runs in polynomial space.

◀

▶ **Proposition 6.8.** *The following problem is in PSPACE: Given an automaton $\mathcal{A}$, is it hard?*

Proof. We use Lemma 6.4. We guess an SCC-path $\pi = s_0, x_0 \rightsquigarrow t_0, y_0 \xrightarrow{a_1} \cdots s_k, x_k \rightsquigarrow t_k, y_k$ and an index i.

We check that $s_i, x_i \rightsquigarrow t_i, y_i$ has infinitely many minimal blocking factors, using Lemma 6.5.

We then enumerate all SCC-paths in $\mathcal{A}$. For each one $\pi' = s'_0, x'_0 \rightsquigarrow t'_0, y'_0 \xrightarrow{a'_1} \cdots s'_\ell, x'_\ell \rightsquigarrow t'_\ell, y'_\ell$ we guess indices j^l and j^r. We check that every portal $s'_j, x'_j \rightsquigarrow t'_j, y'_j$ with $j^l < j < j^r$ is $\simeq$ -equivalent to $s, x \rightsquigarrow t, y$.

Then, we use Lemma 6.6 to check that there is a sequence σ^l that is blocking for $s'_0, x'_0 \rightsquigarrow t'_0, y'_0 \xrightarrow{a'_1} \cdots s'_{j^l}, x'_{j^l} \rightsquigarrow t'_{j^l}, y'_{j^l}$ and not $s_0, x_0 \rightsquigarrow t_0, y_0 \xrightarrow{a_1} \cdots s_i, x_i \rightsquigarrow t_i, y_i$.

Symmetrically, we check that there is a sequence σ^r that is blocking for $s'_{j^r}, x'_{j^r} \rightsquigarrow t'_{j^r}, y'_{j^r} \xrightarrow{a'_1} \cdots s'_\ell, x'_\ell \rightsquigarrow t'_\ell, y'_\ell$ and not $s_i, x_i \rightsquigarrow t_i, y_i \xrightarrow{a_{i+1}} \cdots s_k, x_k \rightsquigarrow t_k, y_k$.

If all those tests succeed, we answer yes, otherwise we answer no. This algorithm is correct and complete by Lemma 6.4. ◀

Our last result is the PSPACE upper bound on the complexity of checking if a language is trivial. It is based on the characterisation of trivial languages given by Lemma 5.38.

▶ **Proposition 6.9.** *One can check if an automaton has a trivial language in PSPACE.*

Proof. By Lemma 5.38, it suffices to enumerate all accepting SCC-paths in the automaton, and then check that all portals appearing in them have a blocking factor. This is feasible in PSPACE, using the automaton $\overline{\mathcal{B}}$ from the proof of Lemma 6.5. ◀

6.2 Hardness of classifying automata

We prove hardness of the triviality problem and easiness problems, concluding on their PSPACE-completeness. We reduce from the universality problem for NFAs, which is well-known to be PSPACE-complete (see e.g. [1, Theorem 10.14]).

▶ **Lemma 6.10.** *The triviality problem is PSPACE-hard.*

Proof. Consider an NFA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ on an alphabet Σ. Without loss of generality, we assume that $\mathcal{A}$ is trim (up to removing unreachable or non-co-reachable states) and that it accepts all words of length less than 2: this can be checked in polynomial time. Let # and ! be two letters that are not in Σ. We apply the following transformations to $\mathcal{A}$:

- add a transition labeled by ! from every final state to the initial state q_0
- add a self-loop labeled by # to each state.

We call the resulting automaton $\mathcal{B} = (Q, \Sigma \cup \{!, \#\}, \delta', q_0, F)$. Note that $\mathcal{B}$ is strongly connected: consider any two states $q, q' \in Q$, we show that q' is reachable from q. As $\mathcal{A}$ is trim, there exists $q_f \in F$ that is reachable from q, and q' is reachable from the initial state q_0. Furthermore, we have put a ! transition from q_f to q_0, hence q' is reachable from q.

Recall that the language of a strongly connected automaton is trivial if and only if it has no minimal blocking factor. Hence, to complete this reduction, we need to show that $\mathsf{MBF}(\mathcal{B})$ is empty if and only if $\mathcal{A}$ is universal.

First, let us describe the language recognized by $\mathcal{B}$. It is given by

$$\mathcal{L}(\mathcal{B}) = \{u_1!u_2! \cdots !u_n \mid \forall i, u_i \in (\Sigma \cup \{\#\})^* \;\wedge\; \pi_\Sigma(u_i) \in \mathcal{L}(\mathcal{A})\},$$

where $\pi_\Sigma(u)$ is the word in Σ^* obtained by removing all letters not in Σ from u.

▷ Claim 6.11. If $\mathcal{A}$ is universal, then $\mathcal{B}$ is also universal.

Proof. Indeed, any word in u in can be uniquely decomposed into $u = u_1!u_2! \cdots !u_n$ where each u_i does not contain the letter "!". As # is idempotent on $\mathcal{B}$, $\delta'(q_0, u_i)$ is equal to $\delta(q_0, \pi_\Sigma(u_i))$ for every i. Since $\mathcal{A}$ is universal, each of the $\delta'(q_0, u_i)$ contains a final state, hence $\delta'(q_0, u_i!) = \{q_0\}$. Therefore, the set $\delta'(q_0, u)$ is equal to $\delta'(q_0, u_n)$, which contains a final state, and u is in $\mathcal{L}(\mathcal{B})$, which shows that $\mathcal{B}$ is universal. ◁

This shows that if $\mathcal{A}$ is universal, then $\mathsf{MBF}(\mathcal{B})$ is empty.

For the converse, we show that a word $w \in \Sigma^*$ not in $\mathcal{L}(\mathcal{A})$ induces minimum blocking factors for $\mathcal{B}$. Consider such a w of minimal size. As we assumed that $\mathcal{A}$ accepts all words of size less than 2, $|w| \geq 2$. Let u, v be words of length at least 1 such that $w = uv$. For all $n \in \mathbb{N}$, at least one of $u\#^n v, !u\#^n v, u\#^n v!, !u\#^n v!$ is a minimal blocking factor (depending respectivelyon whether w is not a factor of any word of $\mathcal{L}(\mathcal{A})$ or is a prefix/suffix of a word of $\mathcal{L}(\mathcal{A})$ or not). As a consequence, $\mathcal{B}$ has infinitely many blocking factors, and is thus hard to test by Theorem 4.2.

In summary, $\mathcal{A}$ is universal if and only if $\mathcal{B}$ is trivial to test. This shows the PSPACE-hardness of the triviality problem. ◀

The above proof can be extended to show the PSPACE-hardness of the easiness problem.

▶ **Corollary 6.12.** *The easiness problem is PSPACE-hard.*

Proof. We proceed as in the proof of Lemma 6.10: given an automaton $\mathcal{A}$ over an alphabet Σ, we build an automaton $\mathcal{B}$ over the alphabet $\Sigma \cup \{!, \#\}$ such that if $\mathcal{A}$ is universal, $\mathsf{MBF}(\mathcal{B})$ is empty, and if $\mathcal{A}$ is not universal, then $\mathsf{MBF}(\mathcal{B})$ is infinite.

To show the hardness of the easiness problem, let $\flat$ denote a new letter not in $\Sigma \cup \{\#, !\}$ and consider the automaton $\mathcal{B}'$ equal to $\mathcal{B}$ but taken over the alphabet $\Sigma \cup \{\#, !, \flat\}$. As there are no transitions labeled by $\flat$ in $\mathcal{B}'$, the word $\flat$ is always a minimum blocking factor of $\mathcal{B}'$. As a result, we have $\mathsf{MBF}(\mathcal{B}') = \mathsf{MBF}(\mathcal{B}) \cup \{\flat\}$, hence $\mathcal{A}$ is universal if and only if $\mathsf{MBF}(\mathcal{B}')$ is finite but non-empty: by Theorem 4.2, this is equivalent to $\mathcal{L}(\mathcal{B}')$ is easy to test. Therefore, the easiness problem is also PSPACE-hard. ◀

This concludes the proof of Theorem 6.1

References

1. Alfred V. Aho and John E. Hopcroft. *The Design and Analysis of Computer Algorithms*. Addison-Wesley Longman Publishing Co., Inc., 1974.
2. Maryam Aliakbarpour, Ilias Diakonikolas, and Ronitt Rubinfeld. Differentially private identity and equivalence testing of discrete distributions. In *International Conference on Machine Learning, ICML*, 2018. URL: https://proceedings.mlr.press/v80/aliakbarpour18a.html.
3. Noga Alon, Richard A. Duke, Hanno Lefmann, Vojtech Rödl, and Raphael Yuster. The algorithmic aspects of the regularity lemma. *Journal of Algorithms*, 16(1):80–109, 1994. doi:10.1006/JAGM.1994.1005.
4. Noga Alon, Eldar Fischer, Michael Krivelevich, and Mario Szegedy. Efficient testing of large graphs. *Combinatorica*, 20(4):451–476, 2000. doi:10.1007/s004930070001.
5. Noga Alon, Michael Krivelevich, Ilan Newman, and Mario Szegedy. Regular languages are testable with a constant number of queries. *SIAM Journal on Computing*, 30(6):1842–1862, 2001. doi:10.1109/SFFCS.1999.814641.
6. Noga Alon and Asaf Shapira. Every monotone graph property is testable. *SIAM Journal of Computing*, 38(2):505–522, 2008. doi:10.1137/050633445.
7. Gabriel Bathie and Tatiana Starikovskaya. Property testing of regular languages with applications to streaming property testing of visibly pushdown languages. In *International Colloquium on Automata, Languages, and Programming, ICALP*, 2021. doi:10.4230/LIPIcs.ICALP.2021.119.
8. Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-testing/correcting with applications to numerical problems. *Journal of Computer and System Sciences*, 47(3):549–595, 1993. doi:10.1016/0022-0000(93)90044-W.
9. Ilias Diakonikolas and Daniel M Kane. A new approach for testing properties of discrete distributions. In *IEEE Symposium on Foundations of Computer Science, FOCS*, pages 685–694. IEEE, 2016. doi:10.1109/FOCS.2016.78.
10. Yevgeniy Dodis, Oded Goldreich, Eric Lehman, Sofya Raskhodnikova, Dana Ron, and Alex Samorodnitsky. Improved testing algorithms for monotonicity. In *International Workshop on Randomization and Approximation Techniques in Computer Science, RANDOM*, pages 97–108. Springer, 1999. doi:10.1007/978-3-540-48413-4_10.
11. Funda Ergün, Sampath Kannan, S.Ravi Kumar, Ronitt Rubinfeld, and Mahesh Viswanathan. Spot-checkers. *Journal of Computer and System Sciences*, 60(3):717–751, 2000. doi:10.1006/jcss.1999.1692.
12. Nathanaël François, Frédéric Magniez, Michel de Rougemont, and Olivier Serre. Streaming Property Testing of Visibly Pushdown Languages. In *European Symposium on Algorithms, ESA*, volume 57, pages 43:1–43:17, Dagstuhl, Germany, 2016. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ESA.2016.43.
13. Oded Goldreich. *Introduction to property testing*. Cambridge University Press, 2017. doi:10.1017/9781108135252.

14 Oded Goldreich, Shari Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, 45(4):653–750, jul 1998. doi:10.1145/285055.285060.

15 Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *The collected works of Wassily Hoeffding*, pages 409–426, 1994.

16 Stefan Kiefer and Corto Mascle. On finite monoids over nonnegative integer matrices and short killing words. *Symposium on Theoretical Aspects of Computer Science, STACS*, 126, 2019. doi:10.4230/LIPIcs.STACS.2019.43.

17 Stefan Kiefer and Corto N Mascle. On nonnegative integer matrices and short killing words. *SIAM Journal on Discrete Mathematics*, 35(2):1252–1267, 2021. doi:10.1137/19M1250893.

18 Frédéric Magniez and Michel de Rougemont. Property testing of regular tree languages. *Algorithmica*, 49(2):127–146, 2007. doi:10.1007/s00453-007-9028-3.

19 Liam Paninski. A coincidence-based test for uniformity given very sparsely sampled discrete data. *IEEE Transactions on Information Theory*, 54(10):4750–4755, 2008. doi:10.1109/TIT.2008.928987.

20 Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):252–271, 1996. doi:10.1137/S0097539793255151.

21 Andrew Chi-Chin Yao. Probabilistic computations: Toward a unified measure of complexity. In *Symposium on Foundations of Computer Science, SFCS*, pages 222–227. IEEE, 1977. doi:10.1109/SFCS.1977.24.

A Properties of minimal blocking factors

In this section, we discuss properties of the set of minimal blocking factors of an NFA. First, we show the set of minimal blocking factors of an automaton is a regular language.

▶ **Lemma A.1.** *Let $\mathcal{A} = (Q, \Sigma, \delta, I, F)$ be a strongly connected NFA with m states and let $\lambda = \lambda(\mathcal{A})$. For every $i \in \mathbb{Z}/\lambda\mathbb{Z}$, the set of minimal blocking factors of $\mathcal{A}$ of the form $(i : u)$ is a regular language recognized by a NFA of size $2^{\mathcal{O}(m)}$.*

Proof. We call blocking factors of $\mathcal{A}$ of the form $(i : u)$ its *i-blocking factors*.

We first show that the set of i-blocking factors of $\mathcal{A}$, but not necessarily minimal ones, is a regular language recognized by an NFA $\mathcal{A}_i$ with $m + 1$ states. The result follows by using a standard construction for the automaton recognizing words in a regular language L that have no proper factor in a regular language L', which gives an automaton of size $2^{\mathcal{O}(m)}$.

Consider the NFA $\mathcal{A}_i$ obtained by adding a new sink state $\bot$ to $\mathcal{A}$, making it the only accepting state, with initial states Q_i. Formally, $\mathcal{A}_i$ is defined as $\mathcal{A}_i = (Q \cup \{\bot\}, \Sigma, \delta', Q_i, \{\bot\})$, where δ' is defined as follows:

$$\forall p \in Q, \forall a \in \Sigma : \delta'(p, a) = \begin{cases} \{\bot\} & \text{if } \delta(p, a) = \emptyset, \\ \delta(p, a) & \text{otherwise.} \end{cases}$$

This automaton[3] recognizes the set of i-blocking factors of $\mathcal{A}$ and has size $\mathcal{O}(m)$. Applying the aforementioned construction to $L = L' = \mathcal{L}(\mathcal{A}_i)$ yields the desired automaton, of size $2^{\mathcal{O}(m)}$. ◀

It follows that the set of minimal blocking factors of $\mathcal{A}$ is also a regular language.

▶ **Corollary A.2.** *Let $\mathcal{A}$ be an NFA with m states. The set of minimal blocking factors of $\mathcal{A}$ is a regular language recognized by an NFA of size $2^{\mathcal{O}(m)}$.*

Therefore, if $\mathsf{MBF}(\mathcal{A})$ is infinite, we can use Kleene's lemma to find an infinite family of minimal blocking factors with a shared structure $\{\phi\nu^r\chi, r \in \mathbb{N}\}$.

▶ **Lemma 4.20.** *If $\mathsf{MBF}(\mathcal{A})$ is infinite, then there exist positional words ϕ, ν_+, ν_-, χ such that:*

1. *the words ν_+ and ν_- have the same length,*
2. *there exists a constant $S = 2^{\mathcal{O}(m)}$ such that $|\phi|, |\nu_+|, |\nu_-|, |\chi| \leq S$,*
3. *there exists an index $i_* \in \mathbb{Z}/\lambda\mathbb{Z}$ and a state $q_* \in Q_{i_*}$ such that for every integer $r \geq 1$, $\tau_{-,r} = \phi(\nu_-)^r z$ is blocking for $\mathcal{A}$, and for every $s < r$, we have*

$$q_* \xrightarrow{\tau_{+,r,s}} q_* \text{ where } \tau_{+,r,s} = \phi(\nu_-)^j \nu_+ (\nu_-)^{r-1-s} \chi.$$

In particular, $\tau_{+,r,s}$ is not blocking for $\mathcal{A}$.

Note that here, the state q_* is the same for *every* integers r, s.

Proof. As $\mathsf{MBF}(\mathcal{A})$ is infinite, there must exist an i_* such that $\mathcal{A}$ has infinitely many minimal i_*-blocking factors; we fix such an i_* in what follows.

As the set of minimal i_*-blocking factors is an infinite regular language recognized by an NFA of size $S = 2^{\mathcal{O}(m)}$, by Kleene's Lemma, there exist positional words τ, μ, η, each of

[3] Our definition of NFAs does not allow for multiple initial states. As there is no constraint of strong connectivity for $\mathcal{A}_i$, this can be solved using a simple construction that adds a new initial state.

length at most S with $|\mu| \geq 1$, such that for any non-negative integer $k, \tau\mu^k\eta$ is a minimal i_*-blocking factor. We can assume w.l.o.g. that neither τ nor η is empty, otherwise we set their value to μ: after this modification, $\tau\mu^k\eta$ is still a minimal i_*-blocking factor for every $k \geq 0$.

Notice that the word $\tau\mu^m$ is not a blocking factor, as a proper factor of the minimal blocking factor $\tau\mu^m\eta$. Therefore, by the pigeonhole principle, there exist integers $k_0, k_1 \geq 1$ with $k_0 + k_1 = m$ and states p, p_1 such that we have

$$p \xrightarrow{\tau\mu^{k_0}} p_1 \xrightarrow{\mu^{k_1}} p_1.$$

Note that, by Fact 3.3, $p_1 \xrightarrow{\mu^{k_1}} p_1$ implies that $k_1 \cdot |\mu| = 0 \pmod{\lambda}$.

Similarly, the word $\mu^m\eta$ is not a blocking factor, since it is a proper factor of the minimal i_*-blocking factor $\tau\mu^m\eta$. Again, there exist integers $k_2 \geq 1, k_3$ summing to m and states p_2 and q such that

$$p_2 \xrightarrow{\mu^{k_2}} p_2 \xrightarrow{\mu^{k_3}\eta} q.$$

Now, define $\phi = \tau\mu^{k_0}, \chi = \mu^{k_3}\eta$ and $\nu_- = \mu^K$, where $K = \rho \cdot k_1 \cdot k_2$. As there are transitions starting from p_1 and p_2 labeled by μ, p_1 and p_2 belong to the same periodicity class. Therefore, by Fact 3.3, as $K \geq \rho$ and $K \cdot |\mu| = 0 \pmod{\lambda}$, there exists a word ν_+ of length $K \cdot |\mu|$ such that $p_1 \xrightarrow{\nu_+} p_2$. This choice of ϕ, ν_+, ν_- and χ satisfies all the conditions of the lemma. ◀

B Hoeffding's inequality

▶ **Lemma B.1** ([15, Theorem 2]). *Let $X_1, \ldots, X_k$ be independent random variables such that for every $i = 1, \ldots, k$, we have $a_i \leq X_i \leq b_i$, and let $S = \sum_{i=1}^k$. Then, for any $t > 0$, we have*

$$\mathbb{P}\left(\mathbb{E}[S] - S \geq t\right) \leq \exp\left(-\frac{2t^2}{\sum_{i=1}^k (b_i - a_i)^2}\right).$$