The State Complexity of Alternating Automata*

Nathanaël Fijalkow[1,2,3]

1 CNRS, LaBRI, Bordeaux, France
2 Alan Turing Institute, London, United Kingdom
3 University of Warwick, United Kingdom

Abstract

This paper studies the complexity of languages of finite words using automata theory. To go beyond the class of regular languages, we consider *infinite* automata and the notion of *state complexity* defined by Karp. We look at *alternating automata* as introduced by Chandra, Kozen and Stockmeyer: such machines run independent computations on the word and gather their answers through boolean combinations.

We devise a lower bound technique relying on boundedly generated lattices of languages, and give two applications of this technique. The first is a hierarchy theorem, stating that there are languages of arbitrarily high polynomial alternating state complexity, and the second is a linear lower bound on the alternating state complexity of the prime numbers written in binary. This second result strengthens a result of Hartmanis and Shank from 1968, which implies an exponentially worse lower bound for the same model.

1998 ACM Subject Classification F.1.1 Models of Computation

Keywords and phrases Automata Theory, State Complexity, Lower Bounds, Alternating Automata, Hierarchy Theorem, Prime Numbers

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

1 Introduction

The seminal paper of Karp [15] defines the *state complexity* of an (infinite) automaton as a function associating with n the number of states reachable by reading a word of length at most n. For a function $f : \mathbb{N} \to \mathbb{N}$, a language $L \subseteq A^*$ has state complexity f if there exists an automaton recognising L of state complexity at most f.

For the case of deterministic automata, state complexity is fully characterised by the celebrated Myhill-Nerode theorem [19], which states the existence of a canonical minimal (potentially infinite) automaton for a given language based on the notion of left quotients. Nevertheless, it is sometimes complicated to understand the structure of this automaton, as demonstrated by the case of the language of prime numbers written in binary: a series of papers culminates in a result of Hartmanis and Shank [14] showing that this language has asymptotically maximal (i.e., exponential) deterministic state complexity.

In this paper, we initiate the study of *alternating state complexity*, which uses Karp's definition instantiated with (infinite) alternating automata. We first motivate the model with some examples and later discuss its relevance. Formal definitions are given in the next section; we stick to intuitive explanations in this introduction.

* This project has received funding from the Alan Turing Institute under EPSRC grant EP/N510129/1 and the DeLTA project (ANR-16-CE40-0007).

Consider the language

$$\textsc{CountEq}_3 = \left\{ w \in \{a,b,c\}^* \mid |w|_a = |w|_b = |w|_c \right\},$$

consisting of words having the same number of a's, b's and c's. (We let $|w|_a$ denote the number of letters a in w.) This language is not regular, but we claim that it is recognised by a deterministic automaton of quadratic state complexity. Indeed, we construct an automaton whose set of states is $\mathbb{Z}^2$, interpreted as two counters. They are initialised to 0 each and maintain the value $(|w|_a - |w|_b, |w|_a - |w|_c)$. To this end, the letter a acts as $(+1,+1)$, the letter b as $(-1,0)$, the letter c as $(0,-1)$. The only accepting state is $(0,0)$. This automaton is of quadratic state complexity: after reading the word w the automaton is in the state $(|w|_a - |w|_b, |w|_a - |w|_c)$, which means that the set of states reachable by words of length at most n has size $(2n+1)^2$.

Consider now the language

$$\textsc{NotEq} = \left\{ u\sharp v \mid u, v \in \{0,1\}^*, u \neq v \right\},$$

consisting of two words u, v over the alphabet $\{0,1\}$ separated by the letter $\sharp$ such that u is different from v. One can easily see that this language does not have subexponential deterministic state complexity: after reading two different words u and u', any deterministic automaton recognising NotEq must be in two different states.

However, it is recognised by a non-deterministic automaton of linear state complexity. Note that there are three ways to have $u \neq v$: either v is longer than u, or v is shorter than u, or there exists a position at which they differ. At the beginning the automaton guesses which of these three situations occur. We focus on the third possibility for the informal explanation. The automaton guesses a position in the first word, stores in the state the position p together with the letter a at this position, and checks whether the corresponding position in the second word indeed differs. To this end, after reading the letter $\sharp$, it decrements the position until reaching 1, and checks whether the letter is indeed different than the letter stored in the state.

Our third example is the language

$$\textsc{Lexicographic} = \left\{ u\sharp v \mid u, v \in \{0,1\}^*, u <_{\text{lex}} v \right\},$$

consisting of two words u, v over the alphabet $\{0,1\}$ separated by the letter $\sharp$ such that u is lexicographically smaller than v. One can see that this language does not have subexponential non-deterministic state complexity (we do not substantiate this claim here). However, we claim that it is recognised by an alternating automaton of linear state complexity.

The notion of alternating (Turing) machines was introduced by Chandra, Kozen and Stockmeyer [6, 16, 5]. A non-deterministic automaton makes guesses about the word, and the computation is accepting if there exists a sequence of correct guesses. In other words, these guesses are disjunctive choices; the alternating model restores the symmetry by introducing disjunctive and conjunctive choices. Whenever the automaton makes a choice, we say that it creates independent copies of itself, one for each alternatives; if the choice was disjunctive, the computation is accepted if some copy accepts, and if the choice was conjunctive, the computation is accepted if all copies accept.

We illustrate this notion by constructing an alternating automaton for Lexicographic. We unravel the inductive definition of the lexicographic order: $u <_{\text{lex}} v$ if and only if

$$(u(0) = 0 \wedge v(0) = 1) \vee (u(0) = v(0) \wedge u(\geq 1) <_{\text{lex}} v(\geq 1)).$$

Here $u(0)$ is the first letter of u, and $u(\geq 1)$ is the word u stripped of its first letter. Upon reading the first letter $u(0)$, the automaton makes a disjunctive guess corresponding to the disjunction in the definition: either $u(0) = 0$ and $v(0) = 1$, or $u(0) = v(0)$ and $u(\geq 1) <_{\text{lex}} v(\geq 1)$. In the latter case, the automaton makes a further choice, conjunctive this time, checking with one copy that $u(0) = v(0)$ and with another that $u(\geq 1) <_{\text{lex}} v(\geq 1)$.

Alternating automata are succinct. It is well-known that finite deterministic, non-deterministic and alternating automata are equivalent. As hinted by the examples discussed above, for infinite automata we do not have such an equivalence. Some classical constructions still apply, for instance the powerset construction to determinise automata, which increases the state complexity exponentially. Similarly one can transform alternating automata into deterministic ones, increasing the state complexity by a two-fold exponential. Hence one can see alternating automata as a class of *succinctly* represented deterministic automata, whose inner boolean structure is made explicit.

Alternating automata are distributed. Another appeal of alternating automata is as a model of distributed computation. Indeed, in the course of its computation, an alternating automaton produces copies of itself that can be run independently on a distributed architecture. The final output is then computed by boolean combinations of the answers of each copy. This point of view echoes the recent work of Reiter [22], which combines ideas from distributed algorithms and alternating automata.

Applications. The notion of state complexity is used as a complexity measure to evaluate how complicated some operations on languages are. We refer to the surveys [28, 29, 10] for more details on this long line of work. The other natural use of state complexity is as a tool for separating models of computations. For instance, the paper of Dawar and Kreutzer [7] generalises the notion of automaticity (see related works) to relational structures and uses it for separating several modal and non-modal fixed-point logics.

Contributions of the paper. We study alternating state complexity, i.e., the number of states required to recognise a given language using an alternating automaton. We devise a generic lower bound technique based on boundedly generated lattices of languages.

We give the basic definitions and show some examples in Section 2. We discuss related works in Section 3. We describe our lower bound technique in Section 4, and give two applications:

- **Hierarchy theorem**: in Section 5, we prove a hierarchy theorem: for each natural number ℓ greater than or equal to 2, there exists a language having alternating state complexity n^ℓ but not $n^{\ell-\varepsilon}$ for any $\varepsilon > 0$.
- **Prime numbers**: in Section 6, we look at the language of prime numbers written in binary. The works of Hartmanis and Shank culminated in showing that it does not have subexponential *deterministic* state complexity [14]. We consider the stronger model of *alternating* automata, and first observe that Hartmanis and Shank's techniques imply a *logarithmic* lower bound on the *alternating* state complexity. Our contribution is to strengthen this result by showing a *linear* lower bound, which is thus an exponential improvement.

2 Definitions

We fix an *alphabet* A, which is a finite set of letters. A *word* is a finite sequence of letters $w = w(0)w(1)\cdots w(n-1)$, where the $w(i)$ are letters from the alphabet A, i.e., $w(i) \in A$. We say that w has length n, and write $|w|$ for the length of w. The empty word is ε. We let A^* denote the set of all words and $A^{\leq n}$ the set of words of length at most n. A language, typically denoted by L, is a set of words.

For a set E, we let $\mathcal{B}^+(E)$ denote the set of boolean formulae over E, i.e., using conjunctions and disjunctions. Throughout the paper we only consider *positive* boolean combinations. For instance, if $E = \{p, q, r\}$, an element of $\mathcal{B}^+(E)$ is $p \wedge (q \vee r)$. A conjunctive formula uses only conjunctions, and a disjunctive formula only disjunctions. For $\delta \in \mathcal{B}^+(E)$ and $X \subseteq E$, we write $X \models \delta$ if δ is true when setting the elements of X to true and the others to false.

▶ **Definition 1** (Alternating Automata [6, 16, 5])**.** An alternating automaton is given by a (potentially infinite) set Q of states, an initial state $q_0 \in Q$, a transition function $\delta : Q \times A \to \mathcal{B}^+(Q)$ and a set of accepting states $F \subseteq Q$.

We use acceptance games to define the semantics of alternating automata. Consider an alternating automaton $\mathcal{A}$ and an word w, we define the acceptance game $\mathcal{G}_{\mathcal{A},w}$ as follows: it has two players, Eve and Adam. Eve claims that the word w should be accepted, and Adam challenges this claim.

The game starts from the initial state q_0, and with each letter of w read from left to right, a state is chosen through the interaction of the two players. If in a state q and reading a letter a, Eve and Adam look at the boolean formula $\delta(q, a)$; Eve chooses which clause is satisfied in a disjunction, and Adam does the same for conjunctions. This leads to a new state p, from which the computation continues. A play is won by Eve if it ends up in an accepting state.

The word w is accepted by $\mathcal{A}$ if Eve has a winning strategy in the acceptance game $\mathcal{G}_{\mathcal{A},w}$. The language recognised by $\mathcal{A}$ is the set of words accepted by $\mathcal{A}$.

As special cases, an automaton is

- *non-deterministic* if for all q in Q, a in A, $\delta(q, a)$ is a disjunctive formula,
- *universal* if for all q in Q, a in A, $\delta(q, a)$ is a conjunctive formula,
- *deterministic* if for all q in Q, a in A, $\delta(q, a)$ is an atomic formula, i.e., if $\delta : Q \times A \to Q$.

▶ **Definition 2** (State Complexity Classes [15])**.** Fix a function $f : \mathbb{N} \to \mathbb{N}$. The language L is in $\mathrm{Alt}\,(f)$ if there exists an alternating automaton recognising L and a constant C such that for all n in $\mathbb{N}$:

$$\begin{aligned}\left|\left\{q \in Q \mid \exists w \in A^{\leq n},\ \text{it is possible to reach } q \text{ in } \mathcal{G}_{\mathcal{A},w}\right\}\right| \\ \leq C \cdot f(n).\end{aligned}$$

Similarly, we define $\mathrm{NonDet}\,(f)$ for non-deterministic automata and $\mathrm{Det}\,(f)$ for deterministic automata.

For the sake of succinctness, the acronym SC will be used in lieu of state complexity. We write $f(n)$ for the function $f : n \mapsto f(n)$, so for instance $\mathrm{Alt}\,(n)$ is the class of languages having linear alternating SC. We say that L has sublinear (respectively subexponential) alternating SC if it is recognised by an alternating automaton of state complexity at most f, where $f = o(n)$ (respectively $f = 2^{o(n)}$).

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

$$\mathrm{Det}\,(1) = \mathrm{NonDet}\,(1) = \mathrm{Alt}\,(1) = \mathrm{Reg},$$

i.e., a language has constant SC if and only if it is regular.

We remark that $\mathrm{Det}\,(|A|^n)$ is the class of all languages. Indeed, consider a language L, we construct a deterministic automaton recognising L of exponential state complexity. Its set of states is A^*, the initial state is ε and the transition function is defined by $\delta(w,a) = wa$. The set of accepting states is simply L itself. The number of different states reachable by all words of length at most n is the number of words of length at most n, i.e., $\frac{|A|^{n+1}-1}{|A|-1}$.

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

3 Related Works

The definition of SC is due to Karp [15], and the first result proved in that paper is that non-regular languages have at least linear deterministic SC. Hartmanis and Shank considered the language of prime numbers written in binary, and showed in [14] that it does not have subexponential deterministic SC. We pursue this question in this paper by considering the alternating SC of the prime numbers. Recently, we investigated the SC of probabilistic automata; we substantiated a claim by Rabin [21], by exhibiting a probabilistic automaton which does not have subexponential deterministic SC [9].

Automaticity was defined by Shallit and Breitbart and studied in depth in a series of four papers [27, 20, 11, 26].

▶ **Definition 3.** The *automaticity* of a language L is the function $\mathrm{Aut}(L) : \mathbb{N} \to \mathbb{N}$ which associates with n the size of the smallest deterministic automaton which agrees with L on all words of length at most n.

The conceptual difference is that automaticity is a *non-uniform notion*, since there is a finite automaton for each n, whereas state complexity is *uniform*, since it considers one infinite automaton. For this reason, the two measures behave completely differently.

For instance, consider the language

$$L_{\log} = \left\{ w \in \{a,b,\sharp\}^* \;\middle|\; \begin{array}{l} w = uv \cdot \sharp \cdot u, \\ u, v \in \{a,b\}^*, |u| = \lfloor \log(|w|) \rfloor \end{array} \right\}.$$

In words: the prefix of w of length $\lfloor \log(|w|) \rfloor$ repeats just after the unique letter $\sharp$.

The automaticity of this language is linear, i.e., rather small: $\mathrm{Aut}(L_{\log})(n) = O(n)$. Indeed, given n, the automaton $\mathcal{A}_n$ stores the prefix up to $\lfloor \log(n) \rfloor$, waits for the letter $\sharp$, and compares it to the word starting after $\sharp$.

On the other hand, the deterministic SC of $L_{\log}$ is maximal, meaning exponential: indeed, since the automaton has no information on how long the prefix to be repeated may be, it has to store the whole word. More formally, for any two words $u \neq v$, any deterministic automaton recognising $L_{\log}$ must be in two different states after reading u and after reading v.

Note that replacing log by a very slow growing function yields examples showing that the gap between automaticity and deterministic SC is arbitrarily large.

Another interesting point to make here is the difference between finite and infinite automata. Indeed, studying the SC of finite alternating automata can be reduced to the SC of finite deterministic automata by reversing the words. The notation u^R stands for the reverse of u:

$$u^R = u(n-1) \cdots u(0).$$

We extend it to languages: $L^R = \{u^R \mid u \in L\}$. The following result is a variant of Brzozowski's minimization by reversal technique [4], and a classical result in automata theory.

▶ **Lemma 4** ([25, 8]).

- *If L is recognised by an alternating automaton with n states, then L^R is recognised by a deterministic automaton with 2^n states.*
- *If L is recognised by an deterministic automaton with n states, then L^R is recognised by an alternating automaton with $\lceil \log(n) \rceil$ states.*

In other words, the number of states of the smallest finite deterministic automaton recognising L is (almost) exactly 2^n, where n is the number of states of the smallest finite alternating automaton recognising L^R.

This result does not extend to SC for infinite automata: indeed, since every language has exponential deterministic SC, this would imply that every language also has linear alternating SC. That does not hold: we exhibit in Subsection 4.3 a language which does not have subexponential alternating SC.

Two notions share some features with alternating SC.

The first is boolean circuits; as explained in [9], the resemblance is only superficial, as circuits do not process the input from left to right. For instance, one can observe that the language Parity, which is hard to compute with a circuit (not in AC^0 for instance), is actually a regular language, so trivial with respect to SC.

The second notion is alternating communication complexity, developed by Babai, Frankl and Simon [3]. In this setting, Alice has an input x in A, Bob an input y in B, and they want to determine $h(x, y)$ for a given boolean function $h : A \times B \to \{0, 1\}$ known by all. Alice and Bob are referees in a discussion involving two individuals, Eve and Adam. Eve tries to convince Alice and Bob that $h(x, y) = 1$, and Adam aims at the opposite. A protocol of exchanging messages depending on the inputs is agreed upon by everyone beforehand. Then the input x is revealed to Alice and y to Bob. Eve and Adam both know the two inputs and exchange messages whose conformity to the inputs is checked by Alice and Bob. The cost of the protocol is the number of bits exchanged.

The main difference between communication complexity and state complexity is that protocols do not have to extract information from the inputs sequentially as an automaton does. For instance, swapping the inputs of Alice and Bob does not make any difference for communication complexity but can completely change the state complexity.

As an example, consider the following language studied in Subsection 4.3.

$$L = \left\{ u\sharp u_1 \sharp u_2 \sharp \cdots \sharp u_k \;\middle|\; \begin{array}{l} u, u_1, \ldots, u_k \in \{0,1\}^*, \\ \exists j \in \{1, \ldots, k\}, u = u_j \end{array} \right\}.$$

Alice receives u of length n and Bob receives $u_1 \sharp u_2 \sharp \cdots \sharp u_k$, and they want to check whether there exists $j \in \{1, \ldots, k\}$ such that $u = u_j$. A simple protocol is for Eve to send j, and then for Adam to send $i \in \{1, \ldots, n\}$ together with the letter $u(i)$, to which Eve answers with the letter $u_j(i)$. If the two letters match the exchange is a success, otherwise it is a failure.

An alternating automaton cannot simulate this protocol, because it would need to choose $j \in \{1, \ldots, k\}$ at the beginning, even before reading u. The formal proof of this intuition is that this language does not have subexponential alternating complexity, as proved in Subsection 4.3.

However, if we swap the two inputs, i.e., the automaton reads $u_1 \sharp u_2 \sharp \cdots \sharp u_k$ before u, then it can simulate the protocol: when reading u_j it non-deterministically decides to store u_j, and later checks using universal guesses that $u_j = u$.

This example shows that using alternating communication complexity would not yield strong lower bounds for alternating state complexity. Building on the ideas behind the language L one can obtain arbitrary gaps between the two notions.

4 A Lower Bound Technique

In this section, we develop a generic lower bound technique for alternating state complexity. It is based on the size of generating families for some lattices of languages; we describe it in Subsection 4.1, and a concrete approach to use it, based on query tables, is developed in Subsection 4.2. We apply it to an example in Subsection 4.3.

4.1 Boundedly Generated Lattices of Languages

Let L be a language and u a word. The left quotient of L with respect to u is

$$u^{-1}L = \{v \mid uv \in L\}.$$

If u has length at most n, we say that $u^{-1}L$ is a left quotient of L of order n.

The notion of left quotients stems from the notion of left Myhill-Nerode equivalence relation [19], which allows us to define a canonical minimal *deterministic* automaton. We use the notion of left quotients to derive lower bounds for *alternating* automata.

A lattice of languages is a set of languages closed under union and intersection. Given a family of languages, the lattice it generates is the smallest lattice containing this family.

▶ **Theorem 5.** *If L is in* $\mathrm{Alt}(f)$*, then there exists a constant C such that for all $n \in \mathbb{N}$, there exists a family of at most $C \cdot f(n)$ languages whose generated lattice contains all the left quotients of L of order n.*

To some extent, Theorem 5 draws from the classical Myhill-Nerode theorem [19]. However, since there is no notion of minimal alternating automaton, the situation is more complicated here. In particular, this suggests that the converse of Theorem 5 may not hold.

Theorem 5 reduces the question of finding lower bounds for alternating state complexity to the following one: given a finite lattice of languages, what is the size of the smallest set of generators for this lattice?

Proof. Let $\mathcal{A}$ be an alternating automaton recognising L of state complexity at most f.

Fix n. Let Q_n denote the set of states reachable by some word of length at most n; by assumption $|Q_n|$ is at most $C \cdot f(n)$ for some constant C. For q in Q_n, let $L(q)$ be the language recognised by $\mathcal{A}$ taking q as initial state, and $\mathcal{L}_n$ the family of these languages.

We prove by induction over n that all left quotients of L of order n can be obtained as boolean combinations of languages in $\mathcal{L}_n$.

The case $n = 0$ is clear, since $\varepsilon^{-1}L = L = L(q_0)$.

Consider a word w of length $n+1$, write $w = ua$. We are interested in $w^{-1}L = a^{-1}(u^{-1}L)$, so let us start by considering $u^{-1}L$. By the induction hypothesis, $u^{-1}L$ can be obtained as a boolean combination of languages in $\mathcal{L}_n$: write $u^{-1}L = \phi(\mathcal{L}_n)$, meaning that ϕ is a boolean formula whose atoms are languages in $\mathcal{L}_n$.

Now consider $a^{-1}\phi(\mathcal{L}_n)$. Observe that the left quotient operation respects both unions and intersections, i.e.,

$$a^{-1}(L_1 \cup L_2) = a^{-1}L_1 \cup a^{-1}L_2,$$

and

$$a^{-1}(L_1 \cap L_2) = a^{-1}L_1 \cap a^{-1}L_2.$$

It follows that $w^{-1}L = a^{-1}(\phi(\mathcal{L}_n)) = \phi(a^{-1}\mathcal{L}_n)$; this notation means that the atoms are languages of the form $a^{-1}M$ for M in $\mathcal{L}_n$, i.e., $a^{-1}L(q)$ for q in S_n.

To finish the proof, we remark that $a^{-1}L(q)$ can be obtained as a boolean combination of the languages $L(p)$, where p are the states that appear in $\delta(q, a)$. To be more precise, we introduce the notation $\psi(L(\cdot))$, on an example: if $\psi = p \wedge (r \vee s)$, then $\psi(L(\cdot)) = L(p) \wedge (L(r) \vee L(s))$. With this notation, $a^{-1}L(q) = \delta(a, q)(L(\cdot))$. Thus, for q in Q_n, we have that $a^{-1}L(q)$ can be obtained as a boolean combination of languages in $\mathcal{L}_{n+1}$.

Putting everything together, it implies that $w^{-1}L$ can be obtained as a boolean combination of languages in $\mathcal{L}_{n+1}$, finishing the inductive proof. ◀

4.2 The Query Table Method

Thanks to Theorem 5, we are now looking at the size of the smallest set of generators for a given finite lattice of languages. To study this quantity we define the notion of query tables.

▶ **Definition 6** (Query Table). Consider a family of languages $\mathcal{L}$. Given a word w, its profile with respect to $\mathcal{L}$, or $\mathcal{L}$-profile, is the boolean vector stating whether w belongs to L, for each L in $\mathcal{L}$. The size of the query table of $\mathcal{L}$ is the number of different $\mathcal{L}$-profiles, when considering all words.

For a language L, its query table of order n is the query table of the left quotients of L of order n.

The name query table comes from the following image, illustrated in Figure 1: the query table of $\mathcal{L}$ is the infinite table whose columns are indexed by languages in $\mathcal{L}$ and rows by words (so, there are infinitely many rows). The cell corresponding to a word w and a language L in $\mathcal{L}$ is the boolean indicating whether w is in L. Thus the $\mathcal{L}$-profile of w is the row corresponding to w in the query table of $\mathcal{L}$.

▶ **Lemma 7.** *Consider a lattice of languages $\mathcal{L}$ generated by k languages. The query table of $\mathcal{L}$ has size at most 2^k.*

Indeed, there are at most 2^k different profiles with respect to $\mathcal{L}$.

▶ **Theorem 8.** *Let L in $\mathrm{Alt}(f)$. There exists a constant C such that for all $n \in \mathbb{N}$, the query table of L of order n has size at most $2^{C \cdot f(n)}$.*

The proof of Theorem 8 relies on the following lemma.

▶ **Lemma 9.** *Consider two families of languages $\mathcal{L}$ and $\mathcal{M}$. If $\mathcal{M} \subseteq \mathcal{L}$, then the size of the query table of $\mathcal{M}$ is smaller than or equal to the size of the query table of $\mathcal{L}$.*

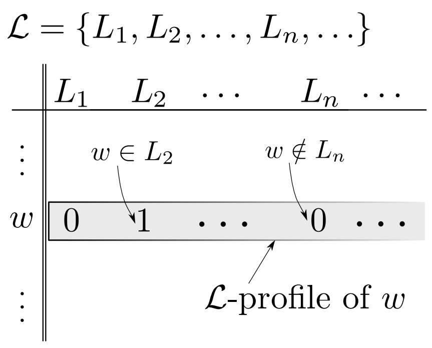
$\mathcal{L} = \{L_1, L_2, \ldots, L_n, \ldots\}$
L_1 L_2 $\cdots$ L_n $\cdots$
$w \in L_2$
$w \notin L_n$
w
0 1 $\cdots$ 0 $\cdots$
$\mathcal{L}$-profile of w

Figure 1 The query table of $\mathcal{L}$.

Proof. It suffices to observe that the query table of $\mathcal{M}$ is "included" in the query table of $\mathcal{L}$. More formally, consider in the query table of $\mathcal{L}$ the sub-table which consists of columns corresponding to languages in $\mathcal{M}$: this is the query table of $\mathcal{M}$. This implies the claim. ◀

We now prove Theorem 8. Thanks to Theorem 5, the family of left quotients of L of order n is contained in a lattice generated by a family of size at most $C \cdot f(n)$. It follows from Lemma 9 that the size of the query table of L of order n is smaller than or equal to the size of the query table of a lattice generated by at most $C \cdot f(n)$ languages, which by Lemma 7 is at most $2^{C \cdot f(n)}$.

Our lower bound apparatus is now complete: thanks to Theorem 8, to prove a lower bound on the alternating SC of a language L, it is sufficient to prove lower bounds on the size of the query tables of L.

4.3 A First Application of the Query Table Method

As a first application of our technique, we exhibit a language which has maximal (i.e., exponential) alternating SC. Surprisingly, this language is simple in the sense that it is context-free and definable in Presburger arithmetic, i.e., in first-order logic with the addition predicate.

Recall that L has subexponential alternating SC if $L \in \mathrm{Alt}(f)$ for some f such that $f = o(C^n)$ for all $C > 1$. Thanks to Theorem 8, to prove that L does not have subexponential alternating SC, it is enough to exhibit a constant $C > 1$ such that for infinitely many n, the query table of the left quotients of L of order n has size at least 2^{C^n}.

▶ **Theorem 10.** *There exists a language which does not have subexponential alternating SC, yet is both context-free and definable in Presburger arithmetic.*

Proof. Let

$$L = \left\{ u\sharp u_1 \sharp u_2 \sharp \cdots \sharp u_k \;\middle|\; \begin{array}{l} u, u_1, \ldots, u_k \in \{0,1\}^*, \\ \exists j \in \{1, \ldots, k\}, u = u_j^R \end{array} \right\}.$$

Recall that the notation u^R stands for the reverse of u defined by $u^R = u(n-1)\cdots u(0)$. Note, and this is very important here, the number of words $u_1, \ldots, u_k$ is not bounded: k is arbitrary.

It is easy to see that L is both context-free and definable in Presburger arithmetic, i.e., in first-order logic with the addition predicate (the use of reversed words in the definition of L is only there to make L context-free).

We show that L does not have subexponential alternating SC. We prove that for all n, the query table of the left quotients of L of order n has size at least 2^{2^n}. Thanks to Theorem 8, this implies the result.

Fix n. Let U be the set of all words u in $\{0,1\}^n$. It has cardinality 2^n. Consider a subset S of U. We argue that there exists a word w such that if u is in U, then the following equivalence holds:

$$w \in u^{-1}L \iff u \in S.$$

This shows the existence of 2^{2^n} different profiles with respect to the left quotients of order n, as claimed.

Let $u_1, \ldots, u_{|S|}$ be the words in S. Consider

$$w = \sharp u_1^R \sharp u_2^R \sharp \cdots \sharp u_{|S|}^R.$$

The word w clearly satisfies the claim above. ◀

5 A Hierarchy Theorem for Languages of Polynomial Alternating State Complexity

▶ **Theorem 11.** *For each $\ell \geq 2$, there exists a language L_ℓ such that:*
- *L_ℓ is in* Alt $\left(n^\ell\right)$,
- *L_ℓ is not in* Alt $\left(n^{\ell-\varepsilon}\right)$ *for any $\varepsilon > 0$.*

Consider the alphabet $\{0,1\} \cup \{\Diamond, \sharp\}$.

Let $\ell \geq 2$, and

$$L_\ell = \left\{ \Diamond^p u \sharp u_1 \sharp u_2 \sharp \cdots \sharp u_k \;\middle|\; \begin{array}{l} u, u_1, \ldots, u_k \in \{0,1\}^*, \\ k \leq p^\ell, \exists j \leq k,\ u = u_j \end{array} \right\}.$$

We note that unlike the language used for proving Theorem 10, the value of k is here bounded by p^ℓ.

Proof. We construct an alternating automaton of state complexity $O(n^\ell)$. The automaton has three consecutive phases:

1. First, a non-deterministic guessing phase while reading $\Diamond^p$, which passes onto the second phase a number j in $\left\{1, \ldots, p^\ell\right\}$.

   Formally, the set of states for this phase is $\mathbb{N}$, the initial state is 0 and the transitions are

$$\begin{array}{l} \delta(0, \Diamond) = 1 \\ \delta(k^\ell, \Diamond) = \bigvee_{j \in \{1, \ldots, (k+1)^\ell\}} j \\ \delta(p, \Diamond) = p. \end{array}$$

   The automaton for this phase has state complexity n^ℓ.

2. Second, a universal phase while reading u. For each i in $\{1, \ldots, |u|\}$, the automaton launches one copy storing the position i, the letter $u(i)$ and the number j guessed in the first phase.

   Formally, the set of states for this phase is

$$\mathbb{N} \times (\{0,1\} \cup \{\bot\}) \times \mathbb{N}.$$

The first component is the length of the word read so far (in this phase), the second component stores the letter read, where the letter $\perp$ stands for undeclared, and the last component is the number j.
The initial state is $(0, \perp, j)$. The transitions are

$$\begin{aligned}\delta((q, \perp, j), a) &= (q+1, \perp, j) \wedge (q, a, j)\\ \delta((q, a, j), b) &= (q, a, j).\end{aligned}$$

The automaton for this phase has quadratic state complexity.

3. Third, a deterministic phase while reading

$\sharp u_1 \sharp u_2 \sharp \cdots \sharp u_k.$

It starts from a state of the form (q, a, j). It checks whether $u_j(q) = a$. Localising u_j is achieved by decrementing the number j by one each time a letter $\sharp$ is read. In the corresponding u_j localising the position q is achieved by decrementing the first component by one at a time.
The automaton for this phase has quadratic state complexity.

We now prove the lower bound.

We prove that for all n, the size of the query table of L_ℓ of order $n + 2^{\frac{n}{\ell}}$ is at least 2^{2^n}. Thanks to Theorem 8, this implies that L_ℓ is not in Alt $\left(n^{\ell-\varepsilon}\right)$ for any $\varepsilon > 0$.

Fix n. Let U be the set of all words u in $\{0,1\}^n$. It has cardinality 2^n.

Observe that $\Diamond^{2^{\frac{n}{\ell}}} u \sharp u_1 \sharp u_2 \sharp \cdots \sharp u_{2^n}$ belongs to L_ℓ if and only if there exists j in $\{1, \ldots, 2^n\}$ such that $u = u_j$.

Consider any subset S of U, we argue that there exists a word w which satisfies that if u is in U, then the following equivalence holds:

$$w \in \left(\Diamond^{2^{\frac{n}{\ell}}} u\right)^{-1} L \Longleftrightarrow u \in S.$$

This shows the existence of 2^{2^n} different profiles with respect to the left quotients of order $n + 2^{\frac{n}{\ell}}$, as claimed.

Let $u_1, \ldots, u_{|S|}$ be the words in S. Consider

$$w = \sharp u_1 \sharp u_2 \sharp \cdots \sharp u_{|S|}.$$

The word w clearly satisfies the claim above. ◀

6 The Alternating State Complexity of Prime Numbers

In this section, we give lower bounds on the alternating state complexity of the language of prime numbers written in binary:

$$\textsc{Primes} = \left\{u \in \{0,1\}^* \mid \text{bin}(u) \text{ is prime}\right\}.$$

By definition $\text{bin}(w) = \sum_{i \in \{0,\ldots,n-1\}} w(i) 2^i$; note that the least significant digit is on the left.

The complexity of this language has long been investigated; many efforts have been put in finding upper and lower bounds. In 1976, Miller gave a first conditional polynomial time algorithm, assuming the generalised Riemann hypothesis [18]. In 2002, Agrawal, Kayal and Saxena obtained the same results, but non-conditional, i.e., not predicated on unproven number-theoretic conjectures [1].

The first lower bounds were obtained by Hartmanis and Shank in 1968, who proved that checking primality requires at least logarithmic deterministic space [13], conditional on number-theoretic assumptions. It was shown by Hartmanis and Berman in 1976 that if the number is presented in unary, then logarithmic deterministic space is necessary and sufficient [12]. The best lower bound from circuit complexity is due to Allender, Saks and Shparlinski: they proved unconditionally in 2001 that PRIMES is not in $\mathrm{AC}^0[p]$ for any prime p [2].

The results above are incomparable to our setting, as we are here interested in state complexity. The first and only result to date about the SC of PRIMES is due to Hartmanis and Shank in 1969:

▶ **Theorem 12** ([14]). *The set of prime numbers written in binary does not have subexponential deterministic state complexity.*

Their result is unconditional, and makes use of Dirichlet's theorem on arithmetic progressions of prime numbers. A related and stronger result has been proved by Shallit [26], which says that the deterministic automaticity of the prime numbers is not subexponential.

Hartmanis and Shank proved the following result.

▶ **Lemma 13** ([14]). *Fix $n > 1$, and consider u and v two different words of length n starting with a 1. Then the left quotients u^{-1}PRIMES and v^{-1}PRIMES are different.*

Lemma 13 directly implies Theorem 12 [14]. It also yields a lower bound of $n - 1$ on the size of the query table of PRIMES of order n. Thus, together with Theorem 8, this proves that PRIMES does not have sublogarithmic alternating SC.

▶ **Corollary 14.** *The set of prime numbers written in binary does not have sublogarithmic alternating state complexity.*

Our contribution in this section is to extend this result by showing that PRIMES does not have sublinear alternating SC, which is an exponential improvement.

▶ **Theorem 15.** *The set of prime numbers written in binary does not have sublinear alternating state complexity.*

Our result is unconditional, but it relies on the following advanced theorem from number theory, which can be derived from the results obtained by Maier and Pomerance [17]. Note that their results are more general; we state a corollary fitting our needs. Simply put, this result says that in any (reasonable) arithmetic progression and for any d, there exists a prime number in this progression at distance at least d from all other prime numbers.

▶ **Theorem 16** ([17]). *For every arithmetic progression $a + b\mathbb{N}$ such that a and b are coprime, for every N, there exists a number k such that $p = a + b \cdot k$ is the only prime number in $[p - N, p + N]$.*

We proceed to the proof of Theorem 15.

Proof. We show that for all $n > 1$, the query table of PRIMES of order n has size at least 2^{n-1}. Thanks to Theorem 8, this implies the result.

Fix $n > 1$. Let U be the set of all words u of length n starting with a 1. Equivalently, we see U as a set of numbers; it contains all the odd numbers smaller than 2^n. It has cardinality 2^{n-1}.

We argue that for all u in U, there exists a word w such that for all v in U, w is in v^{-1}PRIMES if and only if $u = v$. In other words the profile of w is 0 everywhere but on the column u^{-1}PRIMES. Let u in U; write $a = \text{bin}(u)$. Consider the arithmetic progression $a + 2^n\mathbb{N}$; note that a and 2^n are coprime. Thanks to Theorem 16, for $N = 2^n$, there exists a number k such that $p = a + 2^n \cdot k$ is the only prime number in $[p - N, p + N]$. Let w be a word such that $\text{bin}(w) = k$. We show that for all v in U, we have the following equivalence: w is in v^{-1}PRIMES if and only if $u = v$.

Indeed, $\text{bin}(vw) = \text{bin}(v) + 2^n \cdot \text{bin}(w)$. Observe that

$$|\text{bin}(vw) - \text{bin}(uw)| = |\text{bin}(v) - \text{bin}(u)| < 2^n.$$

Since p is the only prime number in $[p - 2^n, p + 2^n]$, the equivalence follows.

We constructed 2^{n-1} words each having a different profile, implying the claimed lower bound. ◀

Theorem 15 proves a linear lower bound on the alternating SC of PRIMES. We do not know of any non-trivial upper bound, and believe that there are none, meaning that PRIMES does not have subexponential alternating SC.

An evidence for this is the following probabilistic argument. Consider the distribution of languages over $\{0,1\}^*$ such that a word u in thrown into the language with probability $\frac{1}{|u|}$. It is a common (yet flawed) assumption that the prime numbers satisfy this distribution, as witnessed for instance by the prime number theorem. One can show that with high probability such a language does not have subexponential alternating SC, the reason being that two different words are very likely to induce different profiles in the query table. Thus it is reasonable to expect that PRIMES does not have subexponential alternating SC.

We dwell on the possibility of proving stronger lower bounds for the alternating SC of PRIMES. Theorem 16 fleshes out the *sparsity* of prime numbers: it constructs isolated prime numbers in any arithmetic progression, and allows us to show that the query table of PRIMES contains all profiles with all but one boolean value set to false.

To populate the query table of PRIMES further, one needs results witnessing the *density* of prime numbers, i.e., to prove the existence of clusters of prime numbers. This is in essence the contents of the Twin Prime conjecture, or more generally of Dickson's conjecture, which are both long-standing open problems in number theory, suggesting that proving better lower bounds is a very challenging objective. Dickson's conjecture reads (we use the equivalent statement given by Ribenboim in [24], called D_1):

▶ Conjecture 1 (Dickson's Conjecture). Fix b and

$$S = \{1 \leq a_1 < \cdots < a_s < b\}$$

such that there exists no prime number p which divides

$$\prod_{a \in S} (b \cdot k + a)$$

for every k in $\mathbb{N}$. Then there exists a number k such that

$$b \cdot k + a_1, b \cdot k + a_2, \ldots, b \cdot k + a_s$$

are consecutive prime numbers.

▶ **Theorem 17.** *Assuming Conjecture 1 holds true, the set of prime numbers written in binary does not have subexponential alternating state complexity.*

Proof. We show that for infinitely many $n > 1$, the query table of PRIMES of order n has size doubly-exponential in n. Thanks to Theorem 8, this implies the result.

Fix $n > 1$. As above, let U be the set of all words u of length n starting with a 1, i.e., odd numbers. For a subset $S = \{1 \leq a_1 < \cdots < a_s < b\}$ of U, let $(\lozenge)$ denote the property that there exists no prime number p which divides $\prod_{a \in S}(b \cdot k + a)$ for every k in $\mathbb{N}$.

Let S be a subset of U satisfying $(\lozenge)$. Thanks to Conjecture 1, there exists a number k such that for $a_1 \leq a \leq a_s$, the number $2^n \cdot k + a$ is prime if and only if a is in S. Let w be a word such that $\text{bin}(w) = k$. It clearly satisfies the condition above. In other words the profile of w for the columns between a_1 and a_s is 1 on the columns corresponding to S, and 0 everywhere else. For each subset S satisfying $(\lozenge)$ with the same extremal elements (a_1 and a_s) we constructed a word such that these words have pairwise different profiles.

To finish the proof, we need to explain why this induces doubly-exponentially many different profiles. For any n, the set S of odd numbers $a \in U$ such that $2^n + a$ is a prime number satisfies $(\lozenge)$. This follows from the remark that no prime number can divide both $\prod_{a \in S} a$ and $\prod_{a \in S}(2^n + a)$. Thanks to the prime number theorem estimating the proportion of prime numbers, we know that for infinitely many n the set S contains a number a_1 smaller than 2^{n-2} and a number a_s larger than $2^n - 2^{n-2}$. Now, each subset of S gives rise to a different profile, which yields doubly-exponentially many of them. ◀

Conclusion

We have developed a generic lower bound technique for alternating state complexity, and applied it to two problems. The first result is to give languages of arbitrary high polynomial alternating state complexity. The second result is to give lower bounds on the alternating state complexity of the language of prime numbers; we show that it is not sublinear, which is an exponential improvement over the previous result. However, the exact complexity is left open; we conjecture that it is not subexponential, but obtaining this result might require major advances in number theory.

We leave two questions open, motivating further research:

- Is the converse of Theorem 5 true, or in other words does the size of the query table completely characterise the alternating state complexity (as it does in the deterministic case)? We believe the answer is “no”, but proving it would require using a stronger lower bound technique to separate alternating state complexity from size of the query table.
- Can we find a notion of reduction between languages which respects the alternating state complexity, inducing a definition of completeness for alternating state complexity classes? The sequence of languages L_ℓ for $\ell \geq 2$ are good candidates for complete languages in the polynomial hierarchy.

References

1 Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. Primes is in P. *Annals of Mathematics*, 2:781–793, 2002.

2 Eric Allender, Michael E. Saks, and Igor Shparlinski. A lower bound for primality. *Journal of Computer and System Sciences*, 62(2):356–366, 2001.

3 László Babai, Peter Frankl, and Janos Simon. Complexity classes in communication complexity theory (preliminary version). In *FOCS'86*, 1986.

4 Janusz Brzozowski. Canonical regular expressions and minimal state graphs for definite events. *Symposium on Mathematical Theory of Automata*, 12:529–561, 1963.

5 Ashok K. Chandra, Dexter Kozen, and Larry J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, 1981.

6 Ashok K. Chandra and Larry J. Stockmeyer. Alternation. In *FOCS'76*, 1976.

7 Anuj Dawar and Stephan Kreutzer. Generalising automaticity to modal properties of finite structures. *Theoretical Computer Science*, 379(1-2):266–285, 2007.

8 Abdelaziz Fellah, Helmut Jürgensen, and Sheng Yu. Constructions for alternating finite automata. *International Journal of Computer Mathematics*, 35(1):117–132, 1990.

9 Nathanaël Fijalkow. The online space complexity of probabilistic languages. In *LFCS'2016*, 2016.

10 Yuan Gao, Nelma Moreira, Rogério Reis, and Sheng Yu. A survey on operational state complexity. *Journal of Automata, Languages and Combinatorics*, 21(4):251–310, 2017.

11 Ian Glaister and Jeffrey Shallit. Automaticity III: polynomial automaticity and context-free languages. *Computational Complexity*, 7(4):371–387, 1998.

12 Juris Hartmanis and Leonard Berman. On tape bounds for single letter alphabet language processing. *Theoretical Computer Science*, 3(2):213–224, 1976.

13 Juris Hartmanis and H. Shank. On the recognition of primes by automata. *Journal of the ACM*, 15(3):382–389, 1968.

14 Juris Hartmanis and H. Shank. Two memory bounds for the recognition of primes by automata. *Mathematical Systems Theory*, 3(2), 1969.

15 Richard M. Karp. Some bounds on the storage requirements of sequential machines and Turing machines. *Journal of the ACM*, 14(3), 1967.

16 Dexter Kozen. On parallelism in Turing machines. In *FOCS'76*, pages 89–97, 1976.

17 Helmut Maier and Carl Pomerance. Unusually large gaps between consecutive primes. *Transactions of the American Mathematical Society*, 322(1):201–237, 1990.

18 Gary L. Miller. Riemann's hypothesis and tests for primality. *Journal of Computer and System Sciences*, 13(3):300–317, 1976.

19 Anil Nerode. Linear automaton transformations. *Proceedings of the American Mathematical Society*, 9(4):541–544, 1958.

20 Carl Pomerance, John Michael Robson, and Jeffrey Shallit. Automaticity II: descriptional complexity in the unary case. *Theoretical Computer Science*, 180(1-2):181–201, 1997.

21 Michael O. Rabin. Probabilistic automata. *Information and Control*, 6(3):230–245, 1963.

22 Fabian Reiter. Distributed graph automata. In *LICS*, 2015.

23 Paulo Ribenboim. *The Book of Prime Number Records*. Discrete Mathematics. Springer-Verlag New York, 1996.

24 Grzegorz Rozenberg and Arto Salomaa. *Handbook of Formal Languages*. Springer-Verlag Berlin Heidelberg. Springer, 1997.

25 Jeffrey Shallit. Automaticity IV: sequences, sets, and diversity. *Journal de Théorie des Nombres de Bordeaux*, 8(2):347–367, 1996.

26 Jeffrey Shallit and Yuri Breitbart. Automaticity I: properties of a measure of descriptional complexity. *Journal of Computer and System Sciences*, 53(1):10–25, 1996.

27 Sheng Yu. State complexity of regular languages. *Journal of Automata, Languages and Combinatorics*, 6(2):221, 2001.

28 Sheng Yu. State complexity of finite and infinite regular languages. *Bulletin of the EATCS*, 76:142–152, 2002.