University of Warsaw
Faculty of Mathematics, Informatics and Mechanics

Maciej Dębski
Student no. 319428

State complexity of complementing unambiguous automata

Master's thesis
in COMPUTER SCIENCE

Supervisor:
dr Wojciech Czerwiński
Institute of Informatics

December 2017

Supervisor's statement

Hereby I confirm that the present thesis was prepared under my supervision and that it fulfils the requirements for the degree of Master in Computer Science.

Date Supervisor's signature

Author's statement

Hereby I declare that the present thesis was prepared by me and none of its contents was obtained by means that are against the law. The thesis has never before been a subject of any procedure of obtaining an academic degree. Moreover, I declare that the present version of the thesis is identical to the attached electronic version.

Date Author's signature

Abstract

Unambiguous finite automata are an interesting, but still not well-understood class of machines. They are known to significantly differ from both deterministic and nondeterministic finite automata in some aspects, but for some problems it is still not known how they behave. One of the open questions asks about the size blow-up needed for complementation. Known lower and upper bounds are wide away, and in fact not much different from these for, correspondingly, deterministic and nondeterministic automata. This is a dynamic fields though, and a new lower bound has recently been proposed. We present some properties of unambiguous automata, highlighting differences and similarities to the other classes, as well as introduce a new, quasi-polynomial solution for complementation in the special case of unary alphabet.

Keywords

automata, unambiguous, complementation, state complexity, nondeterministic

Thesis domain (Socrates-Erasmus subject area codes)

11.3 Informatics

Subject classification

Theory of computation - Regular languages

Tytuł pracy w języku polskim

Rozmiar dopełnienia dla automatów jednoznacznych

Contents

Notations

The following notations are used throughout the paper.

$\mathcal{A}, \mathcal{B}, \ldots$	written capital Latin letters denote automata
$\mathcal{A} = \langle Q_\mathcal{A}, \Sigma_\mathcal{A}, \delta_\mathcal{A}, q_{0_\mathcal{A}}, F \rangle$	a nondeterministic finite automaton is a five-tuple, where
$Q_\mathcal{A}$	set of states of automaton
$\Sigma_\mathcal{A}$	alphabet of automaton - set of input symbols
$\delta_\mathcal{A}$	transition relation, $\delta_\mathcal{A} \subseteq (Q_\mathcal{A} \times \Sigma_\mathcal{A}) \times Q_\mathcal{A}$
$q_{0,\mathcal{A}}$	starting state, $q_{0,\mathcal{A}} \in Q_\mathcal{A}$
$F_\mathcal{A}$	set of accepting states, $F_\mathcal{A} \subseteq Q_\mathcal{A}$
$a, b, a_i, \ldots$	symbols usually denoting letters - elements of alphabet Σ
w, v, u	usually denote words - sequences of letters
w_i	i-th letter of word w
L, K	usually denote languages - sets of words
Σ^*	language (set) of all finite words over alphabet Σ
$\lvert\mathcal{A}\rvert$	size of automaton, defined as it's number of states - $\lvert Q_\mathcal{A}\rvert$
$\bar{L}$	complementation of language L, set of words in Σ^* but not in L
$L(\mathcal{A})$	language of automaton - set of accepted words
$\lvert w\rvert$	length of the word w
wv	concatenation of words w, v

Introduction

This paper treats about a specific class of automata, called unambiguous automata. It is a very interesting class, which in some sense it lays between deterministic and nondeterministic automata. While unambiguous automata are notably more complex than deterministic automata, some problems for them are much easier than for nondeterministic automata. This class is also not yet well-understood. There are no good, general observations linking automaton structure with its ambiguity, and often no developed tools for dealing even with simple questions.

One of such questions is that of complexity of complementing an automaton - that is, finding an automaton that recognizes the complement of the language of a given one. This problem is trivial for deterministic automata yielding a simple solution of size equal to size of original automata, but requires exponential blow-up for nondeterministic automata. Not much is known about this problem for unambiguous automata. It has been conjectured that this can be done in polynomial time, but to our best knowledge no sub-exponential solution has been presented yet. On the other hand, best existing lower-bounds used to be only of size $O(n^2)$, although a recent paper [Ras] proposes an example which could require $O(n^{\log \log n})$ blow-up. Anyway, there still is a big gap between the established bounds.

We concentrate on this problem for a specific, simpler special case, class of automata over unary alphabet. While this may look like a great simplification, and it indeed is, best known counter-examples - for proving the lower bounds - also come from this specific case. We present a novel method which allows for complementing unary unambiguous automata in quasi-polynomial time.
In the first chapter, we introduce basic definitions from automata theory, as well as present some basic theorems. The second chapter treats about unambiguous automata, it contains the definition itself, several theorems highlighting differences from both deterministic and nondeterministic automata, and the complementation hypothesis which is of main interest in this paper. It also discusses the notion of unambiguity in other classes of machines. Chapter three focuses on unary automata - a special class of automata - over an alphabet consisting of a single letter. The main result here is a proof of existence of a normal form for such an automata. We also discuss other recent work on this matter. In Chapter four we use facts described in previous chapters to derive our

method of complementing the language of unambiguous unary automaton in $n^{O(\log n)}$ complexity. An example which proves the $O(n^2)$ lower bound for complementation is also included.

Chapter 1

Non-deterministic automata

As this paper treats about a special class of finite word automata we shall first present some general knowledge on the topic. Automata are one of the formalisms used to describe regular languages. Formally, *nondeterministic automaton* $\mathcal{A}$ is a tuple $\langle Q, \Sigma, \delta, q_0, F \rangle$, where

1. Q is a finite set of *states*
2. alphabet Σ is a finite set of symbols (*letters*)
3. $\delta \subseteq Q \times \Sigma \times Q$ is a *transition relation*
4. $q_0 \in Q$ is an *initial state*
5. $F \subseteq Q$ is a set of *accepting states*

An automaton *accepts* a word $w = a_1 \dots a_k \in \Sigma^*$, if there exist states $r_0, \dots, r_k \in Q$, with $r_0 = q_0$ and $r_k \in F$, such that for all $i \in \{1, \dots, k\}$

$$(r_{i-1}, a_i, r_i) \in \delta.$$

Such sequence of states is called a *run* of automaton $\mathcal{A}$ over word w.

An automaton is often pictured as directed graph where nodes are states and edges labelled with letters depict transition relation. The word is then accepted if we can walk through the graph from initial state to some accepting state following edges labeled with subsequent letters of the word.

The set of words accepted by the automaton is called the *language* of automaton, or language *accepted* or *recognized* by automaton and denoted $L(\mathcal{A})$.

An important special case of nondeterministic automaton is a deterministic one. A nondeterministic automaton $\langle Q, \Sigma, \delta, q_0, F \rangle$ is *deterministic* if relation δ is a function

from $Q \times \Sigma$ to Q. That is, if for each state and letter there is at most one state to which automaton may move reading this letter. Deterministic automata are much simpler than nondeterministic. For example, checking whether a deterministic automaton accepts a word requires storing only a single state in memory, starting from the initial and updating it as we process each letter, while for nondeterministic we need to store a whole set of states which automaton could reach at a given point. Several other problems which we will describe later, such as complementation or universality checking are significantly harder for nondeterministic than deterministic.
As we use automata for investigating languages, it is natural that we care most about the languages they recognize. Therefore, we say that nondeterministic automata $\mathcal{A}$ and $\mathcal{B}$ are *equivalent* if they recognize the same language. One of the most important basic theorems on automata is the one that classes of languages recognized by nondeterministic and deterministic automata are in fact equal. That is, each language recognized by some nondeterministic automaton is also recognized by some deterministic one.

Theorem 1.1
For each nondeterministic automaton $\mathcal{A}$ there exists an deterministic automaton $\mathcal{B}$ recognizing language $L(\mathcal{A})$.

Proof. Let $(A) = \langle Q, \Sigma, \delta, q_0, F\rangle$. We construct $\mathcal{B} = \langle Q_\mathcal{B}, \Sigma, \delta_\mathcal{B}, q_{0,\mathcal{B}}, F_\mathcal{B}\rangle$ as follows. Let $Q_\mathcal{B}$ be the power set $\mathcal{P}(Q)$, starting state $q_{0,\mathcal{B}}$ be singleton $\{q_0\}$ and $F_\mathcal{B}$ be defined as all sets containing any accepting state from F. Relation $\delta_\mathcal{B}$ for each $Q_i \in Q_\mathcal{B}$ and letter $a \in \Sigma$ contains transition (Q_i, a, Q_j) where

$$Q_j = \{p \in Q : (q, a, p) \in \delta, q \in Q_i\}.$$

States of $\mathcal{B}$ can be interpreted as sets of states in which $\mathcal{A}$ could be after reading given word. It is just the initial state at the beginning. Then $\mathcal{B}$ tracks all possible runs by following each possible transition of $\mathcal{A}$. Accepting states are any that contain accepting state of $\mathcal{A}$, so that $\mathcal{B}$ accepts whenever there exists accepting run in $\mathcal{A}$. □

Note that the above construction produces automaton $\mathcal{B}$ of size $2^{|\mathcal{A}|}$. The exponential blow-up is inherent to determinization, but we omit the proof now, as we prove stronger result in Example 2.3.

One of equivalent definitions of regular languages is that those are the ones that are recognized by some nondeterministic (or equivalently - thanks to Theorem 1.1 - deterministic) automaton.

1.1. Complementation

Deterministic automata can be easily complemented. That is, for a deterministic automaton $\mathcal{A}$ an automaton $\mathcal{B}$ may be constructed which accepts exactly these words which $\mathcal{A}$ rejects. This can be done simply by complementing the set of accepting states, i.e. if $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F\rangle$ then $\mathcal{B} = \langle Q, \Sigma, \delta, q_0, Q \setminus F\rangle$ accepts complementation of $L(\mathcal{A})$.

It works because for each word there is exactly one possible run. This in particular proves that complementation can be performed without increasing number of states, as $|\mathcal{B}| = |\mathcal{A}|$.

Similar approach does not work for nondeterministic automata. Consider an automaton with two states $\{q_0, q_1\}$ which for each letter $a \in \Sigma$ either stays in the same state or moves to the other, and let the only accepting state be q_0. This automaton accepts all words, but by complementing set of accepting states we still get an automaton accepting all words.

Another simple method is though possible for nondeterministic automata, thanks to Theorem 1.1. Basically one can build a deterministic automaton equivalent to given and then complement the deterministic one. This however involves building an automaton which is exponential with respect to the size of the original one. It turns out this cannot be avoided.

Example 1.2
Let p_i denote i-th prime number and let $L_n = \left\{a^k : \forall i \leq n \; p_i \mid k\right\}$. The smallest nondeterministic automaton for L_n is of size $\prod_{i=1}^{n} p_i$, while the smallest nondeterministic automaton for $\overline{L_n}$ is of size $1 + \sum_{i=1}^{n} p_i - 1$. Therefore, complementation of $\overline{L_n}$ to L_n requires exponential blow-up.

Proof. Language $\overline{L_n}$ consists of words of length not divisible by some of the first n primes. A nondeterministic automaton recognizing it may be constructed as follows. Let q_0 be the initial state, and for each p_i let $q_0, q_{i,1}, \ldots, q_{i,p_i-1}$ form a cycle - each one has a single transition over a to the next one, and the last to the first one. Therefore, the automaton consists of cycles for each prime, disjoint but for the common state q_0. All states other than the initial one are accepting. If $p_i \nmid k$, then an accepting run for a^k exists in cycle corresponding to p_i, and if k is divisible by all p_i, each run finishes in the non-accepting state q_0.

The nondeterministic automaton consisting of a cycle of length $P = p_1 p_2 \cdots p_n$, accepting only in the initial state, recognizes L_n. An accepting run over word a^P in any automaton recognizing L_n must consist of pairwise different states (but for the first and last being possibly equal) - otherwise, there would be a shorter accepting run, and therefore shorter accepted word, but there are no shorter non-empty words in L_n. Therefore, the automaton needs to have at least $P + 1 - 1 = P$ states. □

Chapter 2

Unambiguous automata

In this chapter we introduce a notion of unambiguous nondeterministic automaton and explore its properties. We present a few facts suggesting that this class is inherently different from both deterministic and nondeterministic ones. Finally, we also glimpse at definition and consequences of unambiguity for different classes of machines.

2.1. Definition and basic properties

A finite nondeterministic word automaton is said to be *unambiguous*, if for each accepted word, there exists only one accepting run. This is a semantic definition, and it is not clear how does it correspond to automaton structure. In the following sections we will present some properties of such automata. For now, let's just note some basic facts.

Lemma 2.1
Let $\mathcal{A}, \mathcal{B}$ be unambiguous automata with disjoint languages. Then there exist an unambiguous automaton recognizing $L(\mathcal{A}) \cup L(\mathcal{B})$, of size $|\mathcal{A}| + |\mathcal{B}| + 1$.

Proof. Consider a disjoint sum of states of $\mathcal{A}$, $\mathcal{B}$ with additional state q_0, the initial state. Let transition relation be defined as follows

$$\delta = \delta_{\mathcal{A}} \cup \delta_{\mathcal{B}} \cup \{(q_0, a, q) : (q_{0,\mathcal{A}}, a, q) \in \delta_{\mathcal{A}}\} \cup \{(q_0, a, q) : (q_{0,\mathcal{B}}, a, q) \in \delta_{\mathcal{B}}\}.$$

It contains all the transitions in original automata and additionally transitions from the initial state to each state in both automata which could come after their initial states. It recognizes $L(\mathcal{A}) \cup L(\mathcal{B})$ and is unambiguous as these languages are disjoint. □

Lemma 2.2
Let $\mathcal{A}, \mathcal{B}$ be unambiguous automata. There exists an unambiguous automaton recognizing $L(\mathcal{A}) \cap L(\mathcal{B})$, of size $|\mathcal{A}| \cdot |\mathcal{B}|$.

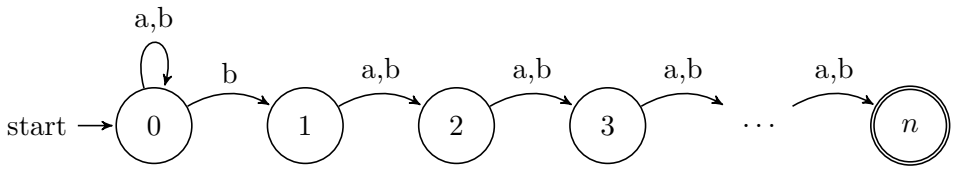
Proof. Consider a product of these automata, accepting whenever both accept. It obviously accepts $L(\mathcal{A}) \cap L(\mathcal{B})$ and is of size $|\mathcal{A}| \cdot |\mathcal{B}|$. It is unambiguous as for each word there exists at most one accepting run in $\mathcal{A}$ and in $\mathcal{B}$, therefore at most one pair of accepting runs. □

2.2. Determinization

As the unambiguous automata are a special case of nondeterministic automata, they can be converted into an equivalent deterministic automata using the usual determinization construction. However, for nondeterministic automata the construction may yield an automaton of size exponential with respect to the size of the original automaton. It turns out this is also the case for unambiguous automata.

Example 2.3 (Exponential determinization blowup)
For any given $N \geq 2$, the smallest deterministic automaton recognizing language $L_N = \{a,b\}^* b \{a,b\}^{N-1}$ is of size 2^N, while the smallest unambiguous automata recognizing this language is of size $N+1$.

Proof. Language L_N may be informally described as words such that N-th letter from the end is b. An unambiguous automaton may simply „guess" that the given b letter is N-th from the end, and then verify there are exactly $N-1$ letters afterwards.

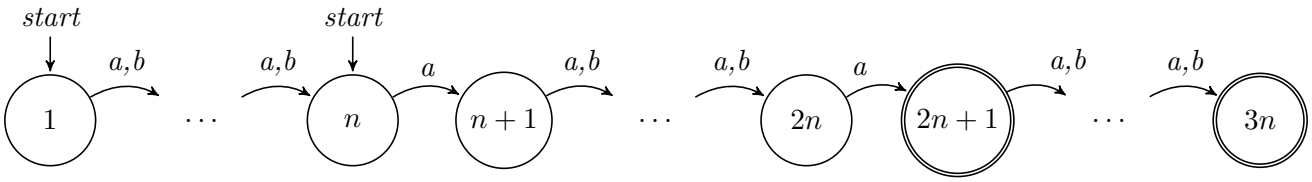


There is only one accepting run for each accepted word, as the only nondeterministic decision is made at the very beginning.
A deterministic automaton for L_N of size 2^N may be obtained from this one by determinization. This is the minimal size, as by Myhill-Nerode theorem each of 2^N words $\{a,b\}^N$ needs a different state. Indeed, let w, v be two such different words, and i be the first position on which they differ. Assume without loss of generality that $w_i = a, v_i = b$. Then for $u = a^{N-i-1}$, $wu \notin L$, while $vu \in L$. □

Similarly, also converting a nondeterministic automaton into a equivalent unambiguous, may require exponential blow-up. We will show such nondeterministic automaton in a slightly different form for easier presentation - as an automaton with multiple initial states. Such automaton may be easily transformed into one with single initial state by adding an extra state q_0 and at most $n \cdot |\Sigma|$ edges - from q_0 to q over a for each edge $p \xrightarrow{a} q$ if p is one of the initial states. The modified automaton with single initial state q_0 is equivalent. The following example is due to [Leu].

Example 2.4 (Exponential disambiguation blowup)
Consider the following $3n$-state nondeterministic automaton A_n with multiple initial states, over alphabet $\Sigma = \{a,b\}$. Transitions form a path - there is one from state q_i

to q_{i+1} for every $i \in 0, \ldots, 3n-1$. Most of these are over both a, b, but for transitions $q_n \to q_{n+1}$ and $q_{2n} \to q_{2n+1}$, which permit only letter a. The first n states are initial, and the last n are accepting.



It accepts language $L_n = \Sigma^{<n} a \Sigma^{n-1} a \Sigma^{<n}$ and has $3n = O(n)$ states, while any unambiguous automaton recognizing this language has at least $O(2^n)$ states.

The lower bound for unambiguous automaton size requires some additional tools and will be presented later, as Example 2.14.

2.3. Unambiguity checking

Another natural problem worth considering is determining whether a given nondeterministic automaton is unambiguous.

Lemma 2.5 (Unambiguity checking)
It can be decided whether a given nondeterministic automaton $\mathcal{A}$ is unambiguous in time polynomial with respect to the automaton size.

Proof. It suffices to build an automaton consisting of two copies of $\mathcal{A}$, run both on the input word and notice whenever they make take different paths. Formally, if $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$, consider nondeterministic automaton $\mathcal{B}$ with states $Q_\mathcal{B} = Q \times Q \times \{Unamb, Amb\}$. Let transition relation $\delta_\mathcal{B}$ be given as

$$\begin{aligned}\delta_\mathcal{B} = &\{((p,p,Unamb), a, (q,q,Unamb)) : (p,a,q) \in \delta, a \in \Sigma\} \cup \\ &\{((p,p,Unamb), a, (q_1,q_2,Amb)) : (p,a,q_1), (p,a,q_2) \in \delta, q_1 \neq q_2, a \in \Sigma\} \cup \\ &\{((p_1,p_2,Amb), a, (q_1,q_2,Amb)) : (p_1,a,q_1), (p_2,a,q_2) \in \delta, a \in \Sigma\}\,.\end{aligned}$$

and the let the initial state $q_{0_\mathcal{B}}$ be $(q_0, q_0, Unamb)$. Notice, that a state (p, q, Amb) is reachable over word w if and only if there exist at least two different runs in $\mathcal{A}$ over word w from initial state to respectively p and q. Setting $F_\mathcal{B} = \{(p, q, Amb) : p, q \in F\}$ and $\mathcal{B} = \langle Q_\mathcal{B}, \Sigma, \delta_\mathcal{B}, q_{0_\mathcal{B}}, F_\mathcal{B} \rangle$ we obtain an automaton of size quadratic with respect to size of $\mathcal{A}$. The language of $\mathcal{B}$ is non-empty if and only if $\mathcal{A}$ accepts some word via more than one run, that is when $\mathcal{A}$ is ambiguous. Therefore, to decide whether $\mathcal{A}$ is unambiguous we can just check $\mathcal{B}$ for emptiness. This can be simply done in polynomial time with a classic algorithm using graph traversal. □

2.4. Communication complexity

In this section we present a useful technique, due to [HSKKS] which plays a great role in some of the later proofs, in particular providing lower bounds for size of unambiguous automata recognizing given language. While it is related to communication complexity as defined in [Yao], the approach here is quite different, and no prior knowledge on the topic is required. The basic definitions are as follows.

Definition 2.6 (Communication relation)
For language L over an alphabet Σ we define the communication relation $\mathrm{Com}(L) \subseteq \Sigma^* \times \Sigma^*$ *as*

$$\mathrm{Com}(L) = \{(v, w) : vw \in L\}.$$

We will often consider restrictions of $\mathrm{Com}(L)$ *to smaller sets of words. If $A, B \subseteq \Sigma^*$, let*

$$\mathrm{Com}_{A\times B}(L) = \{(v, w) : v \in A, w \in B, vw \in L\} \subseteq \mathrm{Com}(L).$$

If A, B are finite, $\mathrm{Com}_{A\times B}(L)$ *may be naturally identified with binary matrix of size $|A| \times |B|$, with rows indexed by words from A and columns by words from B.*

Definition 2.7 (Decomposition)
A decomposition of $R \subseteq \Sigma^* \times \Sigma^*$ is a family of pairs of nonempty sets (A_i, B_i), such that $R = \bigcup A_i \times B_i$. If additionally $A_i \times B_i$ are pairwise disjoint, the decomposition is called unambiguous. An unambiguous decomposition always exists - we may consider family (x_i, y_i) for each $(x_i, y_i) \in R$ - but may be infinite.

Definition 2.8 (Unambiguous complexity)
Unambiguous complexity $\text{unamb-comp}(R) \in \mathbb{N} \cup \{\infty\}$ *for $R \subseteq \Sigma^* \times \Sigma^*$ is a minimal size of an unambiguous decomposition of R.*

Next we will see, how unambiguous decompositions of $\mathrm{Com}(L)$ are related to unambiguous automata recognizing L.

Lemma 2.9 (Unambiguous complexity and unambiguous automata)
Let L be a language over Σ. If an unambiguous automaton $\mathcal{A}$ recognizes L, then

$$\text{unamb-comp}(\mathrm{Com}(L)) \leq |\mathcal{A}|.$$

Proof. Let $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F\rangle$. For $q \in Q$ define

$$L_{q_0,q} - \text{words accepted by } \langle Q, \Sigma, \delta, q_0, \{q\}\rangle,$$

$$L_{q,F} - \text{words accepted by } \langle Q, \Sigma, \delta, q, F\rangle.$$

So those are sets of words over which automaton $\mathcal{A}$ could move from q_0 to q or from q to any state from F, respectively. Observe that family $\{(L_{q_0,q}, L_{q,F})\}_{q\in Q}$ is an unambiguous decomposition of $Com(L)$. Indeed, if $(v, w) \in Com(L)$, so $vw \in L$, there exists a single accepting run $(q_0 = r_0, r_1, \ldots, r_{|v|}, \ldots, r_{|vw|} \in F)$ for vw, so $r_{|v|}$ is a unique state such that $v \in L_{q_0,r_{|v|}}$ and $w \in L_{r_{|v|},F}$. Therefore, there exists an unambiguous decomposition of $\mathrm{Com}(L)$ of size $|\mathcal{A}|$, so $|\mathcal{A}| \geq \text{unamb-comp}(\mathrm{Com}(L))$. □

Let's note a simple property of complexity.

Observation 2.10 (Monotonicity of complexity)
For any language L and languages $A \subseteq C, B \subseteq D$

$$\text{unamb-comp}(\text{Com}_{A\times B}(L)) \leq \text{unamb-comp}(\text{Com}_{C\times D}(L)).$$

In particular,

$$\text{unamb-comp}(\text{Com}_{A\times B}(L)) \leq \text{unamb-comp}(\text{Com}(L)).$$

Proof. It suffices to see that each unambiguous decomposition of $\text{Com}_{C\times D}(L)$ - a set $\{(C_i, D_i)\}$, restricted to $A \times B$ as $\{(C_i \cap A, D_i \cap B)\}$ is an unambiguous decomposition of $\text{Com}_{A\times B}(L)$. □

We will also need the following simple result from linear algebra.

Lemma 2.11 (Rank subadditivity)
Let A, B be matrices of size $n \times k$. Then

$$\text{rank}(A + B) \leq \text{rank}(A) + \text{rank}(B).$$

Proof. Rank of matrix is equal to the dimension of vector space generated by set of its columns. Vector space generated by columns of $A + B$ is contained in this generated by set of columns of both A and B, which is of dimension at most $\text{rank}(A) + \text{rank}(B)$. □

We may now proceed to the main result of this chapter - lower bound for unambiguous complexity, and therefore size of unambiguous automata.

Lemma 2.12 (Lower bound for unambiguous complexity)
Let L be a language over Σ and A, B be finite languages over Σ. Then

$$\text{rank}(\text{Com}_{A\times B}(L)) \leq \text{unamb-comp}(\text{Com}_{A\times B}(L)).$$

Proof. Observe that each decomposition of $\text{Com}_{A\times B}(L)$ is finite, as this relation has only finitely many elements. As noted before, we identify $\text{Com}_{A\times B}(L)$ with a binary matrix of size $|A| \times |B|$. Let $F = \{(A_i, B_i)\}_{i=1}^{N}$ be a minimal unambiguous decomposition of $\text{Com}_{A\times B}(L)$. A pair (A_i, B_i) from a decomposition may be identified with a matrix M_i of the same size too - with entry being one if and only if its row's index is in A_i, and column's index in B_i. Then, if the decomposition is unambiguous,

$$\text{Com}_{A\times B}(L) = \sum_{i=1}^{N} M_i.$$

Where the sum is understood as a sum of matrices. Therefore, thanks to rank subadditivity (2.11),

$$\text{rank}(\text{Com}_{A\times B}(L)) \leq \sum_{i=1}^{N} \text{rank}(M_i).$$

But each M_i has rank one, as all its non-zero rows are identical. So

$$\operatorname{rank}(\operatorname{Com}_{A\times B}(L)) \leq \sum_{i=1}^{N} 1 = N = \text{unamb-comp}(\operatorname{Com}_{A\times B}(L)),$$

as F is minimal. □

The following statement simply joins results of Lemma 2.12 and Lemma 2.9, and will be our main tool in proving lower bounds for unambiguous automata size.

Corollary 2.13
If for some $L \subseteq \Sigma^*$ and finite $A, B \subseteq \Sigma^*$, $\operatorname{rank}(\operatorname{Com}_{A\times B}(L)) = n$ then L cannot be recognized by unambiguous automaton of size smaller than n.

Let the following example serve as illustration of this technique.

Example 2.14
Any unambiguous automaton recognizing language $L_n = \Sigma^{<n} a \Sigma^{n-1} a \Sigma^{<n}$, where $\Sigma = \{a, b\}$ (see Example 2.4), has at least $\Theta(2^n)$ states.

Proof. Consider words $x, y \in \Sigma^n, x, y \neq b^n$. Word xy is of length $2n$, so it belongs to L_n if an only if is of the form $\Sigma^i a \Sigma^{n-1} a \Sigma^{n-i-1}$ for some $i \in \{0, \ldots, n-1\}$, which is when $x_i = y_i = a$. Let $A_x = \{i : x_i = a\}$ and similarly for y. Therefore, $xy \in L_n \iff A_x \cap A_y \neq \varnothing$.

Let $X = \Sigma^n \setminus \{b^n\}$ and $M = \operatorname{Com}_{X\times X}(L)$. By rank subadditivity (Lemma 2.11),

$$\operatorname{rank}(M) + \operatorname{rank}(\mathbf{1}) \geq \operatorname{rank}(\mathbf{1} - M),$$

where $\mathbf{1}$ is matrix of all ones, of rank 1. We will prove $N := \mathbf{1} - M$ is of full rank. Notice that $x \mapsto A_x$ is a bijection of X to family of non-empty subsets of $\{0, \ldots, n-1\}$, and denote the inverse by $S \mapsto x_S$. Now $N_{x_S, x_T} = 1 \iff S \cap T = \varnothing$. Assume some linear combination of rows of N is zero,

$$\sum_{S \subseteq \{0,\ldots,2^n-1\}} a_{x_S} v_{x_S} = [0, 0, \ldots, 0].$$

For each coordinate $y = x_T \in X$, we then have

$$\sum_{S \subseteq \{0,\ldots,2^n-1\}} a_{x_S} \cdot [S \cap T = \varnothing] = 0,$$

$$\sum_{S \subseteq -T} a_{x_S} = 0,$$

so any sum of a_{x_S} is zero, so all coefficients need to be zero for linear combination to vanish, which proves rows of N are linearly independent. Therefore

$$\operatorname{rank}(M) \geq \operatorname{rank}(N) - 1 = 2^n - 2,$$

and by corollary 2.13, any unambiguous automata recognizing L has at least $2^n - 2 = \Theta(2^n)$ states. □

2.5. Universality and equivalence problems

Universality, equivalence and containment testing for general nondeterministic automata are known to be computationally expensive problems. More specifically, all of them are PSPACE-complete (see ex. [HRS]). However, it turns out that for unambiguous automata they become much simpler, and can be solved in polynomial time. This is a strong suggestion that unambiguous automata may be inherently simpler than nondeterministic. This section is based on [StHu].

Lemma 2.15 (Difference bound)
Let $K \subsetneq L$ be languages such that the shortest word $w \in L \setminus K$ is of length n. Then if some unambiguous automaton for K has n_1 states and some unambiguous automaton for L has n_2 states, there must be $n_1 + n_2 \geq n + 1$.

Proof. Let $w = a_1 a_2 \cdots a_n$ be the shortest word in $L \setminus K$. Consider matrix $M_K \in M_{n+1 \times n+1}$ obtained from $Com(K)$ by restricting to rows corresponding to prefixes of word w i.e. $\{\varepsilon, a_1, a_1 a_2, \ldots, a_1 a_2 \cdots a_n\}$, and columns corresponding to suffixes of w i.e. $\{a_1 \cdots a_n, a_2 \cdots a_n, \ldots, a_n, \varepsilon\}$. Similarly, consider matrix M_L obtained from $Com(L)$ via the same procedure. As unambiguous complexity is a lower bound for automaton size (Lemma 2.9) and thanks to the inequality between rank and unambiguous complexity (Lemma 2.12), $n_1 \geq \operatorname{rank}(M_K), n_2 \geq \operatorname{rank}(M_L)$. Therefore, by rank subadditivity (Lemma 2.11),

$$n_1 + n_2 \geq \operatorname{rank}(M_K) + \operatorname{rank}(M_L) = \operatorname{rank}(-M_K) + \operatorname{rank}(M_L) \geq \operatorname{rank}(M_L - M_K). \quad (2.16)$$

However, observe that all the elements at the diagonal of matrix $M_L - M_K$ correspond to the same word w, which is in $L \setminus K$, therefore diagonal of M_K is all zeros and diagonal of M_L is all ones. Moreover, as w is the shortest word in $L \setminus K$, and all cells above the diagonal corresponds to words $v_{ij} = a_1 \cdots a_i + a_j \cdots a_n$ for some $i < j$, so to words shorter than w, those cells are the same in M_K and M_L. Indeed, as $|v_{ij}| < |w|$, then $v_{ij} \notin L \setminus K$, so $v_{ij} \in L \iff v_{ij} \in K$.

Therefore, matrix $M_L - M_K$ has only zeros above the diagonal and ones on the diagonal, which means it is of full rank $n + 1$, which ends the proof thanks to inequality 2.16. □

Corollary 2.17 (Difference witness)
If two unambiguous automata $\mathcal{A}, \mathcal{B}$, each having at most n states, recognize languages K, L respectively such that $K \subsetneq L$, then there exists a word of length at most $2n - 1$ in $L \setminus K$.

Proof. If there were no such word, the shortest word in $L \setminus K$ (which is non-empty, as $L \neq K$), must be of length at least $2n$, so by Lemma 2.15 sum of number of states of $\mathcal{A}$ and $\mathcal{B}$ should at least $2n + 1$, which is a contradiction with assumption both have not more than n states. □

Lemma 2.18
Given an unambiguous automaton $\mathcal{A}$ and $n \in \mathbb{N}$, a number $N_{\mathcal{A}}$ of words of length at most n, accepted by $\mathcal{A}$, can be computed in time polynomial with respect to n and $k := |\mathcal{A}|$.

Proof. Indeed, number of runs of given length between any pair of states can be computed in polynomial time. Let $A_{\leq n}$ be a $k \times k$ matrix, such that a cell corresponding to pair of states (p, q) contains the number of runs from p to q of length at most n. Note that

$$A_{\leq n+1}(p, q) = \sum_{r \in Q} A_{\leq n}(p, r) \cdot A_{\leq 1}(r, q),$$

where $A_{\leq 1}(r, q)$ is just a number of letters over which the automaton passes from r to q (plus one if $r = q$), which can be simply computed from the transition relation. So $A_{\leq n+1} = A_{\leq n} \cdot A_{\leq 1}$ (matrix multiplication), so by induction $A_{\leq n} = A^n_{\leq 1}$. Therefore, we can compute a number of runs of length at most n between any pair of states in polynomial time, or more specifically $O(k^3 \cdot \log n \cdot n)$, where the first factor is matrix multiplication, and the second is exponentiation-by-squaring, and the last takes into account the cost of multiplying the numbers which may be exponentially large with respect to n, so may be represented with $O(n)$ digits. Similarly a number of all *accepting* runs over words of length at most n may be computed as a sum $N = \sum_{q \in F} A_{\leq n}(q_0, q)$. Now, as thanks to unambiguity each accepted word has exactly one accepting run, we claim that N is equal to $N_{\mathcal{A}}$. □

Lemma 2.19 (Exact containment testing)
Given an unambiguous automata $\mathcal{A}, \mathcal{B}$ such that $L(\mathcal{A}) \subseteq L(\mathcal{B})$ it can be checked whether $L(\mathcal{A}) = L(\mathcal{B})$ in time polynomial with respect to $n = |\mathcal{A}| + |\mathcal{B}|$.

Proof. Lemma 2.17 implies that the languages of automata $\mathcal{A}$ and $\mathcal{B}$ are equal if and only if they are equal for words of size at most $2n$. First, we can compute number of words of lengths $\leq 2n$ accepted by $\mathcal{A}$ (denoted $N_{\mathcal{A}}$) and corresponding number for $\mathcal{B}$ (denoted $N_{\mathcal{B}}$). Thanks to Lemma (2.18), this can be done in time polynomial with respect to n. Now, as $L(\mathcal{A}) \subseteq L(\mathcal{B})$, each word accepted by $\mathcal{A}$ is also accepted by $\mathcal{B}$, so $N_{\mathcal{A}} = N_{\mathcal{B}}$ if and only if the languages are equal for words of size at most $2n$, if and only if $L(\mathcal{A}) = L(\mathcal{B})$. □

Corollary 2.20
If $\mathcal{A}, \mathcal{B}$ are unambiguous automata over alphabet Σ and $L = L(\mathcal{A}), K = L(\mathcal{B}), n = |\mathcal{A}| + |\mathcal{B}|$, the following properties may be checked in time polynomial with respect to n.

1. $L \subseteq K$
2. $L = K$
3. $L = \Sigma^*$

Proof.

1. Let $\mathcal{C}$ be the unambiguous automaton for $L \cap K$, as described in Lemma 2.2. Obviously, $L \cap K \subseteq K$, therefore by Lemma 2.19 we can check whether $L \cap K = K$, which holds if and only if $L \subseteq K$. The procedure can be done in time polynomial with respect to $|\mathcal{C}| = n^2$, so polynomial w.r.t. n as well.

2. It suffices to use the previous point twice, as $L = K \iff L \subseteq K \wedge K \subseteq L$.

3. This is trivial from the previous point, but also can be done a bit more efficiently - using directly Lemma (2.19) with $\mathcal{B}$ being a single-state automaton accepting every word.

□

2.6. Unambiguity in other automata classes

The notion of unambiguity can be extended to a wider class of similar nondeterministic devices. Consider any device that accepts some of the possible inputs via some kind of run. The device is called unambiguous if for each accepted input there is only one accepting run. This general description can be simply applied to numerous types of automata as well as Turing machines, which leads to many interesting problems, including quite a few unsolved ones. We shall present here an overview of results for different classes.

1. Turing machines
   Unambiguous Turing machines have the same expressive power as nondeterministic ones, as deterministic and nondeterministic have the same expressive power. The problem of checking whether a given Turing machine is unambiguous is undecidable, as there exists a simple reduction from emptiness problem - M is empty if and only if M extended to accept every word via an additional trivial one-state loop is unambiguous.
   Unambiguous counterparts of well-known time or space limited complexity classes are considered as well. For example, while obviously L ⊆ UL ⊆ NL and P ⊆ UP ⊆ NP, where U denotes the relevant unambiguous class, it is unknown whether any of the inclusions is proper. Unambiguous log-space is especially interesting, as results stating that NL/poly = UL/poly [RA] and that planar reachability [BTV] is in UL suggest that UL and NL may be equal too.

2. Infinite word automata
   Deterministic and nondeterministic Büchi automata differ in expressive power. It turns out however, that the class of languages recognized by unambiguous Büchi automata is equal to the one for nondeterministic [Arn].

3. Transducers
   Unambiguous transducers are closely related to the most commonly studied functional transducers. While unambiguous transducer is required to have at most

one accepting run for each word, functional one only needs to produce the same output over each of (possibly many) accepting runs. However, it turns out that any functional transducer can be transformed into an unambiguous one (while not into a deterministic one), but the process involves exponential blowup [Web].

4. Infinite tree automata
   Deterministic and nondeterministic Büchi automata for trees have different expressive power, as this is already the case even for words. However, in this setting unambiguous automata differ in expressive power from both deterministic and nondeterministic, as described in [CLNW]. Question whether it is decidable if the infinite tree language is accepted by an unambiguous automata is currently open, with some recent partial results as [BiSk].

5. Tropical automata
   Unambiguity is significant for tropical automata, as the two studied kinds of automata - $(\min, +)$ and $(\max, +)$ coincide over unambiguous automata. Indeed, the output is defined as respectively, minimum or maximum of total weight over all accepting runs, which is exactly the same if there is only one accepting run. Interestingly, it turns out that each language that is accepted by both $(\min, +)$ and $(\max, +)$ automaton is also accepted by some unambiguous one, as proved in [LoMa]

2.7. Complementation

Not much seems to be known about the complexity of the problem of complementation of an unambiguous automata in a general case. While it can be simply solved by determinization and complementing the deterministic automaton - by complementing the set of accepting states - this may result in an exponential blow-up of number of states. It has been conjectured though in [Col], that it could always be done with only polynomial growth.

Conjecture 2.21 (Unambiguous polynomial complementation)
There exists a constant k such that for each unambiguous automaton of size n recognizing language L, there exists an unambiguous automaton of size $O(n^k)$ recognizing $\bar{L}$.

Some motivation behind this hypothesis is that universality and equality testing for unambiguous automata can be done in polynomial time (Corollary 2.20). If complementation was polynomial, these results would be obvious - one could complement and possibly intersect automata and check for emptiness - but without this result the proof, as seen above, is not trivial.
A possibly weaker conjecture is considered as well, in which we do not require for the complementing automaton to be unambiguous.

Conjecture 2.22 (Nondeterministic polynomial complementation)
There exists a constant k such that for each unambiguous automaton of size n recognizing language L, there exists an nondeterministic automaton of size $O(n^k)$ recognizing $\bar{L}$.

To our best knowledge, no sub-exponential upper bounds for complexity of complementation have been given to date in either cases. As for the lower bound, for quite a long time the best estimate, coming from a more specific setting of unary languages (see Chapter 3), was equal to $O(n^2)$, as described in Example 4.3.2. Recently however a paper [Ras] was published which claims super-polynomial $O(n^{\log \log n})$ bound. While this would obviously void the hypotheses as stated, the general problem of finding an exact state complexity of complementation remains open.

Chapter 3

Unary alphabet

This chapter is devoted to a more specific case of *unary languages* - languages over an alphabet consisting of a single letter, $\Sigma = \{a\}$. We will also call automata over such alphabet *unary*. While this undoubtedly is a much simpler setting, the problems we consider pose a difficulty even in this situation. We hope that understanding them well in the constrained case may help also in the full generality.

Observation 3.1 (Unary words identified with numbers)
There exists a single word w in Σ^* for each length $|w| \in \mathbb{N}$, so unary words may be identified with natural numbers, and unary languages with subsets of natural numbers.

Because of the identification, we will often say ex. that a language includes some numbers, or that an automaton accepts numbers.

3.1. Normal form

Notice first that an automaton over unary language may be seen as simple directed graph, with additional labels on nodes denoting whether the corresponding state is accepting and whether it is the initial state. In particular, edge labels are redundant, as anyway there is only a single letter in the alphabet. We will make several observations on the structure of the graph.

Lemma 3.2 (Unary deterministic automaton)
The transition function of every unary deterministic finite automaton is of form $q_0 \to q_1 \to \ldots \to q_k$ where either all q_i are pairwise different or $q_k = q_j$ for some j. Informally, each unary deterministic finite automaton consists of a „chain", possibly with a cycle on the end.

Proof. As each node has at most one outgoing edge (which is the definition of determinism), we can traverse the graph starting from the initial state, up to the point we reach already visited state q. The path from initial state to state q is the chain and the patch from q to q again is the cycle. Note the two degenerated cases - q may be the initial state, then the chain is empty and graph is simply a single cycle, or there is no outgoing

edge from some node, then the graph is only a chain and the language of the automaton is finite. □

Next we would like to provide a similar simple form for nondeterministic automata. While in general each graph corresponds to an automaton, so we may get arbitrarily complex graphs, it turns out they may be transformed into a normal form preserving the recognized language. We will first present the normal form, due to [Chr], and then the methods for obtaining it.

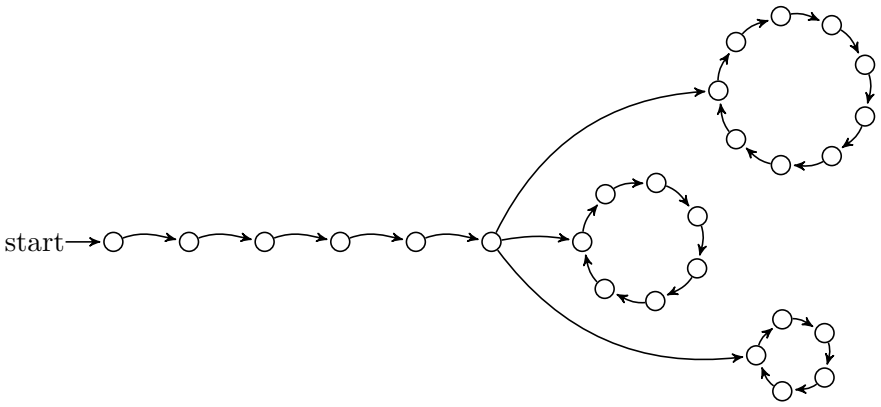
Definition 3.3
A unary automaton $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$*, where* $\Sigma = \{a\}$*, is in the normal form if it has the following properties:*

(a) $Q = \{q_0, \ldots, q_m\} \cup C_1 \cup \ldots \cup C_k$,
where all the sets are pairwise disjoint and $C_i = \{p_{i,0}, \ldots, p_{i,m_i}\}$.

(b)
$$\begin{aligned}\delta = &\{(q_i, a, q_{i+1}), i \in \{0, \ldots, m-1\}\} \cup \\ &\{(q_m, a, p_{i,0}), i \in \{0, \ldots, k\}\} \cup \\ &\{(p_{i,j}, a, p_{i,j+1}), i \in \{0, \ldots, k\}, j \in \{0, \ldots, m_i\}\},\end{aligned}$$
where $(j+1)$ *is computed modulo* m_i.

The set of accepting states may be arbitrary.

Informally, the automaton consists of an m-states chain, splitting into k disjoint cycles, as shown on an example below.



We will denote such automaton $[A, m; A_1 \bmod m_1, A_2 \bmod m_2, \ldots, A_k \bmod m_k]$, where m is the length of the chain, $A \subseteq \{0, \ldots, m-1\}$ is the set of numbers accepted by the chain, m_i are lengths of the cycles, and A_i is the set of numbers in $\{0, \ldots, m_i - 1\}$, denoting accepting states on the corresponding cycle. Note that such automata accepts words w such that

$$w \in A \text{ or } \exists_i (w - m) \bmod m_i \in A_i$$

We will call the sets $A_1, \ldots, A_k$ *residua sets* and numbers m_i *periods.*
We should now prove that each unambiguous unary automaton can be effectively transformed to the normal form. This proof bases on [JDR].

Theorem 3.4
For each unambiguous unary automaton $\mathcal{A}$ there exists an unambiguous unary automaton $\mathcal{B}$ in a normal form, such that $L(\mathcal{A}) = L(\mathcal{B})$ and $|\mathcal{B}| \leq O(|\mathcal{A}|)$.

Proof. We gradually transform $\mathcal{A}$ into $\mathcal{B}$ in a normal form. First, remove from $\mathcal{A}$ any states from which no accepting state is reachable. That obviously does not change the language of $\mathcal{A}$. Let G be the graph for $\mathcal{A}$ and $H = \langle H_V, H_E \rangle$ be a graph of strongly connected components of G. That is, nodes of H form a division of nodes of G, such that each node of H is a strongly connected component of G and there is an edge between $h_1, h_2 \in H_V$ if and only if any node in component h_2 is reachable from any node in component h_1. H is a direct acyclic graph - if there was a cycle, all of the nodes on it form in fact a single strongly connected component. We call a node $h \in H$ a „super node" if it corresponds to a strongly connected component of more than one node in G. Note two important properties of H resulting from the fact that $\mathcal{A}$ is unambiguous:

1. Each super node $h \in H_V$ is a simple cycle. Indeed, if there were two simple paths π_1, π_2 in G from some g_1 to g_2 belonging to h, then as each state in $\mathcal{A}$ is reachable from initial and can reach some accepting state, there would be a word w accepted by a run containing g_1 and g_2, in this order. But as h is a strongly connected component, there is a path π from g_2 to g_1 too, so there are two different cycles from g_1 to itself - $\pi_1\pi$ and $\pi_2\pi$. Let m be the length of the first and k of the second, then $w + mk$ could be accepted by at least two runs, which contradicts unambiguity. One of them would be the run of w with added k loops on the first cycle and the other with m loops on the other.

2. Each path from root to leaf node in H has at most one super node. Indeed, if there were two, one being a cycle of length m and the other of length k, and w was a word accepted by such path, then $w + mk$ could be accepted by at least two runs, just as in the previous point.

Let $\{h_1, \ldots, h_r\}$ be all the super nodes in H_V and $\{l_1, \ldots, l_r\}$ be lengths of the cycles that nodes form. Let L_i are the words in L which are accepted by a run that contains a node from h_i. Thanks to the property 2 they are pairwise disjoint. Let l be the length of the longest path in H. Then L_i may be recognized by a deterministic automaton (see Lemma 3.2) with chain of length at most l and cycle of length l_i. The limit on chain length is because there is not more than l states in the run outside of h_i and the ones inside h_i are already included in the cycle. We should in fact build the deterministic automaton in such a way that the chain consists of precisely l states. This could be done iterating the following procedure. Let p be the last state in the chain, before the cycle, q be the state where chain joins the cycle, and r be next, after q, state in the cycle. Then by duplicating q to q' and replacing transition $p \to q$ with $p \to q'$ and $q' \to r$ we obtain

an equivalent automaton with one state longer chain.
We can now merge those deterministic automata into a single unambiguous in normal form, simply by merging the chains and then splitting into r cycles. This automaton recognizes $\bigcup L_i$, which is almost $L(\mathcal{A})$, but for the words which are accepted by runs avoiding all super nodes. But these are of length at most l, so can be easily accounted for by making some of the states in the chain accepting.
It just remains to notice that $\mathcal{A}$ is of size at least $l + \sum l_i$, while the obtained automaton in normal form is of size at most $l + \sum l_i$. □

Similar, but slightly weaker result holds also for nondeterministic automata. This is historically earlier result, from original Chrobak's paper [Chr], but the proof is somewhat more complicated, and won't be presented here, as the result is not used in following considerations.

Theorem 3.5
For each unary nondeterministic automaton $\mathcal{A}$ there exists a unary nondeterministic automaton $\mathcal{B}$ in a normal form, such that $L(\mathcal{A}) = L(\mathcal{B})$ and $|\mathcal{B}| = |\mathcal{A}|^2 + |\mathcal{A}|$.

3.2. Unambiguity criteria

The normal form allows us to specify more concrete criteria for automaton unambiguity. As for each word a nondeterministic decision is only made at most once, at the end of the chain, when going into one of the cycles, the automaton in normal form is unambiguous if and only if no two cycles accept the same word. Therefore the following lemmas hold.

Lemma 3.6
An automaton $[A, m; A_1 \bmod m_1, A_2 \bmod m_2, \ldots, A_k \bmod m_k]$ is unambiguous if and only if the sets $\{A_i + m_i\mathbb{N}\}, \{A_j + m_j\mathbb{N}\}$ are pairwise disjoint for each $i \neq j$.

Note that in particular the last condition does not depend on the length of the chain m. We are now going to further reformulate the condition.

Lemma 3.7
If $[A, m; A_1 \bmod m_1, A_2 \bmod m_2, \ldots, A_k \bmod m_k]$ is an unambiguous unary automaton in the normal form then each pair m_i, m_j is not coprime.

Proof. Suppose that some m_i, m_j are coprime. Note first that $A_i \cap A_j = \varnothing$. Indeed, otherwise $m+a$ would be accepted by two runs for $a \in A_i \cap A_j$. Let then $a_i \in A_i, a_j \in A_j$ and assume without loss of generality $a_j > a_i$. There exist integers k, l such that

$$m_i k + m_j l = \gcd(m_i, m_j) = 1.$$

Therefore

$$m_i k(a_j - a_i) + m_j l(a_j - a_i) = a_j - a_i.$$

Let then $k' = k(a_j - a_i)$ and $l' = l(a_j - a_i)$, so that

$$m_i k' + m_j l' = a_j - a_i,$$

$$a_i + m_i k' = a_j - m_j l'.$$

By adding $m_i m_j$ to both sides several times, we can make both of them positive.

$$a_i + m_i k' + c m_i m_j = a_j - m_j l' + c m_i m_j > 0,$$

$$a_i + m_i(k' + c m_j) = a_j + m_j + (c m_i - l').$$

The equality shows that word of such length can be accepted by state a_i on cycle m_i as well as by state a_j on cycle m_j. This leads to contradiction with unambiguity by assumption that m_i, m_j are coprime. □

The above condition is necessary, but not sufficient. Below we provide an equivalent condition using similar proof techniques.

Lemma 3.8
If $[A, m; A_1 \bmod m_1, A_2 \bmod m_2, \ldots, A_k \bmod m_k]$ *is an unary automaton in the normal form it is unambiguous if and only if for each* $i \neq j$

$$A_i \bmod \gcd(m_i, m_j) \cap A_j \bmod \gcd(m_i, m_j) = \varnothing.$$

Proof. Let $d = \gcd(m_i, m_j)$. We know already that unambiguity is equivalent to condition from Lemma 3.6, so it suffices to prove that

$$\{A_i + m_i\mathbb{N}\} \cap \{A_j + m_j\mathbb{N}\} = \varnothing \iff A_i \bmod d \cap A_j \bmod d = \varnothing.$$

This is equivalent to

$$\exists_{a_i \in A_i, a_j \in A_j, k, l \in \mathbb{N}} a_i + m_i k = a_j + m_j l \iff \exists_{a_i \in A_i, a_j \in A_j} a_i \equiv a_j \mod d.$$

In fact, a stronger equivalence holds for each a_i, a_j.

$$\exists_{k, l \in \mathbb{N}} a_i + m_i k = a_j + m_j l \iff a_i \equiv a_j \mod d.$$

We shall prove the two implications separately

- „⇒”
  As $d \mid m_i$ and $d \mid m_j$ then $a_i + m_i k = a_j + m_j l$ implies $a_i \equiv a_j \mod d$.
- „⇐”
  Assume without loss of generality $a_j > a_i$. There exist $m, n \in \mathbb{Z}$ such that

  $$m_i m + m_j n = \gcd(m_i, m_j) = d$$

  . As $a_i \equiv a_j \mod d$, we have $a_j - a_i = dc$ for some $c \in \mathbb{N}$. Then

  $$m_i m c + m_j n c = a_j - a_i,$$

so

$$a_i + m_i mc = a_j + m_j nc.$$

Adding $m_i m_j$ several times to both side so that they are positive we obtain

$$a_i + m_i(mc + c'm_j) = a_j + m_j(nc + c'm_i) > 0.$$

Setting $k = mc + c'm_j, l = nc + c'm_i$ we obtain the thesis.

□

Note that for unambiguous automaton the residua sets needn't be disjoint modulo all divisors of cycle lengths. Consider automaton $[0; \{0\} \bmod 30, \{3, 5\} \bmod 15]$. It is unambiguous and the sets are disjoint modulo $\gcd(30, 15) = 15$. However, for divisors of 15 this does not hold, as 5 is equivalent to 0 modulo 5 and 3 is equivalent to 0 modulo 3. In particular, it is not always true that there is be a prime (dividing periods) modulo which the residua sets are disjoint. This is however true in a special case.

Lemma 3.9
If $[A, m; A_1 \bmod m_1, A_2 \bmod m_2, \ldots, A_k \bmod m_k]$ *is an unary automaton in the normal form and* $|A_1| = \ldots = |A_k| = 1$ *it is unambiguous if and only if for each* $i \neq j$ *there exists a prime* $p_{ij} \mid \gcd(m_i, m_j)$ *such that*

$$A_i \bmod p_{ij} \cap A_j \bmod p_{ij} = \varnothing.$$

Proof. Obviously, if the sets are disjoint modulo some divisor of $\gcd(m_i, m_j)$, they are disjoint modulo $gcd(m_i, m_j)$ as well. Now, assuming that automaton is unambiguous, from the Lemma 3.8 we know that residua sets need to be pairwise disjoint modulo greatest common divisors. However if all have only a single element, say $A_i = \{a_i\}$ then two are disjoint modulo n if and only if $a_i \not\equiv a_j \mod \gcd(m_i, m_j)$. However this is the case if and only if there exists a prime such that $a_i \not\equiv a_j \mod p_{ij}$. If this weren't the case for all prime divisors of $\gcd(m_i, m_j)$ the two residua would be also equal modulo $\gcd(m_i, m_j)$. □

3.3. Determinization

Determinization of unary unambiguous automata was extensively studied in [Okh]. We present here some of the results of this and previous works for reference. First consider a following simple method of determinization of automaton in normal form.

Lemma 3.10
For nondeterministic automaton $\mathcal{A} = [A, m; A_1 \bmod m_1, \ldots, A_k \bmod m_k]$ *there exists an equivalent deterministic automaton of size* $m + \operatorname{lcm}(m_1, m_2, \ldots, m_k)$.

Proof. Consider a deterministic automaton with chain of length m and cycle of length $l = \operatorname{lcm}(m_1, m_2, \ldots, m_k)$. By defining the set of accepting states accordingly it can be

made equivalent to $\mathcal{A}$. First, let set of accepting states in the chain be the same as in $\mathcal{A}$. Then, for each A_i, m_i add states $A_i, A_i + m_i, A_i + 2m_i, \ldots, A_i + \left(\frac{l}{m_i} - 1\right) m_i$ on the cycle to the set of accepting states. For any given i, thanks to the fact that every m_i divides l, the cycle will now accept all the words which residue modulo m_i is in A_i. As this is the case for each i, the automaton will be equivalent to $\mathcal{A}$. □

Therefore, if a nondeterministic automaton has n states then there exists equivalent deterministic one with at most $g(n)$ states where

$$g(n) = \max\left\{\mathrm{lcm}(m_1, m_2, \ldots, m_k) : m_1 + m_2 + \ldots \leq n\right\}.$$

The construction works the same for the unambiguous automata. So for n-state unambiguous automaton, there exists equivalent deterministic on with at most $\tilde{g}(n)$ states where

$$\tilde{g}(n) = \max\{\mathrm{lcm}(m_1, m_2, \ldots, m_k) : m_1 + m_2 + \ldots \leq n, \text{ and}$$
$$\exists_{A_1,\ldots,A_k,m}[A, m; A_1 \bmod m_1, \ldots, A_k \bmod m_k] \text{ is unambiguous}\}.$$

Both these functions have non-trivial number theoretic properties. Lyubich in [Lyu] shows that

$$g(n) \sim e^{(1+o(1))\sqrt{n \ln n}},$$

while Okhotin in [Okh] proves that

$$\tilde{g}(n) \sim e^{\theta\left(\sqrt[3]{n \ln^2 n}\right)}.$$

What matters most for us is the conclusion that both of those are super-polynomial with respect to n. It has also been proven in the referenced papers that the bounds are optimal. That is, for sufficiently big n there exist n-state nondeterministic or unambiguous automata for which the smallest equivalent deterministic has correspondingly $O(g(n))$ or $O(\tilde{g}(n))$ states.

This remark is important for our considerations regarding complementation of unambiguous automata. While determinization and complementation of the deterministic automaton is clearly a viable way of complementing unambiguous automata. Okhotin's result shows that this may cause exponential blow-up. Therefore, to obtain lower complexity, we need more subtle methods.

Chapter 4

Complementation over unary alphabet

In this chapter we present some lower and upper bounds for complexity of complementation of an unambiguous unary automaton. We present an algorithm to build a complementation of size $O(n^{c \log n})$ for an automaton of size n and an example proving lower bound of $O(n^2)$.

4.1. Distinguishment graph

Thanks to Lemma 3.4 we can consider only automata in normal form. Let

$$\mathcal{A} = [A, m; A_1 \bmod m_1, A_2 \bmod m_2, \ldots, A_k \bmod m_k]$$

be such an automaton. For automaton $\mathcal{A}$ we define *distinguishment graph* to be any complete labelled graph $G = (V, E, l)$ such that

- $V = v_1, \ldots, v_k$ - nodes corresponding to the cycles of the automaton.
- $l(v_i, v_j) = r_{i,j}$ where $r_{i,j}$ is such that

$$A_i \bmod r_{i,j} \cap A_j \bmod r_{i,j} = \varnothing$$

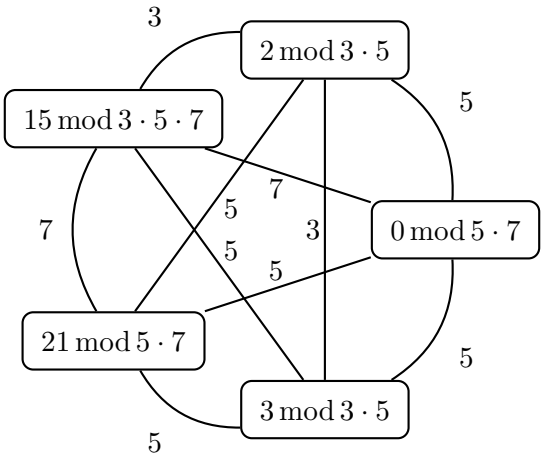
Thanks to Lemma 3.8 such a graph always exists, it suffices to take $r_{i,j} = \gcd(m_i, m_j)$. With an additional assumption, we can use Lemma 3.9 and take $r_{i,j} = p_{ij}$ to obtain a graph where all labels are prime.

Remark 4.1
If $\mathcal{A} = [A, m; A_1 \bmod m_1, A_2 \bmod m_2, \ldots, A_k \bmod m_k]$ *and additionally* $|A_1| = \ldots = |A_k| = 1$*, there exists a distinguishment graph for* $\mathcal{A}$ *such that all* $r_{i,j}$ *are prime.*

Example 4.2
Consider an example automaton

$$[0;\ 15 \bmod 3 \cdot 5 \cdot 7;\ 2 \bmod 3 \cdot 5;\ 0 \bmod 5 \cdot 7;\ 3 \bmod 3 \cdot 5;\ 21 \bmod 5 \cdot 7].$$

The distinguishment graph could look as follows.



It is easy to verify that the required properties hold, for example for edge between nodes corresponding to the first and the third cycle

$$15 \bmod 7 = 1 \neq 0 \bmod 7.$$

The idea behind the graph is that it is easy to check whether a word is accepted by the automaton or not if we knew upfront that it may be only accepted by one of the cycles m_i. Indeed, an automaton counting modulo m_i can do that. Numbers $r_{i,j}$ serve to distinguish cycles from one another - as $A_i \bmod r_{i,j} \cap A_j \bmod r_{i,j} = \varnothing$, then after counting modulo $r_{i,j}$ we know that the word might be accepted only by one (or none) of cycles i, j. This leads to our technique of *tree of questions* of using subsequent $r_{i,j}$ to rule out cycles which could accept given word while controlling the size of produced automaton.

4.2. Tree of Questions

The tree of questions is another abstraction on the way to build an automaton for complementation. It depicts a process of checking reminders modulo subsequent primes to decide whether a word is accepted or not. Formally, it is any rooted tree T with labelled nodes and edges is a *tree of questions* if

1. Labels on inner nodes are primes („questions"), and outgoing edges from any given node are labelled with all residues („answers") modulo the node's label

2. Leaf nodes have no labels

3. For each path from root node to a leaf node, labels of all nodes on the path are pairwise different.

Let N be a node in such a tree, and let $(p_i, r_i)_{i=0}^{k}$ be the labels on all nodes and edges on the path from root to N, not including label of N. The language

$$L_N = \{a^n \mid n \equiv r_0 \mod p_0,\ \ldots,\ n \equiv r_k \mod p_k\}$$

will be called the language of the node. Note the following remarks.

Remark 4.3
For different leaf nodes N, M languages L_N, L_M are disjoint.

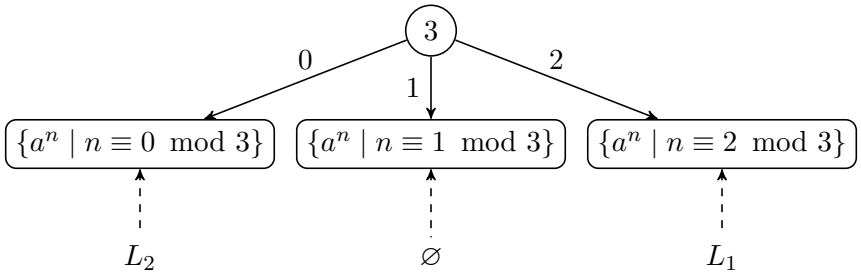
Proof. Consider lowest common ancestor P of N, M, and let its label be p. Then paths from root to N and M must take different outgoing edges from P, let their labels be r_n, r_m, respectively. Therefore lengths of all words in L_N are equivalent to r_n modulo p, while all in L_M to r_m, so the two languages are disjoint. □

Remark 4.4
Union of L_N over all leaf nodes N is Σ^*.

Proof. Note that for any inner node N with label p and children $M_0, \ldots, M_{p-1}$, its language L_N is equal to union of its children's languages $L_{M_0} \cup \ldots \cup L_{M_{p-1}}$. Indeed, L_{M_i} is equal to L_N with an additional constraint that $n \equiv i \mod p$. As labels on outgoing edges are all possible residues modulo p, this shows that the union is equal to L_N.
It is easy to show by induction over the tree structure that union of L_N for all leaf nodes N is equal to language of the root, which is Σ^*. □

Given an unambiguous automaton $\mathcal{A} = [A, m; A_1 \bmod m_1, A_2 \bmod m_2, \ldots, A_k \bmod m_k]$, let $L_i = \{a^n \mid n \bmod m_i \in A_i\}$. We say that a tree of questions T as above is *complete* for the automaton $\mathcal{A}$, if each L_N intersects non-trivially with at most one L_i. Notice that as L_N form a partition of Σ^*, each L_i must intersect non-trivially with some L_N. For given leaf node N in a complete tree, let L_{i_N} be the only one of L_i intersecting non-trivially with L_N if such exists, or $\varnothing$ otherwise.

In practice, completeness condition means that if we knew that a word w is in L_N, we would know it cannot be in any L_i but for possibly one. For example, consider automaton $[\varnothing, 0; 3 \bmod 6, 2 \bmod 6]$. The simple tree depicted below is complete for this automaton. The leaf nodes in the figure contain the description of node's language and corresponding L_{i_N} are below.



The main challenge is how to structure the tree for a given automaton so that the completeness condition is met, while keeping the tree small enough to avoid exponential blow-up in the resulting automaton.

4.2.1. Automaton construction

Constructing the automaton from the tree of questions is relatively straightforward. Let T be a complete tree of questions for an automaton $\mathcal{A}$. We should first describe how to build an automaton recognizing L_N for a leaf node N. Recall that $L_N = \{a^n \mid n \equiv r_0 \mod p_0, \ \ldots, \ n \equiv r_k \mod p_k\}$ if $(p_i, r_i)_{i=0}^k$ are the labels on the path to N. By Chinese Remainder Theorem, there exists r such that the set of conditions for n is equivalent to single condition $n \equiv r \mod p_0 p_1 \cdots p_k$. Therefore

$$L_N = \{a^n \mid n \equiv r \mod p_0 p_1 \cdots p_k\}.$$

This language is recognized by a deterministic automaton $\mathcal{B}_N$ consisting of $P = p_0 p_1 \cdots p_k$ states forming a cycle $q_0 \to q_1 \to \ldots \to q_{P-1}$, with a single state q_r accepting.

Next, we can extend this construction to build automaton $\tilde{\mathcal{B}}_N$, recognizing $L_N \setminus L_{i_N}$. If L_{i_N} is an empty set, there is nothing to be done. Otherwise, it is one of L_i, which is recognized by a deterministic automaton - one of the cycles in $\mathcal{A}$. We can easily complement L_i by complementing the deterministic automaton recognizing it and then build $\tilde{\mathcal{B}}_N$ as $L_N \cap \bar{L}_i$ using Lemma 2.2.

Finally, it remains to join the constructed automata $\tilde{\mathcal{B}}_N$ for all leaf nodes, to create an automaton for union of languages. This can be done one by one using Lemma 2.1, as L_N are pairwise disjoint. The resulting automaton $\mathcal{B}$ recognizes language

$$\bigcup_{N \text{ leaf}} L_N \setminus L_{i_N},$$

which, as L_N are pairwise disjoint, is equal to

$$\left(\bigcup_{N \text{ leaf}} L_N\right) \setminus \left(\bigcup_{N \text{ leaf}} L_N \cap L_{i_N}\right).$$

Notice however that the first term equals Σ^*, thanks to Remark 4.4. The second one, is a union of some empty sets together with all L_i of $\mathcal{A}$, intersected with sets forming a partition. Therefore, it is exactly $L(\mathcal{A})$. This shows that $\mathcal{B}$ recognizes exactly

$$\Sigma^* \setminus L(\mathcal{A}).$$

4.2.2. Our questions strategy

Now we proceed to describe precisely our idea for the structure of the tree which will allow for quasi-logarithmic complementation in a special case. Generalization uses this result and is mostly technical. Let

$$\mathcal{A} = [A, m; A_1 \bmod m_1, A_2 \bmod m_2, \ldots, A_k \bmod m_k]$$

be an unambiguous automaton in the normal form. Assume for now that $m = 0$ and $|A_1| = \ldots = |A_k| = 1$. Let G be a distinguishment graph for automaton $\mathcal{A}$, as defined in Section 4.1. Thanks to the Remark 4.1 we can assume that all $r_{i,j}$ are prime.

First, define tree $T_{v_i,P}$ for node v_i, omitting set of primes P, which would be then used as a subtree of tree of questions for $\mathcal{A}$. Let $(r_{i,j_1}, r_{i,j_2}, \ldots, r_{i,j_k})$ be a sequence of all $r_{i,j}$ (labels on all edges incident with v_i) which are not in P, without repetitions. Tree $T_{v_i,P}$ will have k levels of inner nodes and a level of leaf nodes. The first level is a single node with label r_{i,j_1}. The second has to contain r_{i,j_1} nodes - one for each possible residue. Let all of them have label r_{i,j_2}. Therefore, the third level has to have $r_{i,j_1} \cdot r_{i,j_2}$ nodes. We keep adding nodes on subsequent levels, in the level l all $r_{i,j_1} \cdot \ldots \cdot r_{i,j_{l-1}}$ with the same label r_{i,j_l}. Finally, we fill the last level with leaf nodes with no label. Therefore, $T_{v_i,P}$ is a symmetrical tree of depth k with branching r_{i,j_l} on l-th level.

The process of building a tree will progress as follows. First, replace empty root node with tree $T_{v_i,\varnothing}$ for arbitrary i. Then repeat the following. Consider all leaf nodes N and their languages L_N as well as all languages L_i accepted by cycles of $\mathcal{A}$. Let P_N be the set of labels of all nodes of the path from root to a node N, excluding its label if present. Let the *interest set* $I(N)$ for a node N be the set of these languages L_i which intersect non-trivially with L_N. Note that in particular if $L_i \in I(N)$ then by properties of distinguishement graph either $I(N) = \{L_i\}$ or P_N does not contain all primes $r_{i,j}$. Leaf node N will be considered *active* as long as its interest set contains more than one language. Each phase will comprise of replacing each active leaf node with T_{v_i,P_N} for some v_i such that $L_i \in I(N)$. Thanks to the note above, T_{v_i,P_N} has at least one level. This is repeated until no active leaf node is left. Note that, by definition of active node, this guarantees that the tree will be complete for $\mathcal{A}$, as then each L_N will intersect non-trivially only with at most one language L_i.

It remains to prove that the construction will halt. For each v_i, let $c_{i,N}$ denote the number of different primes amongst $r_{i,j}$ which are not in P_N. Let $c_i = c_{i,R}$ where R is the root of the tree and $c = \max_i c_i$. Consider any leaf node N and a phase in which it

is replaced by a tree T_{v_k,P_N} for some k, and let M be an arbitrary active leaf node of this tree. As $L_M \subseteq L_N$ then $I(M) \subseteq I(N)$. For each i such that $L_i \in I(N)$, $c_{i,M} < c_{i,N}$. Indeed, for each L_i in $I(N)$ with $i \neq k$, $r_{i,k}$ is not in P_N, by definition of distinguishment graph - if it was already checked, then one of L_k, L_i would not be in the interest set. As in T_{v_k,P_N} we add all new $r_{i,k}$, then for each i, $c_{i,M}$ will decrease because of not counting $r_{i,k}$ anymore.

After phase i let d_i be the maximum of $c_{i,N}$ for all leaf nodes N and i such that $L_i \in I(N)$. By the observation above, d_i decreases every phase, as each L_i either disappears from interest sets of active nodes or $c_{i,N}$ is replaced by strictly smaller $c_{i,M}$. Therefore, as $d_1 \leq c$, the construction will end after at most c phases.

4.2.3. Complexity estimation

Size of the automaton is the sum of sizes of all deterministic automata constructed for the leaf nodes. Each of these is of size equal to product of all labels of the nodes on the path to it from root, possibly times size of automaton for L_i - one of the cycles in $\mathcal{A}$. The path consists of at most c segments created in subsequent phases. Each segment corresponds to some v_i, and as such, contains questions only about primes dividing m_i and the same prime is never checked more than once. Therefore product of primes in a given segment is not bigger than m_i, so not bigger than $|L|$. Therefore product on each path is not bigger than $|L|^c$, and automaton for leaf node not bigger than $|L|^c \cdot |L_i| \leq |L|^{c+1}$.

The remaining thing is to compute how many of leaf nodes are there. Notice that in each phase we modify most bottom nodes by adding a tree with many more leaf nodes beneath. More precisely, when extending a node using component v_i, we add nodes labelled with different prime divisors of m_i, and a node labelled with some p adds p nodes below - one for each residue. Therefore, the number of new leaf nodes after a phase is not bigger than a product of different prime divisors of m_i, so not bigger then m_i and also $|L|$. So in each phase the number of leaf nodes grows at most $|L|$ times, so in total there are at most $|L|^c$ leaf nodes.

Therefore the automaton for complementation is not bigger than $|L|^c \cdot |L|^{c+1} = |L|^{2c+1}$. Now it only remains to notice that as $c = \max_i c_i$, there exists j such that $c = c_j$. But then

$$m_j \geq p_{j,1} p_{j,2} \cdot \ldots \cdot p_{j,c_j} \geq 2^{c_j},$$

$$|L| \geq m_j \geq 2^c,$$

$$c \leq \log |L|.$$

Therefore, the automaton for complementation of L is of size $|L|^{O(\log |L|)}$.

4.2.4. General case

We made two technical assumptions about the structure of automaton to produce $|L|^{O(\log|L|)}$ complementation. First was that $m = 0$ (there is no chain), and second that $|A_1| = \ldots = |A_k| = 1$. To lift the first one we need the following lemma.

Lemma 4.5
Let

$$\mathcal{A} = [\varnothing, 0; A_1 \bmod m_1, A_2 \bmod m_2, \ldots, A_k \bmod m_k],$$

$$\mathcal{B} = [\varnothing, 0; B_1 \bmod m'_1, B_2 \bmod m'_2, \ldots, B_l \bmod m'_l],$$

be automata in normal form, such that $\mathcal{B}$ recognizes complement of language of $\mathcal{A}$. Now if

$$\tilde{\mathcal{A}} = [C, c; A_1 \bmod m_1, A_2 \bmod m_2, \ldots, A_k \bmod m_k],$$

$$\tilde{\mathcal{B}} = [\{0, 1, \ldots, c\} \setminus C, c; B_1 \bmod m'_1, B_2 \bmod m'_2, \ldots, B_l \bmod m'_l],$$

then $\tilde{\mathcal{B}}$ recognizes complement of language of $\tilde{\mathcal{A}}$.

Proof. By symmetry, it suffices to show that $L(\tilde{\mathcal{A}}) \subseteq \Sigma^* \setminus L(\tilde{\mathcal{B}})$. Let $w \in L(\tilde{\mathcal{A}})$. If $|w| \leq c$ then $w \in C$, so $w \notin \{1, \ldots, c\} \setminus C$, so $w \notin L(\tilde{\mathcal{B}})$. Otherwise, if $|w| > c$, then $w = c + v$, and as $w \in L(\tilde{\mathcal{A}})$ then $v \in L(\mathcal{A})$. So $v \notin L(\mathcal{B})$, so $w \notin L(\tilde{\mathcal{B}})$. □

For the second one, notice that one may simply transform any automaton to such with only a quadratic size blow-up. Indeed, we replicate $|A_i|$ times each cycle m_i, with each copy accepting only one residue from A_i. The resulting automaton is of size $\sum A_i^2 + m \leq (\sum A_i)^2 + m \leq n^2$.

Therefore, for general unambiguous automaton $\mathcal{A}$, such that $|\mathcal{A}| = n$, we apply the following procedure

1. transform $\mathcal{A}$ to equivalent $\mathcal{A}_1$ in normal form ($|\mathcal{A}_1| = n$)
2. remove chain from $\mathcal{A}_1$ to obtain $\mathcal{A}_2$ ($|\mathcal{A}_2| \leq n$)
3. transform $\mathcal{A}_2$ as described above to obtain equivalent $\mathcal{A}_3$ with only one residue accepted on each cycle ($|\mathcal{A}_3| \leq n^2$)
4. complement $\mathcal{A}_3$ to $\mathcal{B}$ using the tree of questions technique described earlier in this chapter ($|\mathcal{B}| \leq O((n^2)^{\log n^2}) = O(n^{4 \log n})$)
5. re-attach chain - apply Lemma 4.5 to $\mathcal{A}_2$, $\mathcal{B}$ and $\mathcal{A}_1$ to obtain $\tilde{\mathcal{B}}$ ($|\tilde{\mathcal{B}}| \leq O(n^{4 \log n}) + n$)

Now $\tilde{\mathcal{B}}$ recognizes complement of language accepted by $\mathcal{A}$, and $|\tilde{\mathcal{B}}| \leq O(n^{4 \log n}) + n = n^{O(\log n)}$.

4.3. Examples

4.3.1. Prime numbers

Examples and constructions in this chapter freely use prime numbers with some hidden assumptions on their distribution. Here we present, without proof, a result from numbers theory which justifies such usage. The proof may be found in the original paper [Ram]. Let $\pi(n)$ denote the number of primes between 1 and n.

Lemma 4.6
For any given k, there exists R_k such that for each $n \geq R_k$, $\pi(n) - \pi(n/2) \geq k$.

Smallest R_k such that it holds is a prime number and is called Ramanujan prime. This result means that for any given k we can find arbitrarily large n such that there are $p_1, \ldots, p_k$ that all are between $n/2$ and n, so all asymptotically equal to n.

Number theory gives also the following interesting bounds on R_k.

Lemma 4.7
For $k \geq 1$, $2k \ln 2k < R_k < 4k \ln 4k$.

This means that if we need k asymptotically equal primes we may start from one as small as $O(k \ln k)$.

4.3.2. Quadratic lower bound

Example 4.8
Let $p_1, \ldots, p_{2k+1}$ be pairwise different primes. Consider the language $L = \bigcup_{i=1}^{2k+1} L_i$ where

$$L_i = \{a^n \mid n \not\equiv 0 \mod p_i, n \equiv 0 \mod p_{i+1} \cdots p_{i+k}\},$$

where $p_c := p_{c-2k-1}$ for $c > 2k+1$. Then L can be recognized by an unambiguous automaton of size $1 + \sum_{i=1}^{2k+1} p_i \cdots p_{i+k}$, while unambiguous automaton recognizing $\bar{L}$ needs to have at least $p_1 \cdots p_{2k+1}$ states.

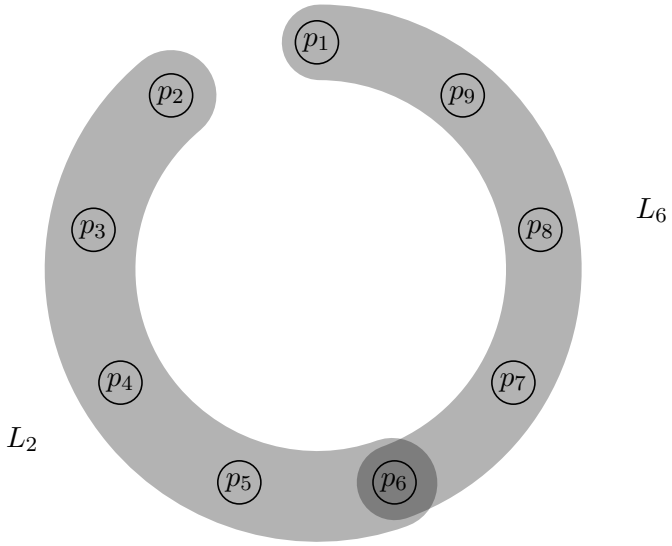


Figure 4.1: Illustration of which primes are taken into account for different L_i, with $k = 4$

Proof. First we should show that languages L_i are pairwise disjoint. Consider some L_i, L_j with $i \neq j$ - then either $p_i \in \{p_j, \ldots, p_{j+k}\}$ or $p_j \in \{p_i, \ldots, p_{i+k}\}$ - this is the case because both sets contain more than half of all primes. Assume that the first alternative holds. Then L_j contains only words not divisible by p_i while L_i contains only words divisible by p_i, so they are indeed disjoint. Therefore an unambiguous automaton for L may be simply constructed as union of automata for L_i, as in Lemma 2.1. Automata for L_i are just cycles of length $p_i \cdots p_{i+k}$, with appropriate states accepting.

Consider any unambiguous automaton recognizing $\bar{L}$ and the equivalent automaton, of the same size, in the normal form (3.3). It accepts all words of length divisible by $p_1 \cdots p_{2k+1}$ - as each of L_i requires non-zero residue modulo some prime. As there are infinitely many of these words, one of the cycles must accept infinitely many of them. Consider the subset of words $S \subset \bar{L}$ accepted by this cycle. It is obviously infinite and contains some word n of length divisible by $p_1 \cdots p_{2k+1}$. Let m be the length of this cycle and assume $p_i \nmid m$. Then

$$w := n + p \cdot p_1 \cdots p_{i-1} p_{i+1} \cdots p_{2k+1} \in S,$$

because $n \in S$ and p is the period. Then $p_i \nmid w$, as $p_i \mid n$ and $p_i \nmid p$, by assumption. On the other hand, for $j \neq i$, $p_j \mid w$, as $p_j \mid n$ and this prime appears in the long product. Therefore w satisfies conditions for belonging to $L_i \subset L$, which is a contradition with $w \in S \subset \bar{L}$. Therefore assumption $p_i \nmid p$ is incorrect for any i, so $p_1 \cdots p_{2k+1} \mid p$. So this cycle, and the whole automaton, is of size at least $p_1 \cdots p_{2k+1}$.

Now if p_i are primes of similar size, $p_i = O(p)$, as provided by Lemma 4.6, then the

size of automaton for L is $n = O(p^{k+1})$ while the size of automaton for $\bar{L}$ is at least $O(p^{2k+1}) = O(n^2)$. □

4.3.3. Putative superpolynomial lower bound

A new paper [Ras] has been published for review which claims to present an example of a language recognized by unambiguous automaton of size n, for arbitrarly large n, which requires an nondeterministic automaton of size at least $n^{O(\log \log n)}$ for recognizing its complement. Very interesting is the fact that the example is unary, which suggests that most of the complexity of complementation problem is already there in this seemingly much easier case. Together with our work, this would mean obtaining quite tight, while still not equal, bounds for this problem for unary automata.

Chapter 5

Conclusions

5.1. Future work

The main area of future work appears to be the general setting with arbitrary alphabets. There is no clear to us way to generalize methods developed in this paper, so this may require entirely different methods. The problems with generalization have two main reasons - first, there is no known useful „normal form" for automata over bigger alphabets, second, the words have much richer structure. In particular, if two unary languages are disjoint, they must also be disjoint modulo some number, or even some prime under additional assumptions as described in Chapter 3. This allows reducing the problems to statements from number theory. For arbitrary alphabets we do not know of any object which could take this role.
If the super-polynomial lower bound claim holds, there seems not much left to be done in the unary case anymore. On the other hand, there still remains a gap between bounds which could be closed.

5.2. Acknowledgements

I am indebted to Wojciech Czerwiński for acquainting me with the subject of unambiguous automata as well as for supervision and invaluable assistance while working on this thesis. Thanks are due to Tomasz Gogacz and Michał Skrzypczak for the ideas that significantly contributed to the main result of this work. I am grateful to Michał Pilipczuk for creative examples which disproved many of our intermediate hypotheses.

Chapter 6

Bibliography

[Arn] A. Arnold, *Rational omega-languages are non-ambiguous*, Theoretical Computer Science 26, 1–2 (1983)

[BTV] C. Bourke, R. Tewari, N. V. Vinodchandran, *Directed Planar Reachability Is in Unambiguous Log-Space*, ACM Transactions on Computation Theory, Volume 1 Issue 1, article No. 4 (2009)

[BiSk] M. Bilkowski, M. Skrzypczak, *Unambiguity and uniformization problems on infinite trees*, LIPIcs-Leibniz International Proceedings in Informatics 23, 81-100 (2013)

[CLNW] A. Carayol, C. Löding, D. Niwiński, I. Walukiewicz, *Choice functions and well-orderings over the infinite binary tree*, Central European Journal of Mathematics 8(4), 662-682 (2010)

[Chr] M. Chrobak, *Finite automata and unary languages*, Theoretical Computer Science 47, 149-158 (1986)

[Col] T. Colcombet, *Unambiguity in Automata Theory*, DCFS 2015: Descriptional Complexity of Formal Systems, 3-18 (2015)

[HRS] H. B. Hunt, D. J. Rosenkrantz, T. G. Szymanski, *On the Equivalence, Containment, and Covering Problems for the Regular and Context-Free Language*, Journal of computer and system sciences 12, 222-268 (1976)

[HSKKS] J. Hromkovič, S. Seibert, J. Karhumäki, H. Klauck, G. Schnitger, *Communication Complexity Method for Measuring Nondeterminism in Finite Automata*, Information and Computation volume 172 issue 2, 202-217 (2002)

[JDR] T. Jiang, E. McDowell, B. Ravikumar, *The structure and complexity of minimal NFAs over a unary alphabet*, International Journal of Foundations of Computer Science Vol. 2, No. 2, 163-182 (1991)

[Leu] H. Leung, *Descriptional complexity of NFA of different ambiguity*, International Journal of Foundations of Computer Science 16, 975-984 (2005)

[LoMa] S. Lombardy, J. Mairesse, *Series which are both max-plus and min-plus rational are unambiguous*, RAIRO - Theoretical Informatics and Applications 40.1, 1-14 (2006)

[Lyu] Yu. Lyubich, *Bounds for the optimal determinization of nondeterministic autonomic automata*, Sibirskii Matematicheskii Zhurnal 2, 337-355 (1964)

[Okh] A. Okhotin, *Unambiguous finite automata over a unary alphabet*, Information and Computation 212, 15-36 (2012)

[RA] K. Reinhardt, E. Allender, *Making Nondeterminism Unambiguous*, SIAM Journal of Computing 29(4), 1118-1131 (1997)

[Ram] S. Ramanujan, *A proof of Bertnard's postulate*, Journal of the Indian Mathematical Society XI, 181-182 (1919)

[Ras] M. Raskin, *A superpolynomial lower bound for the size of non-deterministic complement of an unambiguous automaton*, arXiv:1711.03993 [cs.CC] (2017)

[StHu] R. E. Stearns, H. B. Hunt, *On the equivalence and containment problems for unambiguous regular expressions, grammars, and automata*, SFCS '81 Proceedings of the 22nd Annual Symposium on Foundations of Computer Science, 74-81 (1981)

[Web] A. Weber, *Decomposing a k-valued transducer into k unambiguous ones*, Latin American Symposium on Theoretical Informatics '92, 503-515 (1992)

[Yao] A. Yao, *Some complexity questions related to distributive computing*, STOC '79 Proceedings of the eleventh annual ACM symposium on Theory of computing, 209-213 (1979)