Minimal Automata and Operational State Complexity of Block Languages

Guilherme Manuel Carvalho de Melo Duarte
Mestrado em Ciência de Computadores
Departamento de Ciências de Computadores
Faculdade de Ciências da Universidade do Porto
2024

Minimal Automata and Operational State Complexity of Block Languages

Guilherme Manuel Carvalho de Melo Duarte

Dissertação realizada no âmbito do Mestrado em Ciências de Computadores

Departamento de Ciências de Computadores

2024

Orientador
Rogério Ventura Lages dos Santos Reis, Professor Auxiliar,
Faculdade de Ciências da Universidade do Porto

Coorientador
Nelma Resende Araújo Moreira, Professora Auxiliar,
Faculdade de Ciências da Universidade do Porto

This page is intentionally left blank.

Acknowledgements

First, I would like to express my deepest gratitude to my advisors, Rogério Reis and Nelma Moreira, for their ability to teach, their guidance, and their total availability throughout this thesis.

To Luca Prigioniero, for his valuable suggestions and advice, not only as a co-author.

To my group of friends, for their capacity to make me laugh even during my grumpy moods.

To my family, especially, to my mother and to my brother, for their constant encouragement and immense support not only during this thesis.

To Ana, whose impressive patience and kind heart have been a constant source of comfort and motivation.

To those who now shine bright in the sky.

This page is intentionally left blank.

Agradecimentos

Em primeiro lugar, gostaria de expressar o meu mais profundo agradecimento aos meus orientadores, Rogério Reis e Nelma Moreira, pela sua capacidade de ensinar, pela sua orientação e pela sua total disponibilidade ao longo desta tese.

Ao Luca Prigioniero, pelas suas valiosas sugestões e conselhos, não só como coautor.

Ao meu grupo de amigos, pela capacidade de me fazerem rir mesmo nos meus momentos mais rabugentos.

À minha família, em especial, à minha mãe e ao meu irmão, pelo constante incentivo e imenso apoio não só durante esta tese.

À Ana, cuja impressionante paciência e bondoso coração têm sido uma fonte constante de conforto e motivação.

Àqueles que agora brilham no céu.

This page is intentionally left blank.

Abstract

Formal languages and automata theory are the backbone of theoretical computer science. While formal languages categorise sets of words based on rules, automata theory explores various machines, like finite automata, that can recognise these languages. In fact, these branches are crucial in diverse fields such as pattern recognition, compilers, programming languages, software verification, among others.

In this work we consider block languages, namely sets of words having the same length, motivated primarily by their applications on image processing and code theory. We propose a new representation for these languages, which we call bitmap. In particular, given an alphabet of size k and a length ℓ, a block language can be represented by a bitmap of length k^ℓ, where each bit indicates whether the corresponding word, according to the lexicographical order, belongs, or not, to the language (bit equal to 1 or 0, respectively). This representation turns out to be a good tool for the investigation of several properties of block languages, making proofs simpler and reasoning clearer. First, we show how to convert bitmaps into deterministic and nondeterministic finite automata. We then focus on the size of the machines obtained from the conversion and we prove that their size is minimal.

Furthermore, we give an analysis of the maximum number of states sufficient to accept every block language in the deterministic and nondeterministic case. Moreover, we study the deterministic and nondeterministic state complexity of several standard operations on these languages. Being a subclass of finite languages, the upper bounds of operational state complexity known for finite languages apply for block languages as well. However, in several cases, smaller values were found.

Keywords: regular languages, finite automata, finite languages, operational state complexity.

This page is intentionally left blank.

Resumo

As linguagens formais e a teoria dos autómatos são a coluna da teoria da computação. Enquanto as linguagens formais classificam conjuntos de palavras com base em regras, a teoria dos autómatos explora várias máquinas, como os autómatos finitos, que podem reconhecer essas linguagens. De facto, estes ramos são cruciais em diversos domínios, como o reconhecimento de padrões, compiladores, linguagens de programação, verificação de software, entre outros.

Neste trabalho consideramos as linguagens de bloco, nomeadamente conjuntos de palavras com o mesmo comprimento, motivados principalmente pelas suas aplicações no processamento de imagem e na teoria dos códigos. Propomos uma nova representação para estas linguagens, a que chamamos bitmap. Em particular, dado um alfabeto de tamanho k e um comprimento ℓ, uma linguagem de bloco pode ser representada por um bitmap de comprimento k^ℓ, em que cada bit indica se a palavra correspondente, de acordo com a ordem lexicográfica, pertence ou não à linguagem (bit igual a 1 ou 0, respetivamente). Esta representação revela ser uma boa ferramenta para a investigação de várias propriedades das linguagens de bloco, tornando as provas mais simples e os argumentos mais claros. Primeiro, mostramos como converter bitmaps em autómatos finitos determinísticos e não determinísticos. De seguida, concentramo-nos no tamanho das máquinas obtidas a partir da conversão e provamos que o seu tamanho é mínimo.

Para além disso, apresentamos uma análise do número máximo de estados suficiente para aceitar todas as linguagens de blocos no caso determinístico e não determinístico. Estudamos ainda a complexidade de estados determinística e não determinística de várias operações padrão nestas linguagens. Sendo uma subclasse de linguagens finitas, os limites superiores de complexidade de estado conhecidos para as linguagens finitas aplicam-se também às linguagens de blocos. No entanto, em vários casos, foram encontrados valores inferiores.

Palavras-chave: linguagens regulares, autómatos finitos, linguagens finitas, complexidade de estado operacional.

This page is intentionally left blank.

Contents

This page is intentionally left blank.

List of Tables

This page is intentionally left blank.

List of Figures

List of Algorithms

This page is intentionally left blank.

Chapter 1

Introduction

In the area of formal languages and automata theory, the class of regular languages is one of the most investigated. Classical recognisers for this class are finite automata, in both deterministic and nondeterministic variants. The ability of these machines to represent languages in a more or less succinct way have been widely studied in the area of *descriptional complexity*. In this context, the size of a model is measured in terms of the number of symbols used to write down its description. In the specific case of finite automata, it is often considered the number of states as a measure of complexity. In this area, the minimality of finite automata has been also studied. For example, it is well known that, given a language, the deterministic finite automaton of minimal size accepting it is unique (up to isomorphisms), and there exist efficient algorithms for the minimisation of these machines [AMR12]. It is a more challenging task for the nondeterministic case, since minimal nondeterministic finite automata are not necessarily unique. Furthermore, given an integer n, deciding whether there is a nondeterministic finite automaton with less than n states accepting a language is a PSPACE-HARD problem [SM73].

In this work we consider finite languages where all words have the same length, which are called *homogeneous* or *block* languages. Their investigation is mainly motivated by their applications to several contexts such as code theory [DK12, KMR18] and image processing [KK21, KO14]. A typical problem in code theory is the construction of maximal block languages, that correspond to codes, capable of detecting and correcting errors. On the other hand, an image can be represented by a block language, so automata can be used to generate, compress, and manipulate images.

As a subclass of finite languages, block languages inherit some properties known for that class. For instance, the minimisation of deterministic finite automata can be done in linear time in the case of finite (and hence also block) languages [Rev92]. Due to the fact that all words have the same length, there are some gains in terms of descriptional complexity. It is known that the

elimination of nondeterminism from an n-state nondeterministic finite automaton for a block language costs $2^{\Theta(\sqrt{n})}$ in size [KO14], which is smaller than the general case, for which the cost in size is $2^{\Theta(n)}$ [MF71, SY97]. The maximum number of states of minimal deterministic finite automata for finite and block languages were studied by Câmpeanu and Ho [CH04], and Hanssen and Liu determined the number of block languages that attain that maximum state complexity [KL19].

Here we propose a new representation for block languages. In particular, given an alphabet of size k and a length ℓ, each block language can be represented by a binary string of length k^ℓ, also called *bitmap*, in which each *bit* indicates whether the correspondent word, according to the lexicographical order, belongs to the language (bit equal to 1) or not (bit equal to 0). Then, we show how to convert bitmaps into deterministic and nondeterministic finite automata, respectively, such that the devices yielded by these conversions have minimal size. While the conversion to deterministic finite automata can be done in polynomial time in the size of the bitmap, we prove that the transformation in the nondeterministic case is NP-COMPLETE.

Moreover, we also use bitmaps for studying the complexity of operations on block languages. Due to the distinguishing property of the words sharing the same length, we study standard Boolean binary operations over block languages with the same length, as well as the block complement operation (i.e., $\Sigma^\ell \setminus L$, for a block language L over an alphabet Σ and block length $\ell > 0$). Nonetheless, we also consider operations such as concatenation, Kleene star, and Kleene plus, which are not closed for the class of block languages of a given length, as well as specific operations on block languages such as word removal and addition.

1.1 Main Contributions

From the developed work on this thesis, we submitted two conference papers:

1. G. Duarte, N. Moreira, L. Prigioniero, and R. Reis, Block Languages and their Bitmap Representations, in *28th International Conference on Implementation and Application of Automata, Akita, Japan.* Accepted.

2. G. Duarte, N. Moreira, L. Prigioniero, and R. Reis, Operational Complexity on Block Languages, in *14th International Workshop on Non-Classical Models of Automata and Applications, Göttingen, Germany.* Accepted.

This first paper introduces a new representation of block languages as a binary word called bitmap, as well as several properties that allow us describe the construction from bitmaps to

minimal automata, both deterministic and nondeterministic. In this paper, we also give an analysis of the maximum number of states sufficient to accept every block language in the deterministic and nondeterministic case.

In the second paper, we study the deterministic and nondeterministic state complexity of several operations, and also compare the obtained upper bounds with the ones known for finite languages. We conclude that, in general, operations on block languages have lower complexities compared to the ones in the general case.

All the source code developed was integrated on the FAdo project [RM02], and it is freely available from http://fado.dcc.fc.up.pt/.

1.2 Thesis Structure

This dissertation is organised as follows:

Chapter 2 presents some basic notation and definitions of formal languages and automata theory. In particular, we define regular languages (REs), deterministic (DFAs) and nondeterministic finite automata (NFAs), as well as the state complexity of an automata.

Chapter 3 digs into finite languages, a subset of regular languages. This chapter explores the structural properties of the finite automata that recognise finite languages, as well as their operational state complexity, underscoring that it is generally lower compared to the general case of regular languages. Moreover, this chapter also introduces block languages, which consist of words of the same length, and examine their automata representations and applications.

Chapter 4 marks the beginning of our contributions by introducing a new representation for block languages called bitmaps. Moreover, it contains the description of the constructions of minimal DFAs and NFAs from this bitmap representations.

Chapter 5 contains the analysis of the theoretical boundaries of minimal DFAs for block languages, as well of minimal NFAs. Furthermore, this chapters contains the study of the operational state complexity of various operations on block languages and provides witnesses to reason the tightness of the introduced bounds.

Chapter 6 reviews the contributions given in the previous two chapters and establishes possible research lines of open problems that arose during our work.

This page is intentionally left blank.

Chapter 2

Preliminaries

In this chapter, we present some basic mathematical notions and definitions in *formal languages* and *automata theory*. For more details, we refer the reader to Ullman and Hopcroft's *Introduction to Automata Theory* [HU79] and to Pin's *Handbook of Automata Theory* [Pin21], in particular, the chapters 1, 2, and 12 of the latter.

2.1 Basic Definitions

Given two integers i, j with $i < j$, let $[i, j]$ denote the set of integers from i to j, including both i and j, namely $\{i, i+1, \ldots, j\}$. Moreover, if i is equal to 0, we omit this value, thus $[j] = \{0, 1, ..., j\}$.

Two integers a and b are *congruent modulo* n, namely $a \equiv b \pmod{n}$, if a and b have the same remainder when divided by n. That is,

$$a \equiv b \pmod{n} \Leftrightarrow (\exists k \in \mathbb{Z}) : a - b = kn.$$

Let $f(n), g(n) : \mathbb{Z} \to \mathbb{Z}$ be two functions. To evaluate the asymptotic behaviour of $f(n)$ considering $g(n)$, we use the Bachmann–Landau notation, namely

- $f(n) = O(g(n))$, if $(\exists c, n_0 \in \mathbb{N})(\forall n > n_0) : |f(n)| \leqslant c \cdot g(n)$;
- $f(n) = \Omega(g(n))$, if $(\exists c, n_0 \in \mathbb{N})(\forall n > n_0) : |f(n)| \geqslant c \cdot g(n)$;
- $f(n) = \Theta(g(n))$, if $(\exists c_1, c_2, n_0 \in \mathbb{N})(\forall n > n_0) : c_1 \cdot g(n) \leqslant |f(n)| \leqslant c_2 \cdot g(n)$;
- $f(n) = o(g(n))$, if $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$.

2.2 Symbols, Words, and Languages

Let Σ be a finite set called an *alphabet*, whose elements are called *letters* or *symbols*. A *word* w of Σ is a sequence of elements from Σ, and we denote w by a mere juxtaposition of those elements, that is,

$$w = \sigma_1\sigma_2\cdots\sigma_n,$$

where $n \in \mathbb{N}$ and $\sigma_i \in \Sigma$, for all $i \in [1, n]$. The *length of a word* w is given by its number of symbols and is denoted as $|w|$. It can be inductively defined in the following way:

$$|\varepsilon| = 0,$$
$$|w\sigma| = |w| + 1,$$

where $\sigma \in \Sigma$ and ε denotes the *empty word*. For the number of occurrences of the symbol $\sigma \in \Sigma$ on the word w, we use the notation $|w|_\sigma$. The *reversal* of a word $w = \sigma_1\sigma_2\cdots\sigma_n$ is denoted by w^R and is obtained by reversing the order of symbols of w, i.e., $w^R = \sigma_n\sigma_{n-1}\cdots\sigma_1$.

The set Σ^n is defined as the set of all words of length n over an alphabet Σ. The infinite set of words over the alphabet Σ results from the *star operation* (alternatively, *Kleene closure*) on the set Σ, denoted as $\Sigma^\star$, and is given by

$$\Sigma^\star = \bigcup_{n \geqslant 0} \Sigma^n.$$

For example, over the alphabet $\Sigma = \{a, b\}$ we have

$$\Sigma^\star = \{\varepsilon, a, b, aa, ab, ba, bb, aaa, \ldots\}.$$

Moreover, we define the set Σ^+ as the *plus closure* of an alphabet Σ given by removing the empty word from $\Sigma^\star$, that is, $\Sigma^+ = \Sigma^\star \setminus \{\varepsilon\}$.

Given two words $u = a_1a_2\cdots a_n$, $v = b_1b_2\cdots b_m$, the *concatenation* of u with v is denoted as uv and represents the word

$$uv = a_1a_2\cdots a_nb_1b_2\cdots b_m.$$

The empty word ε is the identity element of concatenation, i.e., $\varepsilon w = w\varepsilon = w$. Moreover, it is easy to see that this operation is associative, that is, $(uv)w = u(vw)$, for all $u, v, w \in \Sigma^\star$, and that is not commutative, i.e., $uv \neq vu$. We also define $w^0 = \varepsilon$ and $w^{n+1} = w^nw$, for $w \in \Sigma^\star$ and $n \in \mathbb{N}^+$. Given a word $w \in \Sigma^\star$, we say that

- $u \in \Sigma^\star$ is a *prefix* of w if $(\exists v \in \Sigma^\star) : w = uv$;
- $u \in \Sigma^\star$ is a *suffix* of w if $(\exists v \in \Sigma^\star) : w = vu$;

- $u \in \Sigma^\star$ is an *infix* of w if $(\exists v, x \in \Sigma^\star) : w = xuv$.

A *language* L of $\Sigma^\star$ is a set of words over the alphabet Σ. Several operations can be defined on languages. Let $L_1, L_2 \subseteq \Sigma^\star$ and $w \in \Sigma^\star$.

1. *Union*: $L_1 \cup L_2 = \{ w \mid w \in L_1 \text{ or } w \in L_2 \}$;
2. *Intersection*: $L_1 \cup L_2 = \{ w \mid w \in L_1 \text{ and } w \in L_2 \}$;
3. *Concatenation*: $L_1 L_2 = \{ uv \mid u \in L_1 \text{ and } v \in L_2 \}$;
4. *Complement*: $\overline{L}_1 = \Sigma^\star \setminus L_1 = \{ w \in \Sigma^\star \mid w \notin L_1 \}$;
5. *Reversal*: $L_1^R = \{ w^R \mid w \in L_1 \}$;
6. *Power*: $L_1^n = \underbrace{L_1 L_1 \cdots L_1}_{n \text{ times}}$;
7. *Star*: $L_1^\star = \bigcup_{n \geqslant 0} L_1^n$;
8. *Plus*: $L_1^+ = L_1^\star \setminus \{\varepsilon\}$;
9. *Left quotient w.r.t. a word* w: $w^{-1} L_1 = \{ w' \in \Sigma^\star \mid ww' \in L_1 \}$.

2.3 Regular Languages

Regular languages (RL) are, according to the Chomsky hierarchy [Cho56], the most simple class in terms of expressiveness. This set can be recursively defined over an alphabet Σ as follows:

- Both the languages $\emptyset$ and $\{\varepsilon\}$ are RL;
- The *singleton* language $\{\sigma\}$ is RL, for each $\sigma \in \Sigma$;
- If L_1 and L_2 are both RL, then $L_1 \cup L_2$, $L_1 L_2$, and $L_1^\star$ are RL.

Now, let L be a language over an alphabet Σ. Two words $x, y \in \Sigma^\star$ are in the relation $\sim_L$, denoted as $x \sim_L y$, if, for every word $z \in \Sigma^\star$, $xz \in L$ if, and only if, $yz \in L$. In other words, $x \sim_L y$ if concatenating the same word to both x and y should either make both resulting words part of the language or make both not part of the language. For a word $w \in \Sigma^\star$, we write $[w]_{\sim_L}$ for the equivalent class of w in the equivalence relation $\sim_L$. Moreover, consider the following theorem known as the Myhill-Nerode Theorem [Ner58]:

Theorem 2.1 (Myhill-Nerode Theorem [Ner58], Part 1). *Let* $L \subseteq \Sigma^\star$ *be a language. Then,*

$$L \text{ is regular } \Leftrightarrow \text{ the relation } \sim_L \text{ has a finite number of equivalence classes.}$$

An immediate corollary of Theorem 2.1 is that if, for a language L, the equivalence $\sim_L$ has infinitely many equivalence classes, it is *not regular*. Consider the following example of how the Myhill-Nerode Theorem can be applied to prove that a language is not regular.

Example 2.2. *Let* $L = \{a^n b^n \mid n \in \mathbb{N}\}$ *be a language defined over* $\{a, b\}^\star$*, and let us prove that it is not regular. By Theorem 2.1, it suffices to find an infinite set of pairwise indistinguishable words. Let* $S = \{a^n \mid n \in \mathbb{N}\}$*. For each pair* $a^i, a^j \in S$*, if* $i \neq j$ *then we have* $a^i \nsim_L a^j$*, since* $a^i b^i \in L$ *but* $a^j b^i \notin L$*.*

Moreover, a language L is regular if it can be defined by a *finite automaton* or, alternatively, by a *regular expression*.

2.4 Finite Automata

A *nondeterministic finite automaton* (NFA) is a 5-tuple $\mathcal{A} = \langle Q, \Sigma, \delta, I, F\rangle$, where Q is a finite set of states, Σ is the alphabet, $\delta : Q \times \Sigma \to 2^Q$ is the transition function, $I \subseteq Q$ is the set of initial states, and $F \subseteq Q$ is the set of final states.

The transition function can be extended to sets of states

$$\begin{aligned} \delta &: 2^Q \times \Sigma \to 2^Q \\ \delta(Q', \sigma) &= \bigcup_{q \in Q'} \delta(q, \sigma), \end{aligned}$$

and also to words

$$\begin{aligned} \delta &: Q \times \Sigma^\star \to 2^Q \\ \delta(q, \varepsilon) &= \{q\}, \\ \delta(q, \sigma w) &= \delta(\delta(q, \sigma), w). \end{aligned}$$

The size of an NFA is given by the cardinality of the set of states, namely $|Q|$.

An NFA might also contain transitions by the empty word ε, that is, $\delta(q, \varepsilon) = q'$ with $q \neq q'$, and we call those ε-transitions. If an NFA contains ε-transitions, we call it an ε-NFA. Although we will not be using them in this work, it is important to note, however, that ε-NFAs and NFAs without ε-transitions have the same expressive power [HU79]. This means that any language that can be recognised by an ε-NFA can also be recognised by an NFA, and vice versa.

A *path* in an automaton $\mathcal{A}$ is a finite sequence of consecutive transitions

$$c : q_0 \xrightarrow{\sigma_1} q_1 \xrightarrow{\sigma_2} q_2 \to \ldots \to q_{n-1} \xrightarrow{\sigma_n} q_n,$$

where $q_i \in \delta(q_{i-1}, \sigma_i)$, for all $i \in [1, n]$, the state q_0 is its *origin*, the state q_n its *end*, the word $\sigma_1\sigma_2\cdots\sigma_n$ its *label*, and n its *length*. The path c is *initial* if $q_0 \in I$ and *final* if $q_n \in F$. An NFA accepts a word $w \in \Sigma^\star$ if there is a path both initial and final with label w.

It is convenient to represent an automata by a labelled directed graph whose vertices are labeled by the states of the automata and the edges are labeled by the transitions, as follows in Figure 2.1. The initial states are identified by having an incoming edge with no origin and the final states by two concentric circles.

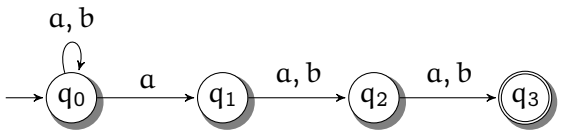

Figure 2.1: An NFA for $L = \{w \in \{a, b\}^\star \mid$ *the third but last symbol is* $a\}$.

The *right language* of a state $q \in Q$, denoted as $\mathcal{L}_q(\mathcal{A})$, is the set of labels of paths that start on q and are final, that is, $\mathcal{L}_q(\mathcal{A}) = \{w \in \Sigma^\star \mid \delta(q, w) \cap F \neq 0\}$. Analogously, one defines the *left language* of q, denoted as $\overleftarrow{\mathcal{L}}_q(\mathcal{A})$, as the set of labels of paths that are initial and end on q, that is, $\overleftarrow{\mathcal{L}}_q(\mathcal{A}) = \{w \in \Sigma^\star \mid q \in \delta(I, w)\}$. The *language accepted* by $\mathcal{A}$, denoted by $\mathcal{L}(\mathcal{A})$, is the set of labels of paths both initial and final, i.e., $\mathcal{L}(\mathcal{A}) = \bigcup_{q\in I} \mathcal{L}_q(\mathcal{A})$. Two NFAs are *equivalent* if they accept the same language.

An automaton can be qualified as *accessible*, *co-accessible*, and *trim*. An NFA is accessible if all its states are accessible, that is, there is an initial path ending in q ($\overleftarrow{\mathcal{L}}_q(\mathcal{A}) \neq \emptyset$), for all states $q \in Q$. It is co-accessible if all its states are co-accessible, i.e., for all states $q \in Q$, there is a final path starting in q ($\mathcal{L}_q(\mathcal{A}) \neq \emptyset$). Finally, an NFA is trim if it is both accessible and co-accessible.

An NFA $\mathcal{A} = \langle Q, \Sigma, \delta, I, F\rangle$ is *deterministic* (DFA) if it contains exactly one initial state ($|I| = 1$), and if, for every state $q \in Q$ and for each symbol $\sigma \in \Sigma$, there exists *at most one* state q' such that $\delta(q, \sigma) = q'$. A DFA is complete if the transition function is *total*, that is, for every state $q \in Q$ and for each symbol $\sigma \in \Sigma$, $\delta(q, \sigma)$ is defined. An incomplete DFA can be completed by adding a *sink-state*, denoted as the symbol Ω, which ensures that the automaton has defined transitions for every possible input symbol from every state. That is, for every $q \in Q$ and $\sigma \in \Sigma$, $\delta(\Omega, \sigma) = \Omega$ and $\delta(q, \sigma) = \Omega$, if $\delta(q, \sigma)$ was previously not defined.

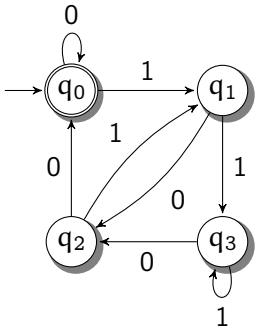

Figure 2.2: A DFA for $L = \{ w \in \{0,1\}^* \mid w \text{ is divisible by } 4 \}$.

2.4.1 Determinisation

To the conversion of an nondeterministic to an equivalent deterministic finite automata we call *determinisation*. Although NFAs have in general a more succinct representation than DFAs, the expressive power of both machines is the same. This equivalence was established through the *subset construction*, presented by Rabin and Scott [RS59].

Proposition 2.3 (Subset Construction [RS59]). *Let* $\mathcal{A} = \langle Q, \Sigma, \delta, I, F \rangle$ *be an NFA and* $\mathcal{D}(\mathcal{A}) = \langle 2^Q, \Sigma, \delta', I, F' \rangle$ *a DFA such that*

$$\delta' : 2^Q \times \Sigma \to 2^Q$$
$$\delta'(Q', \sigma) = \bigcup_{q \in Q'} \delta(q, \sigma),$$

and $F' = \{ Q' \mid Q' \cap F \neq \emptyset \}$. *Then,* $\mathcal{D}(\mathcal{A})$ *is equivalent to* $\mathcal{A}$.

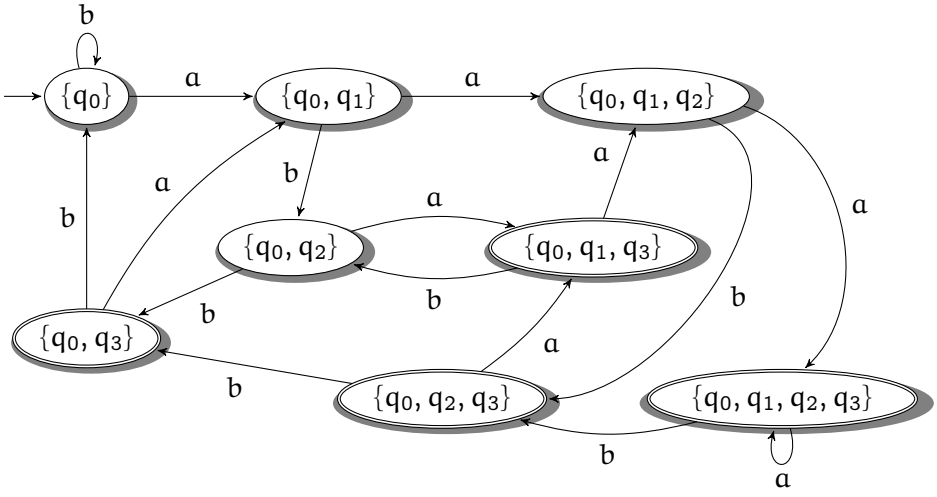

Figure 2.3: Determinisation of the NFA in Figure 2.1 using the subset construction.

Clearly, the number of states of the DFA resulting from the subset construction is at most $2^{|Q|}$ states, which corresponds to all the subsets of the set Q. Later, the upper bound was proven to be tight by Moore [Moo71] who provided an automaton with n states that requires at least 2^n states for its determinisation, depicted in Figure 2.4.

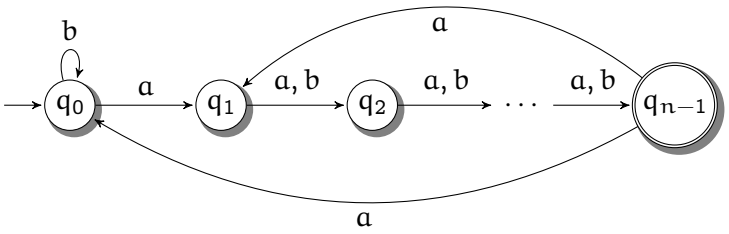

Figure 2.4: Moore's NFA with n states which any DFA requires at least 2^n states.

2.4.2 Minimisation

We say that an NFA is *minimal* if its number of states is the minimum among all NFAs that are equivalent. Furthermore, for each regular language there exists a minimal deterministic finite automaton that recognises it, and it is unique up to isomorphism [HU79]. A DFA $\mathcal{A}$ is *minimal* if each state is accessible and co-accessible, and every two states are *pairwise distinguishable*, that is, $\mathcal{L}_q(\mathcal{A}) \neq \mathcal{L}_{q'}(\mathcal{A})$, for every two states q, q' of $\mathcal{A}$.

Recall from Section 2.3 the discussion of the relation $\sim_L$ and of equivalence classes, as well as the first half of the Myhill-Nerode Theorem (Theorem 2.1). Consider this next result:

Theorem 2.4 (Myhill-Nerode Theorem [Ner58], Part 2)**.** *If a language $L \subseteq \Sigma^\star$, defined over an alphabet Σ, is regular, then the number of states in the minimal deterministic finite automaton accepting L is given by the number of equivalent classes in the relation $\sim_L$.*

In fact, by the equivalence relation, it follows that if $x \sim_L y$ then $x^{-1}L = y^{-1}L$, that is, the quotient of L by each word is identical. By Theorem 2.1, we know that a language is regular if there is a finite number of equivalent classes. Now, Theorem 2.4 tells us that each state of the minimal DFA refers to one of these quotients.

Moore's algorithm [Moo56] is based on the computation of the Myhill-Nerode equivalence by refinements of partitions of the set of states. Given an n-state DFA, Moore's algorithm has quadratic time complexity on the worst-case, but on the average-case it is bounded by $O(n \log n)$. On the other hand, Hopcroft's algorithm [Hop71] runs in $O(n \log n)$ in the worst-case, through a more complex order of operations then Moore's. Brzozowski's algorithm [Brz62] is different from the other two, since its inputs may be nondeterministic automata. Reversing the transitions of

an NFA and switching initial and final states produces an NFA for the reversal of the original language, and Brzozowski observed that reversing the automaton and converting it to a DFA *twice* produces the minimal DFA of the original language. The worst-case complexity is, of course, exponential in the number of state of the input DFA due to the determinisation steps.

The *minimisation of nondeterministic finite automata* is considered a challenge due to its computation complexity. Unlike their deterministic counterpart, where minimisation can be achieved efficiently, the problem of minimising NFAs is classified as PSPACE-COMPLETE [MS72], so it is unknown the existence of polynomial-time algorithms, but there are some procedures to obtain a minimal NFA [BT13]. Moreover, there are several algorithms with a practical performance that permit to reduce the size of the NFAs, without guarantees that the obtained NFA is the smallest one, p.e. *bisimulations* [IY03].

2.5 Regular Expressions

Introduced by Kleene in 1956 [Kle56], regular expressions are a declarative way to express the words that we want to accept. They are used in pattern matching applications such as input validation, search engines, and word processors.

The class of regular expressions (RE) and the languages they represent, $\mathcal{L}(\cdot)$, can be recursively defined as follows:

- $\emptyset$ and ε are RE, $\mathcal{L}(\emptyset) = \emptyset$ and $\mathcal{L}(\varepsilon) = \{\varepsilon\}$;
- σ is a RE and $\mathcal{L}(\sigma) = \{\sigma\}$, for $\sigma \in \Sigma$;
- if α and β are RE, then
  1. $\alpha + \beta$ is also a RE, and $\mathcal{L}(\alpha + \beta) = \mathcal{L}(\alpha) \cup \mathcal{L}(\beta)$,
  2. $\alpha\beta$ is also a RE, and $\mathcal{L}(\alpha\beta) = \mathcal{L}(\alpha)\mathcal{L}(\beta)$;
- if α is a RE, then $\alpha^\star$ is also a RE, and $\mathcal{L}(\alpha^\star) = \mathcal{L}(\alpha)^\star$.

Example 2.5. *The language* $L = \{w \in \{a, b\}^\star \mid$ *the third but last symbol is* $a\}$*, represented by an NFA in Figure 2.1, can also be described by the regular expression*

$$\alpha = (a + b)^\star a(a + b)(a + b).$$

The Thompson algorithm [Tho68] converts any regular expression into an ε-NFA that recognises the same language. Glushkov [Glu61] introduced the *position automata* that permits us to

convert a regular expression into an equivalent NFA without ε-transitions. The existence of these algorithms imply the following result:

Proposition 2.6. *For each regular expression, there is a finite automaton that recognises the same language.*

On the other hand, the Kleene's algorithm [Kle56] transforms an NFA into a regular expression. Brzozowski and McCluskey [BJ63] presented an algorithm that given an automaton without any ε-transitions, it progressively eliminates the states until it ends up with an automaton having a single transition. The label of that transition corresponds to the regular expression of the language recognised by the automaton. Then, we have:

Proposition 2.7. *For each language represented by a finite automaton, there is a regular expression that recognises the same language.*

As a corollary of Propositions 2.6 and 2.7, we have that the set of regular languages and the set of languages recognised by a finite automaton are equal.

2.6 State Complexity

The ability of finite automata to represent a language in a more or less succinct way has been widely studied in the area of descriptional complexity. In this context, the size of a model is measured in terms of the number of symbols used to write down its description. The *state complexity* of a language L, $\mathsf{sc}(L)$, is the size of its minimal DFA (number of states). The *nondeterministitic state complexity* of a language L, $\mathsf{nsc}(L)$, is defined analogously. Since a DFA is in particular an NFA, for any regular language L one has $\mathsf{nsc}(L) \leqslant \mathsf{sc}(L)$. As we saw in Section 2.4.1, converting an n-state NFA to an equivalent DFA can lead to an automaton with up to 2^n states. Thus, $\mathsf{sc}(L) \leqslant 2^{\mathsf{nsc}(L)}$.

The *operational state complexity* is the worst-case state complexity of a language resulting from an operation, and is given as a function of the state complexities of the operands. For instance, the deterministic state complexity of the intersection of two languages can be stated as follows:

Example 2.8. *Given an m-state DFA $\mathcal{A}_1$ and an n-state DFA $\mathcal{A}_2$, how many states are sufficient and necessary, in the worst-case, to accept the language $\mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2)$ by a DFA $\mathcal{A}_3$? And how large does the alphabet have to be to achieve that bound?*

An upper bound can be obtained by providing an algorithm that, given automata for the operands, constructs an automaton that accepts the language resulting from the operation

applied to the language of the operands. The number of states, in the worst-case, of the resulting machine is an upper bound for the state complexity of the referred operation.

Example 2.9. *Proceeding with the intersection operation, let*

$$\mathcal{A}_1 = \langle Q, \Sigma, \delta_1, q_0, F_1 \rangle \text{ and } \mathcal{A}_2 = \langle P, \Sigma, \delta_2, p_0, F_2 \rangle$$

be two minimal DFAs for L_1 *and* L_2*, respectively, where* $sc(L_1) = m$ *and* $sc(L_2) = n$*. Let* $\mathcal{A}_3$ *be the DFA resulting from the product automata of* $\mathcal{A}_1$ *and* $\mathcal{A}_2$*, that is,*

$$\mathcal{A}_3 = \langle S, \Sigma, \delta_3, I, F_3 \rangle ,$$

where:

- $S = Q \times P$*;*
- $(\forall (p, q) \in S)(\forall \sigma \in \Sigma) : \delta_3((p, q), \sigma) = (\delta_1(p, \sigma), \delta_2(q, \sigma))$*;*
- $I_3 = (q_0, p_0)$*;*
- $F_3 = F_1 \times F_2$*.*

It is not hard to see that $\mathcal{A}_3$ *recognises the language* $L_1 \cap L_2$ *and that it has exactly* mn *states. Therefore,* $sc(L_1 \cap L_2) \leqslant mn$*.*

To show that an upper bound is *tight*, a *family of languages* for each operation must be given such that the resulting automata achieve that bound. We call those families *witnesses* or *streams*.

Example 2.10. *For the operational state complexity of the intersection of two regular languages* L_1 *and* L_2 *with* $sc(L_1) = m$ *and* $sc(L_2) = n$*, the bound* mn *is indeed tight. Let* $m, n \geqslant 1$ *and the languages*

$$L_1 = \{ w \in \{a, b\}^\star \mid |w|_a \equiv 0 \pmod{m} \},$$
$$L_2 = \{ w \in \{a, b\}^\star \mid |w|_b \equiv 0 \pmod{n} \}.$$

It is easy to see that both $sc(L_1) = m$ *and* $sc(L_2) = n$*, as Figure 2.5 suggests. Yu et al. [YZS94] showed that any DFA that recognises the language* $L_1 \cap L_2$ *requires at least* nm *states (see Figure 2.6).*

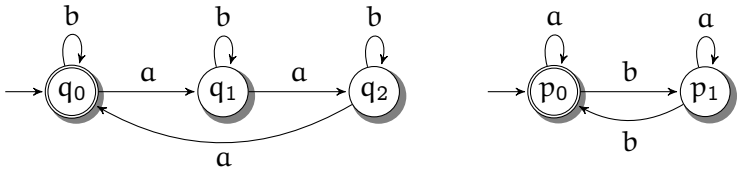

Figure 2.5: Minimal DFAs for L_1 (left) and L_2 (right), defined in Example 2.10, with $m = 3$ and $n = 2$.

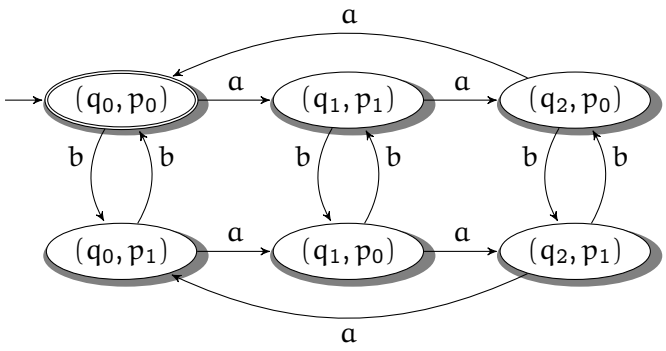

Figure 2.6: Minimal DFA with 6 states for the intersection of L_1 and L_2, defined in Example 2.10, with $m = 3$ and $n = 2$.

In Table 2.1, we review some complexity bounds for regular languages, as well as the size of the smallest alphabets for the family witnesses that attain the bound. When considering unary operations, let $L \subseteq \Sigma^\star$ be a regular language with $\mathrm{sc}(L) = m$ ($\mathrm{nsc}(L) = m$) and let $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F\rangle$ be the complete minimal DFA (a minimal NFA) for L. Furthermore, $k = |\Sigma|$, $f = |F|$, and $l = |F - \{q_0\}|$. In the same way, for binary operations let L_1 and L_2 be regular languages over the same alphabet with $\mathrm{sc}(L_1) = m$ ($\mathrm{nsc}(L_1) = m$) and $\mathrm{sc}(L_2) = n$ ($\mathrm{nsc}(L_2) = n$), and let $\mathcal{A}_i = \langle Q_i, \Sigma, \delta_i, q_i, F_i\rangle$ be complete minimal DFAs (minimal NFAs) for L_i, with $i \in \{1, 2\}$. Furthermore, $k = |\Sigma|$, $f_i = |F_i|$, and $l_i = |F_i - \{q_i\}|$, for $i \in \{1, 2\}$.

Yu *et al.* [YZS94] studied, among others, the deterministic state complexity of union, intersection, star, and reversal. The complement for DFAs is trivial as one has only to exchange the final states and thus, the state complexity of the complement is the same one of the original language. Jirásek *et al.* [JJS05] studied the bound for concatenation of two languages and proved that it is tight. The state complexity for the plus operation on a regular language L coincides with the one for star in the first case, but one state can be saved. The state complexity of basic operations on NFAs was first studied by Holzer and Kutrib [HK03a], and also by Ellul [EoWDoCS02]. Jiráskova [Jir05] proved the bounds for the nondeterministic state complexity of complement, star and reversal. An NFA accepting L^+ coincides with one accepting L except that each final state has also the transitions to the initial state.

Regular Languages				
	sc	$\lvert\Sigma\rvert$	nsc	$\lvert\Sigma\rvert$
$L_1 \cup L_2$	mn	2	$m+n+1$	2
$L_1 \cap L_2$	mn	2	mn	2
$\overline{L}$	m	1	2^m	2
L_1L_2	$m \cdot 2^n - f_1 \cdot 2^{n-1}$, if $n > 1$	2	$m+n$	2
	m, if $n = 1$	1		
$L^\star$	$2^{m-1} + 2^{m-l-1}$, if $m > 1$, $l > 0$	2	$m+1$	2
	m, if $m > 1$, $l = 0$	1		
	$m+1$, if $m = 1$	1		
L^+	$2^{m-1} + 2^{m-l-1} - 1$	2	m	2
L^R	2^m	2	$m+1$	2

Table 2.1: State complexity and nondeterministic state complexity for basic operations on regular languages.

For an in depth study on operational state complexity over multiple classes of regular languages, we refer the reader to [GMRY17].

Chapter 3

Finite Languages

The class of *finite languages* is a proper subset of regular languages, because every finite language can be represented as a finite union of singletons, each corresponding to an individual word in the language. Additionally, finite automata and regular expressions for these languages, have structural properties that lead to more efficient manipulation algorithms and descriptional complexity, as will be shown in Sections 3.1 to 3.3. Moreover, we will also consider languages where all words have the same length in Section 3.4, as well as their representations using automata and some applications.

Let us denote as $\Sigma^{\leqslant \ell}$, for $\ell \geqslant 0$, the set of words over Σ of length at most ℓ. That is,

$$\Sigma^{\leqslant \ell} = \Sigma^0 \cup \Sigma^1 \cup \ldots \cup \Sigma^\ell.$$

A finite language L which the longest word has length ℓ is a subset of $\Sigma^{\leqslant \ell}$.

3.1 Finite Automata for Finite Languages

A trim NFA $\mathcal{A} = \langle Q, \Sigma, \delta, I, F \rangle$, that is, an NFA which every state is both accessible and co-accessible, for a finite language $L \subseteq \Sigma^{\leqslant \ell}$, with $\ell \geqslant 0$, is *acyclic* and *ranked*. An NFA is:

- *acyclic* if it has no cycles, meaning that there are no paths of non-zero length within the automaton that allow a return to a previously visited state. Formally,

$$\mathcal{A} \textit{ is acyclic } \Leftrightarrow (\forall q \in Q)(\forall w \in \Sigma^+) : \delta(q, w) \neq q;$$

- *ranked* if the set of states Q can be partitioned into $\ell + 1$ disjoint sets $Q_0, Q_1, \cdots, Q_\ell$, such that the longest word accepted from each state in Q_i is i. For each $q \in Q$, we first

define $\mathsf{rank}(q)$ as the largest final path starting at q, i.e.,

$$\mathsf{rank} : Q \to [\ell]$$
$$\mathsf{rank}(q) = \max\{\, |w| \mid (w \in \Sigma^\star) : \delta(q, w) \cap F \neq \emptyset \,\}.$$

Then, we formally define

$$\begin{aligned} \mathcal{A} \textit{ is ranked } \Leftrightarrow\ & Q = Q_0 \cup Q_1 \cup \cdots \cup Q_\ell \\ & \wedge\, (\forall i, j \in [\ell]) : i \neq j \implies Q_i \cap Q_j = \emptyset \\ & \wedge\, (\forall i \in [\ell]) : Q_i = \{\, q \in Q \mid \mathsf{rank}(q) = i \}. \end{aligned}$$

We also say that $\mathsf{rank}(\mathcal{A}) = \max\{\, \mathsf{rank}(q_0) \mid q_0 \in I \,\}$. As an NFA for a finite language is acyclic, we have $0 \leqslant \mathsf{rank}(q) \leqslant \ell$, for every $q \in Q$, and that all transitions from states of rank i lead only to states in ranks j, such that $j < i$. We define the *width* of a rank $i \in [\ell]$, namely $\mathsf{width}(i)$, as the cardinality of the set Q_i, and the width of an NFA $\mathcal{A}$ to be the maximal width of a rank, i.e., $\mathsf{width}(\mathcal{A}) = \max\{\, \mathsf{width}(i) \mid i \in [\ell] \,\}$.

A DFA for a finite language is also ranked but it may have a sink-state Ω which is the only state with a self-loop and without a rank. The notion of ranks and widths analogously apply to DFAs.

Example 3.1. *See Figure 3.1 for the minimal DFA of* $L = \{\, ww^R \mid w \in \Sigma^{\leqslant 2} \,\}$*, the language of even size palindromes over an alphabet* Σ*. Note that the general case of infinite languages of palindromes is not regular. In particular, we have that* $\mathsf{rank}(q_9) = 0$*, since* ε *is the largest word* w *such that* $\delta(q_9, w)$ *is a final state. On the other hand,* $\mathsf{rank}(q_0) = 4$*, because there exists a path from* q_0 *to* q_9 *of length 4. The width of this DFA is* 4 *and it is achieved at rank* 2 *(*$\mathsf{width}(2) = 4$*).*

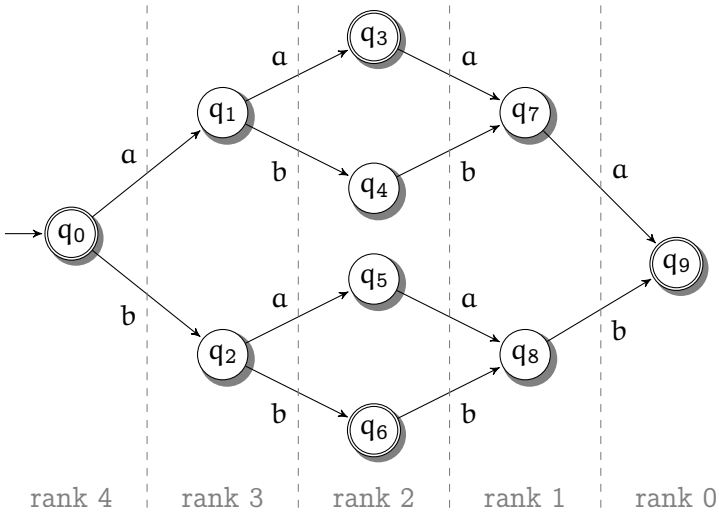

Figure 3.1: A DFA for the finite language $L = \{\, ww^R \mid w \in \Sigma^{\leqslant 2} \,\}$ with the division of states by their ranks. The sink-state Ω is omitted, as well as all transitions from and to it.

3.2 Minimisation of Finite Automata for Finite Languages

As seen in Section 2.4.2, a complete and trim DFA for a finite language is *minimal* if it has no indistinguishable states. However, since any finite automaton for a finite language is ranked, we have the following result:

Lemma 3.2. *Let $\mathcal{A} = \langle Q, \Sigma, \delta, I, F\rangle$ be a DFA for a finite language $L \subseteq \Sigma^{\leqslant \ell}$, for some $\ell \geqslant 0$. Then, if q, q' of Q are indistinguishable they have the same rank, that is,* $\mathsf{rank}(q) = \mathsf{rank}(q')$.

Proof. Two states q and q' are indistinguishable if, and only if, their right language is the same, that is, $\mathcal{L}_q(L) = \mathcal{L}_{q'}(L)$. If $\mathsf{rank}(q) \neq \mathsf{rank}(q')$ and supposing, w.l.o.g., that $\mathsf{rank}(q) > \mathsf{rank}(q')$, then there would exist a word $w \in \mathcal{L}_q(L)$ of length $\mathsf{rank}(q)$ that would not be in $\mathcal{L}_{q'}(L)$. So, it follows that $\mathsf{rank}(q) = \mathsf{rank}(q')$. □

Of course, a minimisation of an acyclic DFA can be done by any algorithm mentioned in Section 2.4.2, but a much more efficient method can be applied. Revuz algorithm [AMR08, Rev92] iteratively minimises each rank, from lowest to highest, using the observation in Lemma 3.2. This algorithm runs in $O(kn)$, where $k = |\Sigma|$ and n is the number of states of the original automaton, outperforming the other techniques.

```
1  def minimisationRevuz(A):
2    assert A is acyclic
3    let A = ⟨Q, Σ, δ, q0, F⟩
4    M = ∅
5    for i ∈ [rank(A)] do
6      for q, q' ∈ Q_i ∧ q ≺ q' do
7        if (q ∈ F ⇔ q' ∈ F) ∧ (Rnm(M, δ(q, σ)) = Rnm(M, δ(q', σ)))_{σ∈Σ} then
8          M ← M ∪ {(q, q')}
9          delete(q)
10   return A
11
12  def Rnm(M, q)
13    if (∃q' ∈ Q) : (q, q') ∈ M then return q'
14    else return q
```

Algorithm 3.1: Revuz Algorithm for minimisation of acyclic DFAs.

A sketch of this algorithm can be seen in Algorithm 3.1. It receives as input an acyclic DFA $\mathcal{A}$ and returns an equivalent minimal DFA. It assumes that $\prec$ is any total order relation on the sets Q_i, for every $i \in [\mathsf{rank}(\mathcal{A})]$, and that Σ is ordered. As the name suggests, the function $\mathit{delete}(q)$

removes q from Q and all related transitions. The correctness of this algorithms follows from the fact that every transition from a state in rank i has a state in rank j as target, with $j < i$, alongside the result in Lemma 3.2.

Câmpeanu and Ho [CH04] determined the exact maximal size of a minimal DFA for a finite language as a function of the size of the alphabet and the length of the largest word of the language.

Theorem 3.3 ([CH04]). *Let $L \subseteq \Sigma^{\leqslant \ell}$, with $\ell \geqslant 1$ and $k = |\Sigma|$, be a finite language recognised by an m-state NFA. Then, the DFA obtained from this NFA by the subset construction has at most $O(k^{\frac{m}{1+\log k}})$ states.*

Salomaa and Yu [SY97] showed that the blow-up of determinising an acyclic NFA is smaller than for the general case.

Theorem 3.4 ([SY97]). *Let $L \subseteq \Sigma^{\leqslant \ell}$, with $\ell \geqslant 1$ and $k = |\Sigma|$, be a finite language. Then, a DFA with*

$$O(k^{\frac{m}{1+\log k}})$$

states is sufficient to recognise L.

3.3 Operational State Complexity for Finite Languages

The *operational state complexity for finite languages* is a well-studied matter. Notably, the operational state complexity for finite languages is generally lower compared to the general case of regular languages, both on the deterministic and nondeterministic case.

In Table 3.1, we review some complexity bounds for finite languages. Recall the notation used in Table 2.1 defined in Section 2.6. Here, we say that $|\Sigma| = f(m, n)$ if the size of Σ must be a function of m and n, in which case we say that Σ is a *growing alphabet*.

Han *et al.* [HS08] proved that the upper bound for union and intersection of DFAs cannot be reached with a fixed alphabet, and presented witnesses that use a growing alphabet. Câmpeanu *et al.* [CCSY01] gave tight upper bounds for concatenation, star and reversal. The nondeterministic state complexity of finite languages was studied by Holzer and Kutrib [HK03b].

Finite Languages				
	sc	$\lvert\Sigma\rvert$	nsc	$\lvert\Sigma\rvert$
$L_1 \cup L_2$	$mn - (m+n)$	$f(m,n)$	$m+n-2$	2
$L_1 \cap L_2$	$mn - 3(m+n) + 12$	$f(m,n)$	$O(mn)$	2
$\overline{L}$	m	1	$\Theta(k^{\frac{m}{1+\log k}})$	2
L_1L_2	$(m-n+3)2^{n-2}-1$, if $m+1 \geqslant n$ $m+n-2$, if $l_1 = 1$	2 1	$m+n-1$	2
$L^\star$	$2^{m-3}+2^{m-l-2}$, if $l \geqslant 2$, $m \geqslant 4$ $m-1$, if $f = 1$	3 1	$m-1$, if $m > 1$	1
L^+	m	1	m, if $m > 1$	1
L^R	$O(k^{\frac{m}{1+\log k}})$	2	m	2

Table 3.1: State complexity and nondeterministic state complexity for basic operations on finite languages.

3.4 Block Languages

We will now consider languages where all words have the same length. These languages are commonly referred in the literature as *homogeneous* or *block languages*, and we will use the latter throughout this work. Formally, a block language L over an alphabet Σ and *block length* $\ell \in \mathbb{N}$ is a subset of Σ^ℓ, that is, $L \subseteq \Sigma^\ell$. In Section 3.4.1 we will look into some gains in terms of descriptional complexity of these languages, and some applications in Section 3.4.2.

3.4.1 Finite Automata for Block Languages

A minimal finite automaton $\mathcal{A} = \langle Q, \Sigma, \delta, I, F\rangle$ for a block language $L \subseteq \Sigma^\ell$, with $\ell \geqslant 0$ and $k > 0$, is also acyclic and ranked as in the general case for finite languages. Therefore, the set of states Q can also be split into $\ell+1$ disjoint sets as $Q = Q_0 \cup Q_1 \cup \ldots \cup Q_\ell$, according to the rank of each state. However, the right language of each state $q \in Q_i$, for some $i \in [\ell]$, contains only words of length i, that is, $\mathcal{L}_q(\mathcal{A})$ is also a block language. As a consequence, transitions in $\mathcal{A}$ are from one rank to the previous one, i.e., if $q' \in \delta(q,\sigma)$ then $q \in Q_i$ and $q' \in Q_{i-1}$, for some $i \in [1,\ell]$. Also, we have $|I| = |F| = 1$, as every state in the respective ranks can be collapsed with a single state.

Karhumäki and Okhotin [KO14] studied the cost in terms of number of states of determinising an m-state NFA which recognises a block language, and they showed that it is smaller than the

general case of finite languages (see Table 3.1). In the same paper, they also proposed a family of witness languages which are recognised by small NFAs but any DFA must have much more states, proving that the provided bound is in fact tight.

Theorem 3.5 ([KO14]). *Let* $L \subseteq \Sigma^\ell$*, with* $\ell \geqslant 1$ *and* $k = |\Sigma|$*, be a block language recognised by an* m*-state NFA. Then, the DFA obtained from this NFA by the subset construction has at most* $2^{O(\sqrt{m})}$ *states.*

Câmpeanu and Ho determined the maximum number of states required to recognise a block language using a minimal deterministic finite automaton.

Theorem 3.6 ([CH04]). *Let* $L \subseteq \Sigma^\ell$*, with* $\ell \geqslant 1$ *and* $k = |\Sigma|$*, be a block language. Then, a DFA with*

$$\sum_{i=0}^{\ell} \min(2^{\ell-i}, 2^{k^\ell} - 1)$$

states is sufficient to recognise L*.*

Hanssen and Liu [KL19] provide a formula for determining the number of block languages with maximal state complexity, that is, the number of languages which minimal DFAs require exactly the number of states in Theorem 3.6. Brzozowski and Konstantinidis [BK09] studied the state complexity of languages where the block length equals the size of the alphabet, that is, languages $L \subseteq \Sigma^\ell$ such that $|\Sigma| = \ell$.

3.4.2 Applications

Finite automata can be used to represent two-dimensional images [KK21, KO14], wherein the value of each pixel is defined by the computation of the automaton using as input the word that represents the pixel's coordinates. For example, consider a square picture with resolution $2^\ell \times 2^\ell$. The coordinates of each pixel in the image are defined by a string of length ℓ over a four-letter alphabet $\Sigma = \{a, b, c, d\}$. Given a word $w \in \Sigma^\ell$ the corresponding pixel is given by progressively subdividing the image into smaller squares, using each symbol of w to choose one of the quadrants. In Figure 3.2 it is shown how to access the pixel defined by the word *dbc* in a 8×8 image.

Using this encoding of pixel's locations, one can define a black-and-white image as a formal language, by listing the coordinates of all black (or white) pixels as the corresponding words. These languages will be, of course, finite and, in particular, block as every pixel requires the same amount of symbols to encode its position. In Figure 3.3 it is depicted the image represented by the set of words of size 7 over the alphabet $\Sigma = \{a, b, c, d\}$ that have at least two a's.

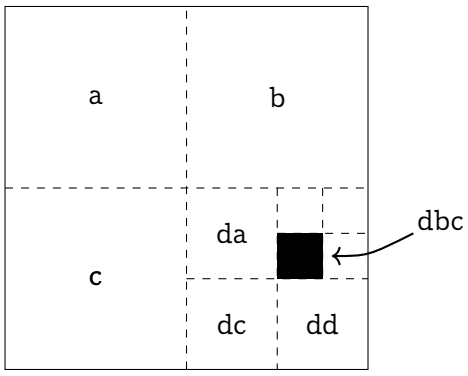

Figure 3.2: Accessing a pixel of a 8×8 image by a word of length 3.

Moreover, operations on images become operations on languages. For instance, the image obtained by inverting the colours of an image represented by a block language $L \subseteq \Sigma^\ell$, is represented by the language $\Sigma^\ell \setminus L$, that is, the complement of L closed for words of length ℓ. Karhumäki *et al.* [KPR03] studied the *zoom* operation on images, that is, given an n-state DFA representing a $2^\ell \times 2^\ell$ image, the task is to produce a DFA for a $2^{\ell'} \times 2^{\ell'}$ subimage of the image, with $\ell' \leqslant \ell$. They concluded that this operation is represented by a language which minimal DFA requires $\Theta(n^{2.5})$ states and that the bound is tight.

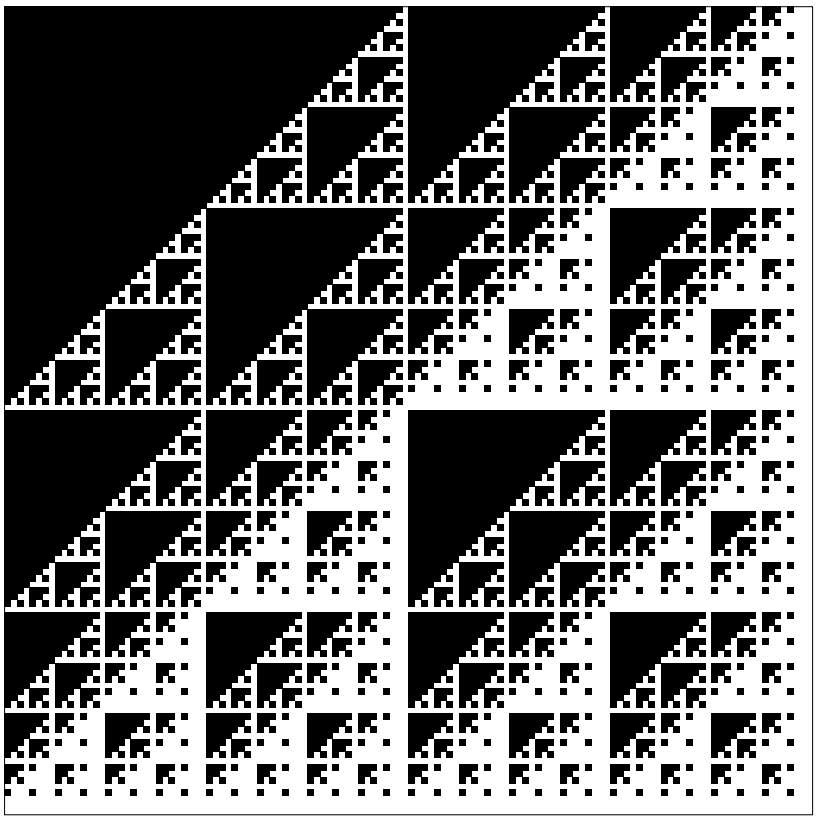

Figure 3.3: Image with resolution $2^\ell \times 2^\ell$ represented by the language $L = \{ w \in \Sigma^\ell \mid |w|_a \geqslant 2 \}$, for $\ell = 7$ and $\Sigma = \{a, b, c, d\}$.

Block languages are also used in code theory. In *block coding*, a sequence of information over a finite alphabet is divided into *message blocks* of fixed length, each consisting of n information symbols. Then, each message is encoded into a distinct *code-word* of length $\ell > n$, where the extra symbols are used to detect and correct possible errors introduced by an imperfect channel. This set of encoded message is often called a *block code*. In this field, a DFA for a block language L is referred as a *trellis* for a block code C. In [SDDR03], the authors study the partition of a block code C into disjoint subcodes, and algorithmically combine the minimal subtrellis corresponding to these subcodes in order to get a reduced trellis for C. A typical problem in code theory is the construction of *maximal* code blocks for error correcting [DK12, KMR18]. A code block C is said to be maximal if extending it with additional code-words would compromise its ability to detect and correct errors. Konstantinidis *et al.* [KMR18] proved that the problem of deciding if a trellis recognises a maximal block code is coNP-HARD.

Chapter 4

Bitmaps for Block Languages

In this chapter, we introduce a new representation for block languages, namely sets of words that have the same size, and call them *bitmaps*. In Section 4.1 we define what a bitmap is, as well as several associated properties. In Section 4.2 and Section 4.3, we explain how one can construct the minimal DFA and a minimal NFA, respectively, from a bitmap representation of a block language.

4.1 Bitmaps

Given an alphabet $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_k\}$ of size $k > 0$ and an integer $\ell \geqslant 0$, let $L \subseteq \Sigma^\ell$ be a block language of block length $\ell \geqslant 0$ over Σ. The language L can be characterised by a word in $\{0,1\}^{k^\ell}$ that we call *bitmap* and denote as

$$B(L) = b_1 b_2 \cdots b_{k^\ell},$$

where $b_i = 1$ if the i-th word of Σ^ℓ is in L, in the lexicographical order. In this case, we denote i by $\mathsf{ind}(w)$. We will denote the bitmap of a language simply as B when it is unambiguous to which language the bitmap refers to. We also say that $\mathcal{L}(B)$ is the block language that B represents.

Example 4.1. *Let* $\Sigma = \{a, b\}$, $\ell = 4$, *and*

$$L = \{aaaa, aaba, aabb, abab, abba, abbb, babb, bbaa, bbab, bbba\}.$$

The bitmap of L *is*

$$B = 1011011100011110.$$

For example, $b_1 = 1$ *since* $aaaa \in L$ *and* $\mathsf{ind}(aaaa) = 1$. *Also,* $b_7 = 1$ *because* $\mathsf{ind}(abba) = 7$ *and* $abba \in L$. *On the other hand,* $b_{16} = 0$ *as* $bbbb \notin L$.

Given a block length $\ell > 0$ and an alphabet size $k > 0$, it is easy to see that no two block languages have the same bitmap characterisation, unless they are over different alphabets. Thus, the following result applies:

Lemma 4.2. *A bitmap* $B \in \{0,1\}^{k^\ell}$*, for some alphabet* Σ *of size* $k > 0$ *and block length* $\ell > 0$*, represents a unique language up to renaming of alphabet symbols.*

Proof. Let $L_1, L_2 \subseteq \Sigma^\ell$ be two different block languages over the same alphabet Σ. Because $L_1 \neq L_2$, there exists a word $w \in L_1$ such that, w.l.o.g., $w \notin L_2$. So, their bitmap representation should be different at the $\mathsf{ind}(w)$-th bit. Thus, $B(L_1) \neq B(L_2)$. □

The only exception of the previous lemma are the languages $\{\varepsilon\}$ and $\{\sigma^\ell\}$. Since the first language has block length $\ell = 0$ and the second is a unary language ($k = 1$), both these languages get represented by the same bitmap $B = 1$.

A bitmap $B \in \{0,1\}^{k^\ell}$ can be *divided into segments* of length k^i, for $i \in [\ell]$. That is, the bitmap B can be seen as

$$B = s_1^i \, s_2^i \cdots s_{k^{\ell-i}}^i,$$

such that s_j^i denotes the j-th segment of length k^i, for each $j \in [1, k^{\ell-i}]$. Whenever $i > 0$, each segment of length k^i can also be split into k segments, so s_j^i is inductively defined as

$$s_j^i = \begin{cases} b_j, & \text{if } i = 0; \\ s_{(j-1)k+1}^{i-1} \, s_{(j-1)k+2}^{i-1} \cdots s_{jk}^{i-1}, & \text{otherwise.} \end{cases}$$

Example 4.3. *Recall Example 4.1, where* $B = 1011011100011110$*,* $k = 2$ *and* $\ell = 4$*. We have, for instance, when* $i = 1$

$$s_1^1 = s_8^1 = 10,\ s_2^1 = s_4^1 = s_7^1 = 11,\ s_3^1 = s_6^1 = 01,\ \textit{and}\ s_5^1 = 00.$$

For $i = 2$*,*

$$s_1^2 = 1011,\ s_2^2 = 0111,\ s_3^2 = 0001,\ \textit{and}\ s_4^2 = 1110.$$

For $i = 4$*, we have* $s_1^4 = B$*.*

The following lemma corresponds to the observation that each segment s_j^i of a bitmap $B \in \{0,1\}^{k^\ell}$, over a k-letter alphabet, a block length $\ell \geqslant 0$, $i \in [\ell]$, and $j \in [1, k^{\ell-i}]$, represents the bitmap of a quotient of L, that is, $s_j^i = B(w^{-1}L)$, for some $w \in \Sigma^{\ell-i}$.

Lemma 4.4. *Let* $L \subseteq \Sigma^\ell$ *be a block language, where* $|\Sigma| = k$ *and* $\ell \geqslant 0$*, and let* B *be the bitmap of* L*. Also, let* $i \in [\ell]$*,* $j \in [1, k^{\ell-i}]$*, and* $w \in \Sigma^{\ell-i}$ *be the* j*-th word of length* $\ell - i$*, i.e.,* $\mathsf{ind}(w) = j$*. Then,* s_j^i *corresponds to the bitmap representation of the language* $w^{-1}L$*.*

Proof. Let us prove by induction on $i \in [\ell]$:

- *Base case* $i = 0$: by definition, $s^i_j = b_j$. By construction of the bitmap, we have that $b_j = 1$ if the word $w \in \Sigma^\ell$ with $\mathsf{ind}(w) = j$ is in L. Since $|w| = \ell$, either $w^{-1}L = \{\varepsilon\}$ or $w^{-1}L = \emptyset$, according to the membership or not of w in L.
- *Inductive step*: since $i > 0$, we have that $s^i_j = s^{i-1}_{(j-1)k+1} s^{i-1}_{(j-1)k+2} \cdots s^{i-1}_{jk}$. By hypothesis, we have that $s^{i-1}_{(j-1)k+k_0}$, for each $k_0 \in [1, k]$, corresponds to the bitmap of the language $w^{-1}_{k_0}L$, where $|w_{k_0}| = \ell - (i-1)$ and $\mathsf{ind}(w_{k_0}) = (j-1)k + k_0$. One can observe that the words $\{w_{k_0}\}_{k_0 \in [1,k]}$ are all equal on the first $\ell - i$ symbols, corresponding to the j-th word of length $\ell - i$. Thus, s^i_j corresponds to the bitmap of the language $w^{-1}L$ such that $|w| = \ell - i$ and $\mathsf{ind}(w) = j$.

□

Example 4.5. *Recall again Example 4.1, where* $B = 1011011100011110$, $\Sigma = \{a, b\}$ *and the block length is* $\ell = 4$. *We have that* $s^2_1 = 1011$ *is the bitmap of the language* $(aa)^{-1}L = \{aa, ba, bb\}$, $s^3_2 = 00011110$ *is the bitmap of* $b^{-1}L = \{abb, baa, bab, bba\}$, *and* $s^4_1 = B$ *is the bitmap of* $\varepsilon^{-1}L = L$.

Given a bitmap $B \in \{0,1\}^{k^\ell}$ of a block language over an alphabet of size $k > 0$ and a block length $\ell \geqslant 0$, let $\mathcal{B}_i$ be the set of segments of B of length k^i, for $i \in [\ell]$, in which there is at least one bit different than zero. Formally,

$$\mathcal{B}_i = \{\, s \in \{0,1\}^{k^i} \mid (\exists j \in [1, k^{\ell-i}]) : s = s^i_j \text{ and } s^i_j \neq 0^{k^i} \,\}.$$

Example 4.6. *For the bitmap of Example 4.1,* $B = 1011011100011110$ *with* $k = 2$ *and* $\ell = 4$, *we have* $\mathcal{B}_0 = \{1\}$, $\mathcal{B}_1 = \{10, 11, 01\}$, $\mathcal{B}_2 = \{1011, 0111, 0001, 1110\}$, $\mathcal{B}_3 = \{10110111, 00011110\}$, *and* $\mathcal{B}_4 = \{B\}$.

The size of $\mathcal{B}_i$ is bounded by the number of segments with size k^i, for each $i \in [\ell]$. Also, each segment is a composition of segments from the previous set $\mathcal{B}_{i-1}$ or by zeros. These two conditions are formally stated in the following lemma.

Lemma 4.7. *Let* $L \subseteq \Sigma^\ell$ *be a block language of length* $\ell \geqslant 0$ *over an* k*-letter alphabet* Σ, *with a correspondent bitmap* B. *Then, for each* $i \in [\ell]$, *the cardinality of* $\mathcal{B}_i$ *is bounded by* $|\mathcal{B}_i| \leqslant \min(k^{\ell-i}, 2^{k^i} - 1)$.

Proof. When $i = 0$, the result is trivial as $\mathcal{B}_0$ contains at most the segment 1. This happens when L is not empty. Since there are at most $k^{\ell-i}$ unique segments of size k^i in a bitmap of

size k^ℓ, for $i \in [\ell]$, then $|\mathcal{B}_i| \leqslant k^{\ell-i}$. Now, let $s \in \mathcal{B}_i$, for some $i \in [1,\ell]$. By definition, s can be partitioned into $s = s_1 s_2 \cdots s_k$ (with $|s_j| = k^{i-1}$) and either $s_j \in \mathcal{B}_{i-1}$ or it is composed only by zeros (implying that $s_j \notin \mathcal{B}_{i-1}$), for every $j \in [1,k]$. Moreover, since $s \in \mathcal{B}_i$ it must have at least one bit equal to 1, so $s \neq 0^{k^i}$. Therefore, $|\mathcal{B}_i| \leqslant (|\mathcal{B}_{i-1}|+1)^k - 1$. For simplicity, let $f(i) = (|\mathcal{B}_{i-1}|+1)^k - 1$ and let us prove that $f(i) \leqslant 2^{k^i} - 1$, by induction on $i \in [1,\ell]$.

- *Base case* $i = 1$: $f(1) = (|\{1\}|+1)^k - 1 = 2^k - 1$;
- *Inductive step*:

$$\begin{aligned} f(n+1) &= (|\mathcal{B}_n|+1)^k - 1 \\ &\leqslant ((|\mathcal{B}_{n-1}|+1)^k - 1 + 1)^k - 1 \\ &\overset{\text{i.h.}}{\leqslant} (2^{k^n} - 1 + 1)^k - 1 = 2^{k^{n+1}} - 1. \end{aligned}$$

Thus, $|\mathcal{B}_i| \leqslant 2^{k^i} - 1$, as desired. □

One can also consider bitwise operations on bitmaps which correspond to set operations on block languages. Let $s_1 = b_1 b_2 \cdots b_{k^\ell}$, and $s_2 = b'_1 b'_2 \cdots b'_{k^\ell}$ be two bitmaps of block languages over Σ^ℓ, with $|\Sigma| = k$ and $\ell \geqslant 0$. We define:

1. $s_1 \wedge s_2 = (b_1 \wedge b'_1)(b_2 \wedge b'_2) \cdots (b_{k^\ell} \wedge b'_{k^\ell})$ and $\mathcal{L}(s_1 \wedge s_2) = \mathcal{L}(s_1) \cap \mathcal{L}(s_2)$;
2. $s_1 \vee s_2 = (b_1 \vee b'_1)(b_2 \vee b'_2) \cdots (b_{k^\ell} \vee b'_{k^\ell})$ and $\mathcal{L}(s_1 \vee s_2) = \mathcal{L}(s_1) \cup \mathcal{L}(s_2)$;
3. $\overline{s_1} = (\neg b_1)(\neg b_2) \cdots (\neg b_{k^\ell})$ and $\mathcal{L}(\overline{s_1}) = \Sigma^\ell \setminus \mathcal{L}(s_1)$.

In the following sections, we will describe how one can construct minimal finite automata, both deterministic and nondeterministic, from a bitmap representation of a block language.

4.2 Minimal DFAs for Block Languages Described by Bitmaps

In this section we relate the bitmap of a block language to its minimal DFA. Given a bitmap B representing a block language $L \subseteq \Sigma^\ell$, with $|\Sigma| = k$ and $\ell \geqslant 0$, one can directly build a minimal DFA $\mathcal{A}$ for L. Let $Q = \bigcup_{i=0}^{\ell} \mathcal{B}_i$ be the set of states of $\mathcal{A}$, and the transition function δ will map the states in $\mathcal{B}_i$ with the ones in $\mathcal{B}_{i-1}$, for $i \in [1,\ell]$. We will now detail this construction.

We start by the final state q_f, which will be the segment $1 \in \mathcal{B}_0$, as well as the sink-state Ω corresponding to the segment 0. Then, for each rank $i = 1, 2, \ldots, \ell$, we consider every segment $s \in \mathcal{B}_i$ as a state in rank i. As stated in Lemma 4.4, every segment s corresponds to the

bitmap of the quotient of the language L by some word w. The transitions from s are then given by the segmentation of s into $s_1 s_2 \cdots s_k$, where $|s_j| = k^{i-1}$, for every $j \in [1, k]$. More precisely, we set $\delta(s, \sigma_j) = s_j$, for $j \in [1, k]$. Note that, if the language L is not empty, this construction creates exactly one initial and one final state, since $|\mathcal{B}_0| = |\mathcal{B}_\ell| = 1$.

```
1  def toMinDFA(B, Σ, ℓ):
2      k ← |Σ|
3      Q ← {1, 0}
4      δ ← {(0 --σ--> 0)}_{σ∈Σ}
5      for i ∈ [1, ℓ] do
6          B_i ← segment(B, k^i)
7          Q ← Q ∪ B_i
8          for s ∈ B_i do
9              S ← segment(s, k)
10             for (σ_j, s_j) ∈ Σ × S do
11                 δ ← δ ∪ (s --σ_j--> s_j)
12     return ⟨Q, Σ, δ, B, 1⟩
```

Algorithm 4.1: Construction of the minimal DFA for a block language L from its bitmap representation B.

Consider Algorithm 4.1, which implements the above construction. Given a bitmap B for a language over a k-letter alphabet Σ and block length ℓ, returns the minimal DFA for $\mathcal{L}(B)$. First, it initialises the set of states Q with the final state 1 and the sink-state 0, and the transition function δ with a loop on the sink-state by every symbol on the alphabet. Then, for every rank $i = 1, 2, \ldots, \ell$, the procedure partitions the bitmap B into segments of length k^i (using the function $segment$) retrieving the set $\mathcal{B}_i$ and adding it to set of states Q. For each element $s \in \mathcal{B}_i$, it decomposes it again in k segments, and adds the transitions to the δ function as previously described. At the end of the for loop, the algorithm returns the automaton $\langle Q, \Sigma, \delta, B, 1\rangle$ with initial state B and final state 1.

The following lemma formalises the correctness of the construction, that is, the mapping from bitmaps to DFAs preserves the language.

Lemma 4.8. *Let* $L \subseteq \Sigma^\ell$ *be a block language with bitmap* B*, where* $\ell \geqslant 0$*. Then, the DFA* $\mathcal{A}$ *obtained by applying Algorithm 4.1 to* B *recognises* L*, that is,* $\mathcal{L}(\mathcal{A}) = L$*.*

Proof. Let us show that both $\mathcal{L}(\mathcal{A}) \subseteq L$ and $L \subseteq \mathcal{L}(\mathcal{A})$.

- $\mathcal{L}(\mathcal{A}) \subseteq L$: let $w \in \Sigma^\ell \setminus L$. By construction, each state of $\mathcal{A}$ is a segment of B which, by Lemma 4.4, is the bitmap of the quotient of L by some word. Since $w \notin L$, w can be split into two words $w = w_1 w_2$ such that $w_1^{-1} L = \emptyset$. As for every word x, $x^{-1}\emptyset = \emptyset$, we get

$w^{-1}L = (w_1w_2)^{-1}L = {w_2}^{-1}(w_1^{-1}L) = \emptyset$. The empty language corresponds to the bitmap associated with the sink-state, therefore the initial path with label w has the state 0 as its end. Thus, $w \notin \mathcal{L}(\mathcal{A})$.

- $L \subseteq \mathcal{L}(\mathcal{A})$: let $w \in L$. Then, $w^{-1}L = \{\varepsilon\}$, whose bitmap is 1. Therefore, the initial path with label w is final, since its end is 1. Thus, $w \in \mathcal{L}(\mathcal{A})$.

□

Moreover, this construction yields the minimal DFA which accepts the language represented by the bitmap, as stated in the following lemma.

Lemma 4.9. *Let $L \subseteq \Sigma^\ell$ be a block language with bitmap B, where $\ell \geqslant 0$ and $|\Sigma| = k$. Then, the DFA $\mathcal{A}$ obtained by applying Algorithm 4.1 to B is minimal.*

Proof. Recall the second part of Myhill-Nerode Theorem, introduced in Theorem 2.4. We must prove that the number of equivalence classes on the relation $\sim_L$ is equal to the size of Q, the set of states of $\mathcal{A}$ which, by construction, is given by $Q = \bigcup_{i=0}^{\ell} \mathcal{B}_i$. Let s_1, s_2 be two different states of $\mathcal{A}$. As $\mathcal{A}$ is deterministic, we have $\overleftarrow{\mathcal{L}}_{s_1}(\mathcal{A}) \cap \overleftarrow{\mathcal{L}}_{s_2}(\mathcal{A}) = \emptyset$. Now, let $w_1 \in \overleftarrow{\mathcal{L}}_{s_1}(\mathcal{A})$ and $w_2 \in \overleftarrow{\mathcal{L}}_{s_2}(\mathcal{A})$. It also follows that $w_1^{-1}L \neq w_2^{-1}L$ as $s_1 \neq s_2$, and s_1 and s_2 represent the quotients of L by the words w_1 and w_2, respectively, as of Lemma 4.4. Then, $w_1 \nsim_L w_2$. □

Combining the results of Lemmas 4.8 and 4.9, we obtain:

Theorem 4.10. *Let $L \subseteq \Sigma^\ell$ be a block language, for some $\ell \geqslant 0$. The construction of a DFA from the bitmap $B(L)$, detailed in Algorithm 4.1, yields the minimal DFA for L.*

Example 4.11. *Let $L \subseteq \{a, b\}^4$ be the language of Example 4.1 with bitmap representation $B = 1011011100011110$. The correspondent minimal DFA obtained by applying Algorithm 4.1 is depicted in Figure 4.1. The final state, found in rank 0, is the state 1, and the sink-state (0) is omitted, as well as all transitions from and to it. States in rank 1 correspond to 2-bit segments, in this case: 10, 11, and 01. In particular, we have $\delta(10, a) = \delta(01, b) = \delta(11, a) = \delta(11, b) = 1$. States in rank 2 correspond to $2^2 = 4$-bit segments: 1011, 0111, 0001, and 1110. And we have, for instance, $\delta(0111, a) = 01$ and $\delta(0111, b) = 11$. Similarly for ranks 3 and 4. The initial state corresponds to B.*

4.3 Minimal NFAs for Block Languages Described by Bitmaps

In this section, we will extend the use of bitmaps for block languages, by describing a construction of minimal NFAs from the bitmap representation. We will also present a proof on how, on

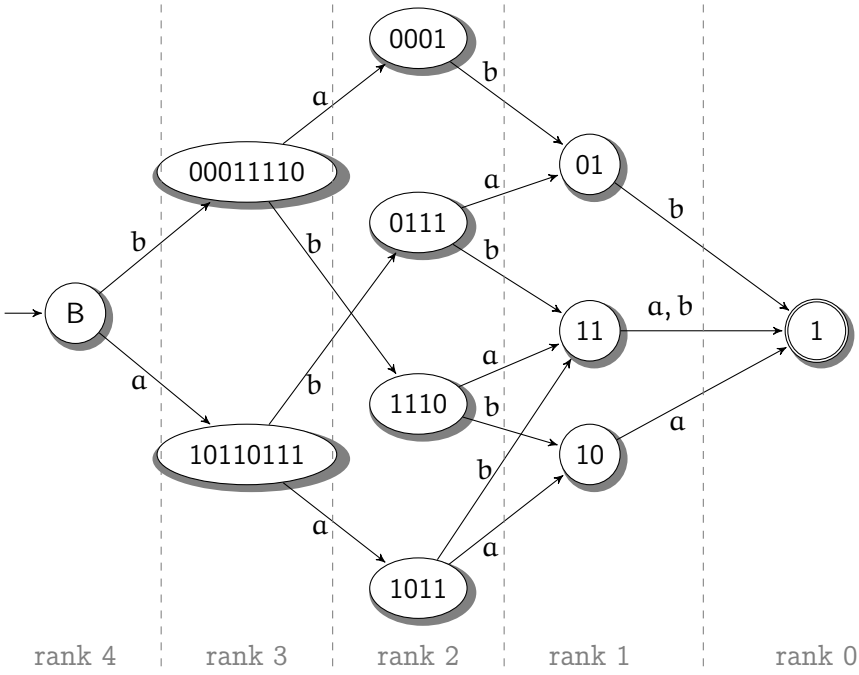

Figure 4.1: The minimal DFA accepting the language of Example 4.1.

this case, for this class of languages, the problem is NP-COMPLETE.

Given a bitmap B representing a block language $L \subseteq \Sigma^\ell$, for some block length $\ell \geqslant 0$ and $|\Sigma| = k$, one can build a minimal NFA similarly to the previous construction for minimal DFAs, by iteratively finding the minimal number of states required at each rank of the NFA. The main difference with the deterministic counterpart is that the quotients of the language can be represented by a set of states, instead of a single one.

First, let us define a *cover of a word*, in this context. Let $\mathcal{C}$ be a finite set of binary words of length n, that is, $\mathcal{C} \subseteq 2^{\{0,1\}^n}$, for some $n \in \mathbb{N}$. We say that $\mathcal{C}$ is a *cover for a word* $s \in \{0,1\}^n$ (or, alternatively, s *is covered* by $\mathcal{C}$) if there is a subset of words in $\mathcal{C}$ such that the bitwise disjunction of those words equal s. Formally,

$$\mathcal{C} \text{ covers } s \Leftrightarrow (\exists m \in [1, |\mathcal{C}|])(\exists \{c_1, \ldots, c_m\} \subseteq \mathcal{C}) : \bigvee_{i=1}^{m} c_i = s.$$

Most importantly, we extend this definition to sets in the natural way, that is, $\mathcal{C}$ is a *cover of a finite set* of binary words $\mathcal{B}$ if every word in $\mathcal{B}$ is covered by $\mathcal{C}$. Moreover, we say that $\mathcal{C}$ is a *minimal cover* for $\mathcal{B}$ if there is no other set $\mathcal{C}'$ smaller than $\mathcal{C}$ that covers $\mathcal{B}$.

Example 4.12. *Let* $\mathcal{C} = \{1100, 1010, 0001\}$ *and* $\mathcal{B} = \{1100, 1110, 1101, 1111\}$. *Then,* $\mathcal{C}$ *covers* $\mathcal{B}$ *because* $\mathcal{C}$ *covers every word from* $\mathcal{B}$*:*

- $1100 = 1100$;

- $1110 = 1100 \vee 1010$;
- $1101 = 1100 \vee 0001$;
- $1111 = 1100 \vee 1010 \vee 0001$.

One can observe that the set $\mathcal{C}$ is a minimal cover for $\mathcal{B}$, but not unique since $\mathcal{C}' = \{1100, 0010, 0001\}$ also covers $\mathcal{B}$ and $|\mathcal{C}| = |\mathcal{C}'|$.

Let us now show how to obtain an NFA for a non-empty block language L with block length $\ell \geqslant 0$, over a k-letter alphabet Σ, and with bitmap representation B. The construction starts as before, where the final state 1 is added at rank 0. Additionally, we define the function $\rho : \{0,1\}^\star \to 2^{\{0,1\}^\star}$ that maps segments into covers, that is, $\rho(s) = \{c_1, c_2, \ldots, c_m\}$ if $|s| = |c_i|$ and $\bigvee_{i=1}^{m} c_i = s$. Initially, we set $\rho(1) = \{1\}$. Then, for each rank $i = 1, 2, \ldots, \ell$, we look for the minimal set $\mathcal{C}_i$ that cover the set $\mathcal{B}_i$, the collection of segments of B with length k^i with at least one bit set to 1. The states at rank i will be the segments found in the cover $\mathcal{C}_i$. Subsequently, for each segment $s \in \mathcal{B}_i$ we set $\rho(s) = \{c_1, c_2, \ldots, c_m\} \subseteq \mathcal{C}_i$, such that $\rho(s)$ is the cover of s in $\mathcal{C}_i$. The transitions from rank i to rank $i-1$ will then be determined in a similar way to the DFA construction. For each state $c \in \mathcal{C}_i$ in rank i, we segment c into $c_1 c_2 \cdots c_k$, where $|c_j| = k^{i-1}$, for every $j \in [1, k]$, and set $\delta(c, \sigma_j) = \rho(c_j)$, if only $c_j \neq 0^{k^{i-1}}$. We must also guarantee that ρ is defined for each c_j or, alternatively, that $c_j \in \mathcal{B}_{i-1}$. For that, we need to limit the search space of the cover $\mathcal{C}_i$, so that each word in the set is a composition of k words from $\mathcal{B}_{i-1}$ or $0^{k^{i-1}}$. Formally, $\mathcal{C}_i \subseteq (\mathcal{B}_{i-1} \cup 0^{k^{i-1}})^k \setminus 0^{k^i}$. Also, as we previously saw, $\mathcal{B}_\ell = \{B\}$, so the minimal cover for $\mathcal{B}_\ell$ is itself. This result implies that B will be the single initial state at rank 0.

Let us look at Algorithm 4.2, containing the implementation of the above construction. Given a bitmap B for a language over a k-letter alphabet Σ and block length ℓ, returns a minimal NFA for $\mathcal{L}(B)$. First, the algorithm initialises the set of states Q with the final state 1, and the transition function δ. Additionally, it also defines the map ρ as we previously described, and $prev$ as the search space for the cover of the current rank, initially set to $prev = \mathcal{B}_0 \cup \{0\} = \{1, 0\}$. Then, for every rank $i = 1, 2, \ldots, \ell$, the procedure partitions the bitmap B into segments of length k^i with the use of the previously described function $segment$, retrieving the set $\mathcal{B}_i$. As already anticipated, it finds the minimal cover $\mathcal{C}_i$ for $\mathcal{B}_i$ by only using words in $prev$ through the procedure $minCover$, which will be described next. In the algorithm we denote, for simplicity, $\mathcal{C}_i(s)$ as the cover of s in $\mathcal{C}_i$, for each $s \in \mathcal{B}_i$. Once a cover is found it first defines ρ as the cover of s in $\mathcal{C}_i$, and then, adds each element of the cover to the set of states Q. Finally, it adds to δ the transitions from rank i to $i-1$ as we described above, and updates $prev$ at the end of each iteration. The algorithm then returns the automaton $\langle Q, \Sigma, \delta, B, 1\rangle$, where the states B and 1 are, respectively, the initial and the final state,.

```
1  def toMinNFA(B, Σ, ℓ):
2      k ← |Σ|
3      Q ← {1}
4      δ ← ∅
5      ρ ← {1 : {1}}
6      prev ← {1, 0}
7      for i ∈ [1, ℓ] do
8          B_i ← segment(B, k^i)
9          C_i ← minCover(B_i, prev, k, i)
10         Q ← Q ∪ C_i
11         for s ∈ C_i do
12             ρ(s) ← C_i(s)
13         for c ∈ C_i do
14             C ← segment(c, k)
15             for (σ_j, c_j) ∈ Σ × C ∧ c_j ≠ 0^{k^{i-1}} do
16                 δ ← δ ∪ (c --σ_j--> c')_{c' ∈ ρ(c_j)}
17         prev ← B_i ∪ {0^{k^i}}
18     return ⟨Q, Σ, δ, B, 1⟩
```

Algorithm 4.2: Construction of a minimal NFA for a block language L from its bitmap representation B.

The function $minCover(\mathcal{B}_i, prev, k, i)$, that returns the minimal cover $\mathcal{C}_i$ for $\mathcal{B}_i$, can be implemented with the help of an SMT-solver [KS16]. Let $\mathcal{D}_i$ be the set that contains all words of length k^i, such that each word is formed by concatenating k words from $prev$, excluding 0^{k^i}. Also, let m denote the size of the minimal cover, that incrementally increases until a solution is found, initially set to 1. Then, $minCover$ consists in the following steps:

1. let x_j be the Boolean variable that indicates whether the j-th word of $\mathcal{D}_i$ is in $\mathcal{C}_i$ or not, for $j \in [1, |\mathcal{D}_i|]$;
2. let $y_{j,t}$ be the Boolean variable that indicates whether the j-th word of $\mathcal{D}_i$ is in the cover for the t-th word of $\mathcal{B}_i$, for $j \in [1, |\mathcal{D}_i|]$, and $t \in [1, |\mathcal{B}_i|]$;
3. ensure that there are m elements in the cover, that is, $\sum_{j=1}^{|\mathcal{D}_i|} x_j = m$;
4. if $y_{j,t}$ is 1 then x_j must also be 1, i.e., $y_{j,t} \rightarrow x_j$;
5. for each $s \in \mathcal{B}_i$, ensure that there must be a subset in $\mathcal{C}_i$ that covers s. That is, let $s_t \in \mathcal{B}_i$ be the t-th element of $\mathcal{B}_i$ and $c_j \in \mathcal{D}_i$ be the j-th element of $\mathcal{D}_i$. Then, for all $t \in [1, |\mathcal{B}_i|]$,

$$s_t = \bigvee_{j=1}^{|\mathcal{D}_i|} c, \text{ where } c = \begin{cases} c_j, & \text{if } y_{j,t} = 1; \\ 0^{k^i}, & \text{otherwise.} \end{cases}$$

Given these constraints, if a solution is found, then the cover $\mathcal{C}_i$ is given by the set of words in $\mathcal{D}_i$ with index j such that $x_j = 1$. If no solution is found, increment m and repeat.

Theorem 4.13. *Given a set of words $\mathcal{B}$, each with length $n \in \mathbb{N}$, the problem of finding the minimal cover $\mathcal{C}$ for $\mathcal{B}$ is* NP-COMPLETE.

Proof. The problem we aim to solve can be characterised as:

- *Instance*: Collection of segments $\mathcal{B}$ and a positive integer m.
- *Question*: Is there a collection of segments $\mathcal{C}$ of size m such that, for each $s \in \mathcal{B}$, there is a sub collection of $\mathcal{C}$ whose bitwise disjunction is exactly s?

By Lemma 4.4, each segment in the sets $\mathcal{B}$ and $\mathcal{C}$ represent a set of words of the same length. Moreover, the bitwise disjunction over segments corresponds to the union over sets of words. Therefore, this problem can be seen as: given a collection of sets $\mathcal{B}$, is there a collection of sets $\mathcal{C}$ of size m, such that, for each set $s \in \mathcal{B}$, there exists a subset of $\mathcal{C}$ whose union equals s? This problem corresponds exactly to the SET-BASIS problem, and Stockmeyer proved that the SET-BASIS problem is NP-COMPLETE by reduction to the VERTEX-COVER problem [Sto76]. □

The construction of NFAs from bitmaps described above preserves the language, as the following lemma states.

Lemma 4.14. *Let $L \subseteq \Sigma^\ell$ be a block language with bitmap B, where $\ell \geqslant 0$. Then, the NFA $\mathcal{A}$ obtained by applying Algorithm 4.2 to B recognises L, that is, $\mathcal{L}(\mathcal{A}) = L$.*

Proof. Let $\mathcal{A}'$ be the DFA given by the construction of the minimal DFA from a bitmap (Algorithm 4.1). In Lemma 4.8, we proved that $\mathcal{L}(\mathcal{A}') = L$. Now, let us prove that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$. Let $s \in \mathcal{B}_i$ be a state from $\mathcal{A}'$ in rank i, for some $i \in [\ell]$. By construction, $\rho(s) = \{c_1, \ldots, c_m\}$, where $\{c_j\}_{j \in [1,m]}$ are states in $\mathcal{A}$ that exactly cover s. As $\mathcal{L}(s) = \bigcup_{j=1}^{m} \mathcal{L}(c_j)$, one concludes that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$. □

Moreover, the NFA given by this construction is minimal, as follows in the next lemma.

Lemma 4.15. *Let $L \subseteq \Sigma^\ell$ be a block language with bitmap B, where $\ell \geqslant 0$. Then, the NFA $\mathcal{A}$ obtained by applying Algorithm 4.2 to B is minimal.*

Proof. Let $w \in \Sigma^{\ell - i}$, for some $i \in [\ell]$, and P be the set of states reachable from the initial state q_0 of $\mathcal{A}$ after consuming w, that is, $P = \delta(q_0, w)$. Let P_1, P_2 be two non-empty subsets

of P such that $P_1 \cap P_2 \neq \emptyset$, and let $s_1 = \bigvee_{q \in P_1} q$ and $s_2 = \bigvee_{q \in P_2} q$ correspond to the bitmaps of the right languages of the sets P_1 and P_2, respectively. Suppose that $s_1 = s_2$, so P_1 and P_2 cover the same bitmaps. Then, $\mathcal{C}_i \setminus P_1$ (alternatively, $\mathcal{C}_i \setminus P_2$) would also cover the set $\mathcal{B}_i$, hence $\mathcal{C}_i$ would not be minimal. Thus, $s_1 \neq s_2$. □

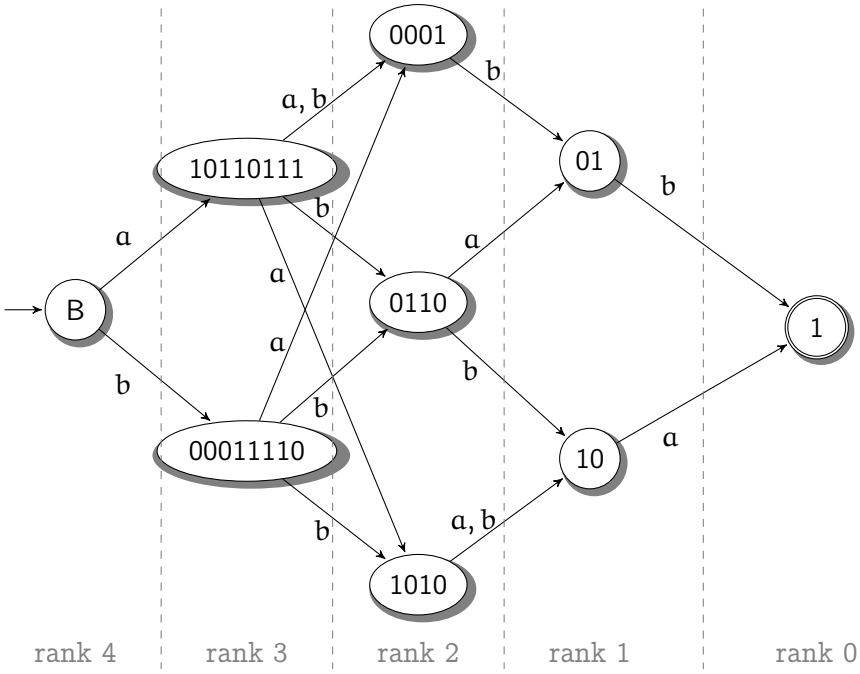

Figure 4.2: A minimal NFA accepting the language of Example 4.1.

Combining the results of Lemmas 4.14 and 4.15, we obtain:

Theorem 4.16. *Let* $L \subseteq \Sigma^\ell$ *be a block language, for some* $\ell \geqslant 0$*. The construction of a NFA from the bitmap* $B(L)$*, detailed in Algorithm 4.2, yields a minimal NFA for* L*.*

Example 4.17. *Let* $L \subseteq \{a, b\}^4$ *and* $B(L) = 1011011100011110$ *be the language of Example 4.1 and its bitmap representation, respectively. A correspondent minimal NFA obtained by applying the above construction is depicted in Figure 4.2. The final state, found in rank* 0*, is the state* 1*. In rank* 1*, we have* $\mathcal{B}_1 = \{01, 11, 10\}$ *and* $\mathcal{C}_1 = \{01, 10\}$ *is a minimal cover for* $\mathcal{B}_1$*, thus only two states are needed in this rank of the NFA. For rank* 2*,* $\mathcal{B}_2 = \{1011, 0111, 0001, 1110\}$ *and* $\mathcal{C}_2 = \{1010, 0110, 0001\}$*, because* $1011 = 0001 \vee 1010$*,* $0111 = 0110 \vee 0001$*, and* $1110 = 1010 \vee 0110$*. In rank* 3 *two states are needed and rank* 4 *has only the initial state.*

This page is intentionally left blank.

Chapter 5

Block Languages State Complexity

In this chapter, we will analyse the state complexity of block languages, namely, the maximal size of a minimal finite automaton for a block language, both for the deterministic and nondeterministic case (Section 5.1 and Section 5.2, respectively). In Section 5.3, we present results for the operational state complexity.

5.1 Maximal Size of Minimal DFAs

Câmpeanu and Ho [CH04] studied the maximal number of states that a minimal DFA may need to accept a block language over a k-letter alphabet and block size ℓ, with $k > 0$ and $\ell \geqslant 0$. They derived the following result:

Theorem 5.1 ([CH04], Corollary 10). *Let* $L \subseteq \Sigma^\ell$ *be a block language over a* k*-letter alphabet and block length* $\ell \geqslant 0$*. Then,*

$$sc(L) \leqslant \frac{k^{\ell-r+1}-1}{k-1} + \sum_{i=0}^{r-1}(2^{k^i}-1)+1,$$

where $r = \min\{\, i \in [\ell] \mid k^{\ell-i} \leqslant 2^{k^i}-1 \,\}$.

Proof. Let B be the bitmap representation of L, $\mathcal{A}$ be the minimal DFA for L and Q it set of states. As seen in Section 4.2, the width of rank $i \in [\ell]$ of $\mathcal{A}$ is given by the cardinality of the set $\mathcal{B}_i$. By Lemma 4.7, we know that $|\mathcal{B}_i| \leqslant \min(k^{\ell-i}, 2^{k^i}-1)$. A DFA is of maximal size if each rank has maximal width, that is, if the rank i has $\min(k^{\ell-i}, 2^{k^i}-1)$ states, for each

rank $i \in [\ell]$, and if it also contains a sink-state. Thus, we have

$$\begin{aligned} |Q| &\leqslant \sum_{i=r}^{\ell} k^{\ell-i} + \sum_{i=0}^{r-1} (2^{k^i} - 1) + 1 \\ &= \frac{k^{\ell-r+1} - 1}{k-1} + \sum_{i=0}^{r-1} (2^{k^i} - 1) + 1, \end{aligned}$$

as desired. □

In the next result we give an estimation of the value of r.

Theorem 5.2. *Let* $\ell > 0$, $k > 1$, *and* $r = \min\{\, i \in [\ell] \mid k^{\ell-i} \leqslant 2^{k^i} - 1 \,\}$. *Then,*

$$r = \lfloor \log_k \ell \rfloor + 1 + x, \text{ for some } x \in \{-1, 0, 1\}.$$

Proof. By definition of r, we have that both the following inequalities hold: $k^{\ell-r} \leqslant 2^{k^r} - 1$ and $k^{\ell-(r-1)} > 2^{k^{r-1}} - 1$. From the first inequality we obtain:

$$\begin{aligned} k^{\ell-r} \leqslant 2^{k^r} - 1 &\implies k^{\ell-r} < 2^{k^r} \\ &\implies (\ell - r) \log_2 k < k^r \\ &\implies \log_k(\ell - r) + \log_k(\log_2 k) < r \\ &\implies \log_k(\ell - r) < r \\ &\implies \log_k(\ell(1 - \frac{r}{\ell})) < r \\ &\implies \log_k \ell + \log_k(1 - \frac{r}{\ell}) < r \implies \log_k \ell < r. \end{aligned}$$

While, from the second inequality, we get:

$$\begin{aligned} k^{\ell-r+1} > 2^{k^{r-1}} - 1 &\implies k^{\ell-r+1} \geqslant 2^{k^{r-1}} \\ &\implies (\ell - r + 1) \cdot \log_2 k \geqslant k^{r-1} \\ &\implies \log_k(\ell - r + 1) + \log_k(\log_2 k) \geqslant r - 1 \\ &\implies \log_k(\ell - r + 1) > r - 2 \\ &\implies \log_k \ell + \log_k(1 + \frac{1 - r}{\ell}) > r - 2 \implies \log_k \ell > r - 2. \end{aligned}$$

Moreover, r is a natural number, so it is not hard to see that r can be written as the floor of $\log_k \ell$ plus a small constant, as claimed. □

We now present a family of *witness languages* whose minimal DFAs that recognise them are of maximal size, according to the bounds given in Theorem 5.1. The bitmap representation of these languages correspond to sequences of binary representations of the first positive integers, as we will see in Lemma 5.4.

We denote the binary representation of a positive integer i by $i_{[2]}$. To access the t-th rightmost symbol of a word s we will use the notation $\mathsf{last}(s,t)$, that is, let $s = s_1 s_2 \cdots s_n$ be a word of length $n > 0$ and $t \in [1,n]$. Then, $\mathsf{last}(s,t) = s_{n-t+1}$. Also, recall that given a block language $L \subseteq \Sigma^\ell$ and a word $w \in L$, we denote by $\mathsf{ind}(w)$ the index of w in the lexicographical ordered list of the words of $\Sigma^{|w|}$.

Let $\ell > 0$, and $r = \min\{\, i \in [\ell] \mid 2^{\ell-i} \leqslant 2^{2^i} - 1 \,\}$ as in Theorem 5.1 but with a fixed $k = 2$. In a minimal DFA with maximal size, the rank having the largest width is either r or $r-1$, depending on whether $2^{\ell-r} > 2^{2^{r-1}} - 1$ holds or not, respectively. Let r_ℓ be that rank and $t_\ell = \max(2^{\ell-r}, 2^{2^{r-1}} - 1)$ its width, i.e., $\mathsf{width}(r_\ell) = t_\ell$. Then, we consider the following family of languages:

$$\begin{aligned} MAX_\ell = \{\, w_1 w_2 \mid w_1 \in \Sigma^{\ell - r_\ell},\ w_2 \in \Sigma^{r_\ell},\\ i = \mathsf{ind}(w_1),\ j = \mathsf{ind}(w_2),\ \mathsf{last}(i_{[2]}, j) = 1 \,\}, \end{aligned}$$

defined over $\Sigma = \{a, b\}$. Informally, these languages contain words of size ℓ that can be split in w_1 of size $\ell - r_\ell$ and w_2 of size r_ℓ, with corresponding indices $i = \mathsf{ind}(w_1)$ and $j = \mathsf{ind}(w_2)$, such that the j-th rightmost symbol of $i_{[2]}$ is 1.

Example 5.3. *The maximal widths for each rank of a minimal DFA for a language with words of length $\ell = 5$ and $k = 2$ is $\{1, 3, 8, 4, 2, 1\}$, according to the bounds studied in Theorem 5.1. Then, $r_\ell = 2$ and $t_\ell = \mathsf{width}(r_\ell) = 8$. For this configuration, we have*

$$\begin{aligned} MAX_5 = \{&aaaaa, aabab, abaaa, abaab, abbba, baaaa,\\ &baaba, babab, babba, bbaaa, bbaab, bbaba, bbbbb\}. \end{aligned}$$

For instance, let $w_1 = baa$ where $i = \mathsf{ind}(w_1) = 5$ and $i_{[2]} = 101$. For $j = 1$ and $j = 3$, we have that $\mathsf{last}(i_{[2]}, j) = 1$, which correspond to the words aa and ba, respectively. Thus, $baaaa, baaba \in MAX_5$.

Now, let $\mathsf{pad}(s,t)$ be the function that concatenates leading zeros to a binary word s until its length equals t. We have the following pattern on the bitmap of MAX_ℓ:

Lemma 5.4. *Let r, r_ℓ, and t_ℓ be defined as before for $\ell > 0$ and alphabet size $k = 2$. Let*

$$P_{t_\ell, r_\ell} = \prod_{i=1}^{t_\ell} \mathsf{pad}(i_{[2]}, 2^{r_\ell})^R.$$

Then, the bitmap of the language MAX_ℓ is given by

$$B(MAX_\ell) = \begin{cases} P_{t_\ell, r_\ell}, & \text{if } t_\ell = 2^{\ell - r};\\ P_{t_\ell, r_\ell} \cdot 0^{2^{r_\ell}}, & \text{if } t_\ell = 2^{2^{r-1}} - 1. \end{cases}$$

Proof. Let us show for either value of t_ℓ that the bitmap has length 2^ℓ, which is not clear for the case $t_\ell = 2^{2^{r-1}} - 1$, and that the bitmap represent the language MAX_ℓ, that is, each bit equals to 1 in $B(MAX_\ell)$ corresponds to a word in MAX_ℓ, and vice versa.

1. $t_\ell = 2^{\ell - r}$:
   In this case $r_\ell = r$. Also, $|B(MAX_\ell)| = 2^{\ell-r} \cdot 2^{r_\ell} = 2^\ell$, so the bitmap has the intended length. Now, let us prove that if a word w is in MAX_ℓ, then the corresponding bit on the bitmap is 1. Let $w_1 w_2 \in MAX_\ell$, such that $w_1 \in \Sigma^{\ell - r_\ell}$, $i = \mathsf{ind}(w_1)$, $w_2 \in \Sigma^{r_\ell}$, and $j = \mathsf{ind}(w_2)$. According to Lemma 4.4, the segment of $B(MAX_\ell)$ which corresponds to the bitmap of $w_1^{-1}MAX_\ell$ is $\mathsf{pad}(i_{[2]}, 2^{r_\ell})^R$. So let us prove that the j-th leftmost bit of $\mathsf{pad}(i_{[2]}, 2^{r_\ell})^R$ is 1, implying that w_2 belongs to the quotient of MAX_ℓ w.r.t to w_1 according to $B(MAX_\ell)$. In fact, the j-th *leftmost* bit of $\mathsf{pad}(i_{[2]}, 2^{r_\ell})^R$ is 1 if, and only, the j-th *rightmost* bit of $\mathsf{pad}(i_{[2]}, 2^{r_\ell})$ is 1. The latter condition is equivalent to check if $\mathsf{last}(i_{[2]}, j) = 1$, which holds, by definition of the language. To prove that each bit of the bitmap corresponds to a word in the language, one can follow the described steps in reverse.

2. $t_\ell = 2^{2^{r-1}} - 1$:
   Now, $r_\ell = r - 1$. Let us first prove that the size of $B(MAX_\ell)$ is 2^ℓ. From the value of t_ℓ, we have $t_\ell \geqslant 2^{\ell - r}$. This implies that:

$$\begin{aligned} t_\ell \geqslant 2^{\ell-r} &\implies 2^{2^{r-1}} - 1 \geqslant 2^{\ell-r} \\ &\implies 2^{2^{r-1}} \geqslant 2^{\ell-r} \\ &\implies 2^{r-1} > \ell - r. \end{aligned}$$

   From the definition of r, we have that $t_\ell \leqslant 2^{\ell-r+1}$, leading to:

$$\begin{aligned} t_\ell \leqslant 2^{\ell-r+1} &\implies 2^{2^{r-1}} - 1 \leqslant 2^{\ell-r+1} \\ &\implies 2^{2^{r-1}} \leqslant 2^{\ell-r+1} && 2^{2^{r-1}} > 1 \\ &\implies 2^{r-1} \leqslant \ell - r + 1. \end{aligned}$$

   Then, these two conditions imply that $2^{r-1} = \ell - r + 1$, so $B(|MAX_\ell|) = 2^{r_\ell} 2^{2^{r-1}} = 2^{r_\ell + \ell - r + 1} = 2^\ell$, as desired.
   In 1. we already ensure that the bits equal to 1 on the bitmap correspond to words which are in MAX_ℓ. Then, let us prove that the padding of 2^{r_ℓ} zeros concatenated at the end of the bitmap correspond to words that do not belong to MAX_ℓ. By Lemma 4.4, the last 2^{r_ℓ} bits of the bitmap correspond to the bitmap of $w_1^{-1}MAX_\ell$, such that $w_1 \in \Sigma^{\ell - r_\ell}$ and $i = \mathsf{ind}(w_1) = t_\ell + 1 = 2^{2^{r-1}}$. Thus, it suffices to prove that $w_1^{-1}MAX_\ell = \emptyset$. By the definition of the language, $w_1 w_2 \in MAX_\ell$ if, and only if, $\mathsf{last}(i_{[2]}, j) = 1$, for $w_2 \in \Sigma^{r_\ell}$ and $j = \mathsf{ind}(w_2)$. Since i is a power of 2, j must be at least $2^{r-1} + 1$ for the condition to hold. However, we have $\mathsf{ind}(w_2) \leqslant 2^{r-1}$, for all $w_2 \in \Sigma^{r_\ell}$. Therefore, $w_1^{-1}MAX_\ell = \emptyset$.

□

Example 5.5. *By Lemma 5.4, the bitmap of the language* MAX_5 *from Example 5.3, where* $t_\ell = 8$ *and* $r_\ell = 2$*, is*

$$B(MAX_5) = \prod_{i=1}^{8} \mathsf{pad}(i_{[2]}, 4)^R = 1000\ 0100\ 1100\ 0010\ 1010\ 0110\ 1110\ 0001.$$

We now prove that the minimal DFA for MAX_ℓ is of maximal size.

Lemma 5.6. *Let* r*,* r_ℓ*, and* t_ℓ *be defined as before for* $\ell > 0$ *and alphabet size* $k = 2$*. Then, the minimal DFA accepting the language* MAX_ℓ *has maximal size.*

Proof. Let B be the bitmap representation of MAX_ℓ, according to Lemma 5.4, and $\mathcal{A}$ be the minimal DFA for MAX_ℓ given by the construction in Algorithm 4.1. As we previously saw, the width of rank $i \in [\ell]$ of $\mathcal{A}$ is given by the cardinality of the set $\mathcal{B}_i$. Then, $\mathcal{A}$ is of maximal size if each rank has maximal width, that is, if the rank j has $k^{\ell-j}$ states, for $j \in [r, \ell]$, or $2^{k^j} - 1$ states, for $j \in [r-1]$. Let us consider both possible values of t_ℓ:

1. $t_\ell = 2^{\ell - r}$:

   (a) $(\forall j \in [r-1]) : |\mathcal{B}_j| = 2^{2^j} - 1$:
   In particular, $\mathcal{B}_r$ contains the binary representation of the consecution integers from 1 to t_ℓ with 2^r bits. For the ranks $j < r$, the set $\mathcal{B}_j$ will contain the binary representation of the same numbers but using 2^j bits, as the segments of this set have length 2^j. Then, we have
   $$\mathcal{B}_j = \{\, \mathsf{pad}(i_{[2]}, 2^j)^R \mid \forall i \in [1, t_\ell] \,\}.$$
   Clearly, the size of $\mathcal{B}_j$ is bounded by the number of positive integers which binary representation uses at most 2^j bits, so $|\mathcal{B}_j| \leqslant 2^{2^j} - 1$. On the other hand, $t_\ell \geqslant 2^{2^j} - 1$, by the definition of t_ℓ and r. Thus, the proposition holds.

   (b) $(\forall j \in [r, \ell]) : |\mathcal{B}_j| = 2^{\ell - j}$:
   From the observation in 1.(a) referring to the set $\mathcal{B}_r$, it is easy to see that its cardinality is $|\mathcal{B}_r| = 2^{\ell - r}$. Also, the set $\mathcal{B}_j$ is given by the segmentation of B into segments of length 2^j. Of course, $|\mathcal{B}_j| = 2^{\ell - j}$.

2. $t_\ell = 2^{2^{r-1}} - 1$:

   (a) $(\forall j \in [r-1]) : |\mathcal{B}_j| = 2^{2^j} - 1$:
   Analogous to the first case 1.(a).

(b) $(\forall j \in [r, \ell]) : |\mathcal{B}_j| = 2^{\ell - j}$:

As we saw on the previous case 1.(b), it suffices to show that $|\mathcal{B}_r| = 2^{\ell - r}$. Recall that, in this case, $r_\ell = r - 1$. By construction, B is composed by t_ℓ consecutive segments with at least one bit equals to 1 followed by a single segment of zeros, all of length 2^{r_ℓ}. Since m is odd, each element of $\mathcal{B}_r$, that has length 2^r, will be equal either to the binary representation of two consecutive numbers or the second number represented is zero. Then, $|\mathcal{B}_r| = 2^{2^{r_\ell} - 1}$ and, as mentioned in proof of Lemma 5.4 1.(b), $2^{r-1} = \ell - r + 1$ for this particular value of t_ℓ. Thus, $|\mathcal{B}_r| = 2^{2^{r-1} - 1} = 2^{\ell - r + 1 - 1} = 2^{\ell - r}$, as desired.

□

Example 5.7. *In Figure 5.1 it is depicted the minimal DFA for* MAX_5*, where* $t_\ell = 8$ *and* $r_\ell = 2$.

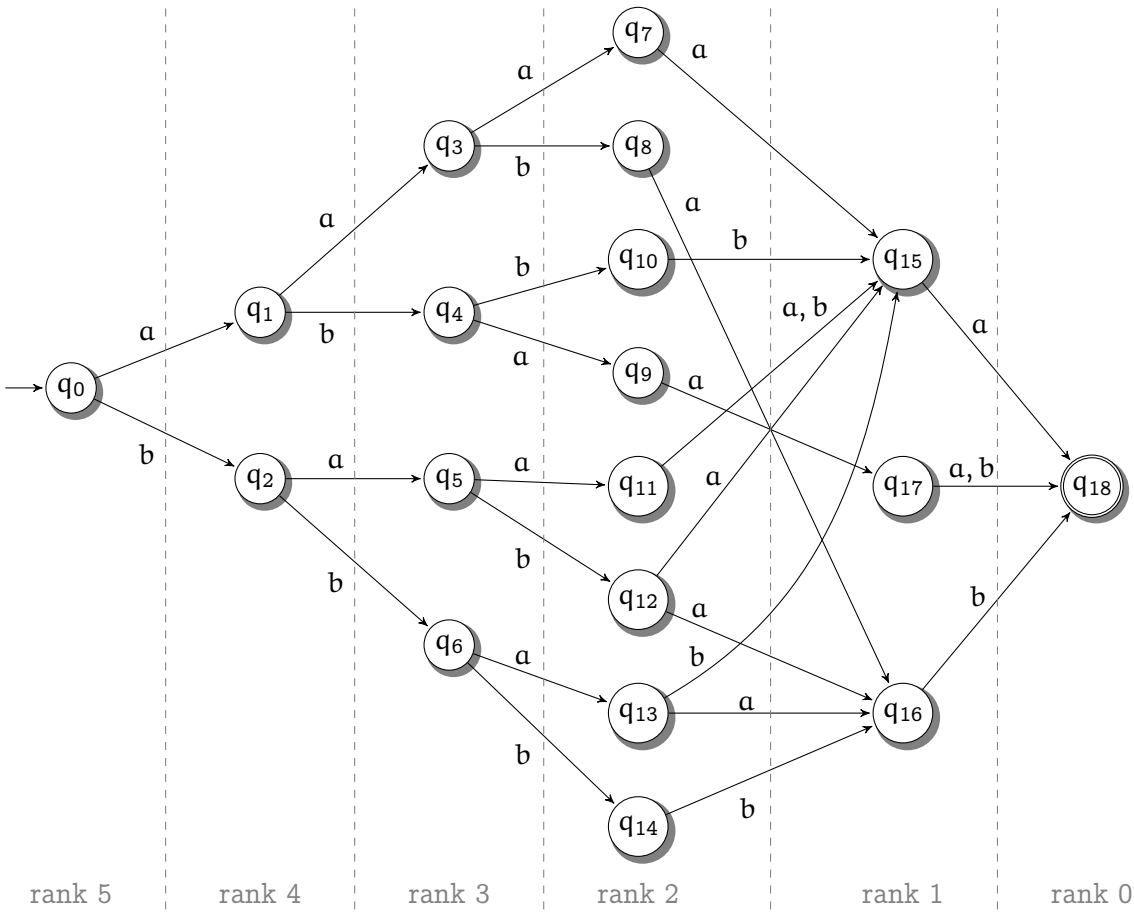

Figure 5.1: The minimal DFA accepting the language MAX_ℓ for $\ell = 5$. The sink-state is omitted, as well as all transitions from and to it.

5.2 Maximal Size of Minimal NFAs

Let $L \subseteq \Sigma^\ell$ be a block language of block length $\ell \geqslant 0$ and $k = |\Sigma|$, with bitmap representation B. Now, let $\mathcal{A}$ be a minimal NFA for L, given by the construction described in Section 4.3. The states on rank i of $\mathcal{A}$ are given by the set $\mathcal{C}_i$, a minimal cover for the set of segments of length k^i of B, namely $\mathcal{B}_i$, for $i \in [\ell]$. The size of the set $\mathcal{C}_i$ is bounded by the size of $\mathcal{B}_i$. Also, it can be noticed that k^i words suffice to cover any set $\mathcal{B}_i$. These bounds are formally stated in the following lemma:

Lemma 5.8. *Let $L \subseteq \Sigma^\ell$ be a block language of words of length $\ell \geqslant 0$ over an k-letter alphabet Σ, with a correspondent bitmap B. Then, for each $i \in [\ell]$, the cardinality of $\mathcal{C}_i$, a minimal cover for $\mathcal{B}_i$, is bounded by $|\mathcal{C}_i| = \min(k^{\ell-i}, k^i)$.*

Proof. The bound $|\mathcal{C}_i| \leqslant k^{\ell-i}$, for $r \in [\ell]$, refers to the maximal size of $\mathcal{B}_i$ (Lemma 4.7). It is easy to see that $\mathcal{B}_i$ covers itself, for all the sets $\mathcal{B}_i$, so the bound follows. On the other hand, we have $|\mathcal{C}_i| \leqslant k^i$ since the set $\mathcal{B}_r$ can be covered by the set of unit segments $\{u_i\}_{i \in [1, k^i]}$ of size k^i, where u_i represents the word of zeros, apart from the i-th position which is 1. □

We say that a minimal NFA is of maximal size if, for each $i \in [\ell]$, the width of rank i is $\min(k^{\ell-i}, k^i)$. As a result of the previous lemma, we are able to determine an upper bound for the nondeterministic state complexity of a block language.

Theorem 5.9. *Let $L \subseteq \Sigma^\ell$ be a block language over a k-letter alphabet and block length $\ell \geqslant 0$. Then,*

$$\mathsf{nsc}(L) \leqslant \begin{cases} 2 \cdot \dfrac{k^{\frac{\ell}{2}} - 1}{k-1} + k^{\frac{\ell}{2}}, & \text{if } \ell \text{ is even;} \\ \\ 2 \cdot \dfrac{k^{\lceil \frac{\ell}{2} \rceil} - 1}{k-1}, & \text{otherwise.} \end{cases}$$

Proof. If ℓ is even, there are an odd number of ranks, and the width of the minimal NFA with the maximal size is attained at the rank j that satisfies $k^{\ell-j} = k^j$, according to Lemma 5.8. This implies $j = \frac{\ell}{2}$, so we have:

$$\mathsf{nsc}(L) \leqslant 2 \cdot \sum_{i=0}^{\frac{\ell}{2}-1} k^i + k^{\frac{\ell}{2}} = 2 \cdot \frac{k^{\frac{\ell}{2}} - 1}{k-1} + k^{\frac{\ell}{2}}.$$

If ℓ is odd, there are an even number of ranks, and the width of the minimal NFA with maximal size is reached both in rank $\lfloor \frac{\ell}{2} \rfloor$ and $\lceil \frac{\ell}{2} \rceil$. Then, we have:

$$\mathsf{nsc}(L) \leqslant 2 \cdot \sum_{i=0}^{\lfloor \frac{\ell}{2} \rfloor} k^i = 2 \cdot \frac{k^{\lfloor \frac{\ell}{2} \rfloor + 1} - 1}{k-1} = 2 \cdot \frac{k^{\lceil \frac{\ell}{2} \rceil} - 1}{k-1},$$

as desired. □

We will now present a family of *witness languages* which minimal NFAs that recognise them are of maximal size. Let $k \geqslant 1$ and $d \geqslant 0$. Then, the languages

$$L_{k,d} = \{ ww^R \mid w \in \Sigma^d \},$$

defined over $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_k\}$, correspond to the set of palindromes of size $2d$.

Lemma 5.10. *Any minimal NFA that recognises the language* $L_{k,d}$*, for* $k > 0$ *and* $d \geqslant 0$*, is of maximal size.*

Proof. Let $\mathcal{A}$ be a minimal NFA for $L_{k,d}$, and m_i be the width of $\mathcal{A}$ at rank $i \in [2d]$. Let us prove that m_i is maximal, that is:

1. $(\forall i \in [d, 2d]) : m_i = k^{2d-i}$:
   Let $w_1, w_2 \in \Sigma^{2d-i}$ be two words that differ in at least one symbol. Let us prove that $w_1^{-1}L_{k,d} \cap w_2^{-1}L_{k,d} = \emptyset$. Let $w_3 \in \Sigma^i$ s.t. $w_1 w_3 \in L_{k,d}$. Then, w_3 can be decomposed into $uu^R w^R$, such that $|u| = \Sigma^{i-d}$. On the other hand, $w_2 w_3 \in L_{k,d}$ if, and only if, $w_1 = w_2$, which is a contradiction. Thus, $w_1^{-1}L_{k,d} \cap w_2^{-1}L_{k,d} = \emptyset$ and, as a consequence, k^{2d-i} states are needed at rank i, one for each word in Σ^{2d-i}.

2. $(\forall i \in [d]) : m_i = k^i$:
   Let us look at $\mathcal{A}^R$, the NFA for $L_{k,d}^R$ given by reversing every transition in $\mathcal{A}$ and swapping the initial with the final states. In fact, it is easy to see that $L_{k,d} = L_{k,d}^R$, hence $\mathcal{L}(A) = \mathcal{L}(\mathcal{A}^R)$. In 1., we proved that $m_j = k^{2d-j}$, for every rank $i \in [d, 2d]$. The i-th rank in $\mathcal{A}$ corresponds to the $(2d - i)$-th rank in $\mathcal{A}^R$, so that bound must be preserved.

□

By the arguments provided in the previous proof, we have the following corollary:

Corollary 5.11. *The minimal NFA for* $L_{k,d}$ *is deterministic.*

5.3 Operational State Complexity

In this section we consider operations on block languages defined by their bitmap representations and study both the deterministic and nondeterministic state complexity of those operations. As

introduced in Section 2.6, the operational state complexity is the state complexity, in the worst-case, of a language resulting from an operation, and is given as a function of the state complexity of the operands.

We will consider operations that are closed for block languages, i.e., the resulting language is also a block language, such as union and intersection of two block languages with the same block length, reversal, complement closed to the block, word addition and word removal. We will also analyse concatenation between two block languages whose words are of different lengths, as well as the Kleene star and the plus operations. The upper bounds of operational state complexity known for finite languages apply for block languages, which can be consulted in Table 3.1.

Additionally, we show how to build the bitmap of the language resulting by applying each operation and, to show that the provided bounds are tight, we also present a family of witness languages parameterised by the state complexity of the operands. In general, the witnesses provided may also be parametrised by other more natural parameters for block languages as, for example, the block length ℓ and the width of each rank of the automata.

5.3.1 Reversal

In the following, given a bitmap B of a block language $L \subseteq \Sigma^\ell$, $|\Sigma| = k$, and $\ell > 0$, we compute the bitmap for the reversal language L^R, namely B^R. The bitmap for L^R can be constructed by swapping the values of b_i and b_j from B, such that $i = \mathsf{ind}(w)$ and $j = \mathsf{ind}(w^R)$, for some $w \in \Sigma^\ell$. Trivial solutions require k^ℓ additional space to either store in a set the indexes already swapped or to construct a new bitmap. We now propose a routine that constructs the bitmap B^R in-place, that is, by performing operations on B and without requiring extra space.

The *perfect shuffle* of length 1 of two words $u = u_1u_2\cdots u_n$ and $v = v_1v_2\cdots v_n$ of same length n, denoted by $u \sqcup\!\sqcup_1 v$, is obtained by interleaving the letters of u and v, namely,

$$u \sqcup\!\sqcup_1 v = u_1v_1\cdots u_nv_n.$$

If $j \in \mathbb{N}$ is a divisor of n, the perfect shuffle of length j, denoted as $\sqcup\!\sqcup_j$, of u and v is the perfect shuffle of blocks of length j, that is,

$$u \sqcup\!\sqcup_j v = u_1\cdots u_jv_1\cdots v_j \ \cdots\ u_{n-j+1}\cdots u_nv_{n-j+1}\cdots v_n.$$

Moreover, this shuffle operation can be extended for any number $m \geqslant 2$ of words $w_1, w_2, \ldots, w_m$ of the same length n. We define $\sqcup\!\sqcup_j^m(w_1w_2\cdots w_m)$ as the perfect shuffle of blocks of length j taken from each of the w_i words. That is, with $w_i = w_{i,1}w_{i,2}\cdots w_{i,n}$, for each $i \in [1, m]$,

$$\sqcup\!\sqcup_j^m(w_1\cdots w_m) = w_{1,1}\cdots w_{1,j}\cdots w_{m,1}\cdots w_{m,j} \ \cdots\ w_{1,n-j+1}\cdots w_{1,n}\cdots w_{m,n-j+1}\cdots w_{m,n}.$$

Let us now show that the shuffle operation can be used to obtain the bitmap representation of the reversal of a language. Given a bitmap B of length k^ℓ, with $k, \ell > 0$, we define R_i, for each $i \in [\ell - 1]$, as follows:

$$R_i = \begin{cases} B, & \text{if } i = 0; \\ \sqcup_{k^{i-1}}^{k}(R_{i-1}), & \text{otherwise.} \end{cases}$$

Lemma 5.12. *Let $L \subseteq \Sigma^\ell$ be a block language, for some $\ell > 0$ and $|\Sigma| = k$, with bitmap representation B. The bitmap for the reversal of L, namely B^R, is given by*

$$R_{\ell-1} = \sqcup_{k^{\ell-2}}^{k}(\sqcup_{k^{\ell-3}}^{k}(\ldots(\sqcup_{1}^{k}(B))\ldots)).$$

Proof. Let us prove that $\mathcal{L}(R_{\ell-1}) = L^R$. For $i = 0$, of course $\mathcal{L}(R_0) = \mathcal{L}(B) = L$. Next, for $i = 1$ we have $R_1 = \sqcup_1^k(B)$. This operation performs the cyclic permutation $S_1 = (1\,2\cdots\ell)$ in each word of $\mathcal{L}(B)$, that is, each symbol of every word in $\mathcal{L}(B)$ is shifted one position to their right and the last symbol becomes the first. The following operation, $R_2 = \sqcup_k^k(R_1)$, performs the permutation $S_2 = (2\,3\cdots\ell)$ in every word of $\mathcal{L}(R_1)$. Analogously, in this transformation each symbol apart from the first of every word in $\mathcal{L}(R_1)$ is shifted one position to their right but the last symbol becomes the second. In general, the j-th shuffle performs the permutation $S_j = (j\,(j+1)\cdots\ell)$, for $j \in [1, \ell - 1]$. The composition of the transformations $S_1, S_2, \ldots, S_{\ell-1}$ ensure that $\mathcal{L}(R_{\ell-1}) = L^R$ [PLW83]. □

Example 5.13. *Let $\Sigma = \{a, b\}$ and $\ell = 3$. Let $B = b_1b_2b_3b_4b_5b_6b_7b_8$ be a bitmap for a block language L such that $b_1 = b_4 = b_5 = 1$, and the remaining bits are 0. We have*

$$\begin{aligned} R_0 &= b_1b_2b_3b_4b_5b_6b_7b_8 \quad \textit{and} \quad \mathcal{L}(R_0) = \{aaa, abb, baa\}, \\ R_1 &= b_1b_5b_2b_6b_3b_7b_4b_8 \quad \textit{and} \quad \mathcal{L}(R_1) = \{aaa, bab, aba\}, \\ R_2 &= b_1b_5b_3b_7b_2b_6b_4b_8 \quad \textit{and} \quad \mathcal{L}(R_2) = \{aaa, bba, aab\}, \end{aligned}$$

and $\mathcal{L}(R_2) = L^R$, as desired.

Now we turn to the analyse of the state complexity of this operation.

Lemma 5.14. *Given an m-state DFA for a block language L, $2^{O(\sqrt{m})}$ states are sufficient for a DFA accepting L^R.*

Proof. A DFA for L^R, with $\ell > 0$, can be given by reversing each transition of the DFA for L, swapping the initial and final states, and then determinising the resulting NFA. The cost of the determinisation of an m-state NFA for a block language is $2^{\Theta(\sqrt{m})}$ in terms of number of states [KO14], so the state complexity of the reversal must also be limited by this bound. □

In the following, we show that this bound is tight. Recall the language MAX_ℓ, defined in Section 5.1, over the alphabet $\Sigma = \{a, b\}$ and for $\ell > 0$. As before, let

1. $r = \min\{\, i \in [\ell] \mid 2^{\ell-i} \leqslant 2^{2^i} - 1 \,\}$, where $r = \lfloor \log_k \ell \rfloor + 1 + x$, for some $x \in \{-1, 0, 1\}$;
2. $t_\ell = \max(2^{\ell-r}, 2^{2^{r-1}} - 1)$;
3. $r_\ell = r$, if $t_\ell = 2^{\ell-r}$, or $r_\ell = r - 1$, if $t_\ell = 2^{2^{r-1}} - 1$.

We claim that the minimal DFA for the reversal of MAX_ℓ, namely MAX_ℓ^R, has its width bounded by $2^{r_\ell+1}$. That is, for every rank $i \in [\ell]$, $\mathsf{width}(i) \leqslant 2^{r_\ell+1}$.

Lemma 5.15. *A minimal DFA $\mathcal{A}$ such that* $\mathsf{width}(\mathcal{A}) \leqslant 2^{r_\ell+1}$ *is sufficient to accept the reversal of the language* MAX_ℓ*, for* $\ell > 0$.

Proof. Let $\mathcal{A}$ be a DFA such that the last $r_\ell + 1$ ranks have maximal width (according to Theorem 5.1), that is, $\mathsf{width}(i) = 2^{\ell-i}$, for each $i \in [\ell - r_\ell, \ell]$. In particular, we have that the width of the rank $\ell - r_\ell$ is 2^{r_ℓ}. We can order the states in this rank in such a way that q_j is the state whose left language is the reverse of the j-th word of Σ^{r_ℓ}, for each $j \in [1, 2^{r_\ell}]$. Formally, $\overleftarrow{\mathcal{L}}_{q_j}(\mathcal{A}) = \{w^R\}$ such that $|w| = r_\ell$ and $\mathsf{ind}(w^R) = j$. Moreover, let us define the right language of q_j as

$$\mathcal{L}_{q_j}(\mathcal{A}) = \{\, w \in \Sigma^{\ell-r_\ell} \mid i = \mathsf{ind}(w^R), \mathsf{last}(i_{[2]}, j) = 1 \,\}.$$

From the definition of the language, it is easy to see that $\mathcal{A}$ accepts MAX_ℓ^R. Consider $\mathcal{A}_{q_j}$ as the DFA $\mathcal{A}$ with initial state q_j, and let us show that $\mathsf{width}(\mathcal{A}_{q_j}) \leqslant 2$, for all $j \in [1, 2^{r_\ell}]$.

Let $j \in [1, 2^{r_\ell}]$, $r' \in [\ell - r_\ell - 1]$, and $\mathcal{B}_{r'}$ be the set of segments of size $2^{r'}$ of the bitmap of MAX_ℓ^R. Let $s \in \mathcal{B}_{r'}$ and $w_1 \in \Sigma^{\ell-r_\ell-r'}$ such that s represents the language $w_1^{-1}\mathcal{L}_{q_j}(\mathcal{A})$, according to Lemma 4.4. From the definition of the language, $w_1w_2 \in \mathcal{L}_{q_j}(\mathcal{A})$ if, and only if, $\mathsf{last}(i_{[2]}, j) = 1$, for $w_2 \in \Sigma^{r'}$ and $i = \mathsf{ind}((w_1w_2)^R) = \mathsf{ind}(w_2^R w_1^R)$. We will now show that the number of states on the rank r' of $\mathcal{A}_{q_j}$ is bounded by 2, by arguing that $|\mathcal{B}_{r'}| \leqslant 2$. Consider the following two cases:

1. $j < \ell - r_\ell - r'$:
   In this case, the membership of w_1w_2 in $\mathcal{L}_{q_j}(\mathcal{A})$ has already been decided by the choice of w_1. Then, it is sufficient to check if $\mathsf{last}(i'_{[2]}, j) = 1$, where $i' = \mathsf{ind}(w_1^R)$, since $|w_1^R| = \ell - r_\ell - r'$. We have that either $i'_{[2]}$ has its j-th but last symbol (bit) equal to 1, which implies that $s = 11\cdots1$, or has not, implying that $s = 00\cdots0$, so $s \notin \mathcal{B}_{r'}$. Therefore, $|\mathcal{B}_{r'}| = 1$.
2. $j \geqslant \ell - r_\ell - r'$:
   Now, $w_1w_2 \in \mathcal{L}_{q_j}(\mathcal{A})$ depends on the choices of w_1. If $w_1 \in \Sigma^{\ell-r_\ell-r'} \setminus \{b^{\ell-r_\ell-r'}\}$,

the binary representation of i', with $i' = \mathsf{ind}(w_1^R)$, requires at most $\ell - r_\ell - r'$ bits. Then, the j-th but last symbol of i, corresponds to the $(j - \ell + r_\ell + r')$-th symbol of w_2. Hence, $u^{-1}\mathcal{L}_{q_j}(\mathcal{A}) = v^{-1}\mathcal{L}_{q_j}(\mathcal{A})$, for all $u, v \in \Sigma^{\ell - r_\ell - r'} \setminus \{b^{\ell - r_\ell - r'}\}$. On the other hand, if $w_1 = b^{\ell - r_\ell - r'}$, it results on a different quotient, since $\ell - r_\ell - r' + 1$ bits are needed for the binary representation of $\mathsf{ind}(w_1)$. Therefore, $|\mathcal{B}_{r'}| = 2$.

This result implies that $\mathsf{width}(\mathcal{A}_{q_j}) = 2$. As a consequence, the width of the ranks $r' \in [\ell - r_\ell - 1]$ of $\mathcal{A}$ are bounded by $2^{r_\ell + 1}$, as desired. □

We now show that a DFA for MAX_ℓ has exponentially many states than a DFA for MAX_ℓ^R.

Lemma 5.16. *Let $\mathcal{A}_1$ be an m-state DFA for MAX_ℓ^R. Then, a DFA $\mathcal{A}_2$ for MAX_ℓ needs at least $2^{\Omega(\sqrt{m})}$ states.*

Proof. For this proof, assume that $2^{\ell - r} > 2^{2^{r-1}} - 1$ (i.e., $t_\ell = 2^{\ell - r}$), which implies that $r_\ell = r$. A similar proof follows, otherwise.

Let Q and P be the set of states of $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively. By Lemma 5.6, $\mathcal{A}_2$ is of maximal size and $\mathsf{width}(\mathcal{A}_2) = 2^{\ell - r} = t_\ell$. Therefore, the number of states of $\mathcal{A}_2$ is

$$
\begin{aligned}
|P| &= \sum_{i=0}^{r_\ell - 1} (2^{2^i} - 1) + \sum_{i=r_\ell}^{\ell} 2^{\ell - i} \\
&= \sum_{i=0}^{r_\ell - 1} 2^{2^i} - r_\ell + 2^{\ell - r_\ell + 1} - 1 \\
&\geqslant 2^{2^{r_\ell - 1}} - (r_\ell + 1) && 2^{\ell - r_\ell + 1} > 0 \\
&\geqslant 2^{2^{\log_2 \ell + 2} - 1} - (\log_2 \ell + 3) && 2^{2^{r_\ell - 1}} \gg r_\ell,\ r = r_\ell, \text{ and } r < \log_2 \ell + 2 \\
&\geqslant 2^{4\ell - 1} - (\log_2 \ell + 3) = 2^{\Omega(\ell)}.
\end{aligned}
$$

By Lemma 5.15, $\mathcal{A}_1$ has its width bounded by $2^{r_\ell + 1}$. Moreover, the width of each rank $i \in [r-1]$ of $\mathcal{A}_1$ is also bounded by $2^{2^i} - 1$, as we have seen in Theorem 5.1. Then, let $r' = \min\{\, i \in [r-1] \mid 2^{r_\ell + 1} \leqslant 2^{2^i} - 1 \,\}$. In particular, we have

$$
\begin{aligned}
2^{r_\ell + 1} > 2^{2^{r'-1}} - 1 &\implies 2^{r_\ell + 1} \geqslant 2^{2^{r'-1}} \\
&\implies r_\ell + 1 \geqslant 2^{r'-1} \\
&\implies \log_2 \ell + 3 \geqslant 2^{r'-1} && r = r_\ell \text{ and } r < \log_2 \ell + 2 \\
&\implies r' \leqslant \log_2(\log_2 \ell + 3) + 1.
\end{aligned}
$$

The value of r' tells us how many ranks in $\mathcal{A}_1$ can achieve the maximal width of $2^{r_\ell+1}$. Then, the number of states of $\mathcal{A}_1$ is bounded by

$$\begin{aligned}
|Q| &\leqslant \sum_{i=0}^{r'-1}(2^{2^i}-1) + 2^{r_\ell+1}(\ell - r_\ell - r') + \sum_{i=\ell-r_\ell}^{\ell} 2^{\ell-i} \\
&\leqslant 2^{2^{r'}} - r' + 2^{r_\ell+1}(\ell - r_\ell - r' + 1) - 1 \\
&\leqslant 2^{2^{\log_2(\log_2 \ell+3)+1}} + 2\cdot 2^{r_\ell}(\ell+1) && r' \leqslant \log_2(\log_2 \ell + 3) + 1 \\
&\leqslant 2^6 \cdot 2^{\log_2 \ell^2} + 2^3 \cdot 2^{\log_2 \ell}(\ell+1) && r = r_\ell \text{ and } r < \log_2 \ell + 2 \\
&\leqslant 2^6\ell^2 + 2^3(\ell^2+\ell) = O(\ell^2).
\end{aligned}$$

Thus, we have that $sc(MAX_\ell^R) = m = O(\ell^2)$ and $sc(MAX_\ell) = 2^{\Omega(\sqrt{m})}$, as desired. □

With the results in Lemmas 5.14 and 5.16, we have:

Theorem 5.17. *Let $L \subseteq \Sigma^\ell$, for some $\ell > 0$, such that $sc(L) = m$. Then, $sc(L^R) \leqslant 2^{O(\sqrt{m})}$, and the bound is tight.*

The NFA for the reversal of a language $L \subseteq \Sigma^\ell$ is given by reversing the transitions on the NFA for L and swapping the initial and final states. In fact, the nondeterministic state complexity of the reversal of a finite language coincides with the nondeterministic state complexity of the language, so no better result can be obtained for the block languages.

Theorem 5.18. *Let $L \subseteq \Sigma^\ell$, for some $\ell > 0$. Then, $nsc(L^R) = nsc(L)$.*

Proof. The construction above shows that $nsc(L^R) \leqslant nsc(L)$. The following family of languages shows that, in particular, the equality holds. Let $L_\ell = \{a^\ell\}$, with $\ell > 0$. Since $L_\ell = L_\ell^R$, $nsc(L_\ell) = nsc(L_\ell^R)$. □

5.3.2 Word Addition and Word Removal

Consider a language $L \subseteq \Sigma^\ell$, for some $\ell > 0$, over an alphabet of size k. The operations of adding or removing a word $w \in \Sigma^\ell$ from the language, $L \setminus \{w\}$ and $L \cup \{w\}$, respectively, correspond to the not operation on the $ind(w)$-th bit of B, the bitmap of L. From that observation, we can estimate the state complexity of these operations.

Theorem 5.19. *Let $L \subseteq \Sigma^\ell$ be a block language such that $sc(L) = m$. Let $L' = L \oplus \{w\}$, for $\oplus \in \{\setminus, \cup\}$ and $w \in \Sigma^\ell$. Then, $m - (\ell - 1) \leqslant sc(L') \leqslant m + (\ell - 1)$.*

Proof. Let B and B' be the bitmap representations of L and L', respectively. Let us assume that the operation $\oplus$ results in a different language. Then, the bitmaps B and B' differ exactly for one bit. Moreover, for every $i \in [\ell]$, there is exactly one $j \in [1, k^{\ell-i}]$ such that $s_j^i \neq t_j^i$, where s_j^i and t_j^i denote the j-th bitmap segment of size k^i of B and B', respectively. Also, recall $\mathcal{B}_i$ (resp. $\mathcal{B}'_i$), the set of segments of size k^i of B (resp. B'). Then, there are four possible cases:

1. $s_j^i \in \mathcal{B}'_i$ and $t_j^i \in \mathcal{B}_i$: the two sets have the same size;
2. $s_j^i \in \mathcal{B}'_i$ and $t_j^i \notin \mathcal{B}_i$: $\mathcal{B}'_i$ has one more element than $\mathcal{B}_i$;
3. $s_j^i \notin \mathcal{B}'_i$ and $t_j^i \in \mathcal{B}_i$: $\mathcal{B}_i$ has one more element than $\mathcal{B}'_i$;
4. $s_j^i \notin \mathcal{B}'_i$ and $t_j^i \notin \mathcal{B}_i$: the two sets have the same size.

Therefore, the difference on the number of states from a minimal DFA which accepts the language L' and the DFA which accepts L is bounded by $\ell-1$, which is the number of ranks neither initial nor final. □

These bounds also extend to the nondeterministic state complexity, as proved in the following result.

Theorem 5.20. *Let $L \subseteq \Sigma^\ell$ be a block language with, such that $\mathsf{nsc}(L) = n$. Let $L' = L \oplus \{w\}$, for some $\oplus \in \{\setminus, \cup\}$ and $w \in \Sigma^\ell$. Then, $n - (\ell - 1) \leqslant \mathsf{nsc}(L') \leqslant n + (\ell - 1)$.*

Proof. Consider the proof of Theorem 5.19 and its notation. If the case (2) verifies, that is, $|\mathcal{B}'_i| = |\mathcal{B}_i| + 1$, then the cover will require, at most, one more segment to cover the new set. Analogously, the size of the cover for $\mathcal{B}'_i$ can be smaller by one than the cover for $\mathcal{B}_i$, for the case (3). □

The upper bounds from Theorems 5.19 and 5.20 are reachable, as stated in the following theorem.

Theorem 5.21. *The bounds given in Theorems 5.19 and 5.20 are tight.*

Proof. Let $\Sigma = \{a, b\}$ and $\ell > 0$. For word removal, consider $L_\ell = (a + b)^\ell$ and let $w = a^\ell$. We have $\mathsf{sc}(L_\ell) = \mathsf{nsc}(L_\ell) + 1 = \ell + 2$, while $\mathsf{sc}(L_\ell \setminus \{w\}) = \mathsf{nsc}(L_\ell \setminus \{w\}) + 1 = 2\ell + 1$, as Figure 5.2 suggests. For word addition, consider $L'_\ell = \{a^\ell\}$ and let $w = b^\ell$. We have $\mathsf{sc}(L'_\ell) = \mathsf{nsc}(L'_\ell) + 1 = \ell + 2$, while $\mathsf{sc}(L'_\ell \cup \{w\}) = \mathsf{nsc}(L'_\ell \cup \{w\}) + 1 = 2\ell + 1$. □

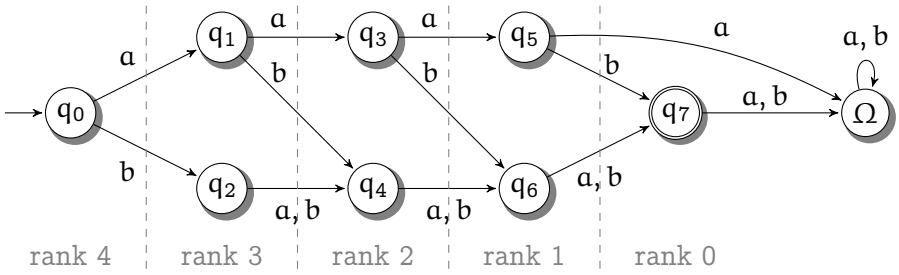

Figure 5.2: The minimal DFA for $L_\ell \setminus \{w\} = (a+b)^\ell \setminus a^\ell$, for $\ell = 4$.

5.3.3 Intersection

Let $L_1, L_2 \subseteq \Sigma^\ell$ be two block languages, for some $\ell > 0$ and $|\Sigma| = k$, and their respective bitmaps $B(L_1), B(L_2)$. The bitmap of the intersection of L_1 and L_2 is $B(L_1) \wedge B(L_2)$.

Let $\mathcal{A}_1 = \langle Q \cup \{\Omega_1\}, \Sigma, \delta_1, q_0, \{q_f\}\rangle$ and $\mathcal{A}_2 = \langle P \cup \{\Omega_2\}, \Sigma, \delta_2, p_0, \{p_f\}\rangle$ be the minimal DFAs for L_1 and L_2, respectively. As L_1 and L_2 are, in particular, both finite languages, a DFA $\mathcal{A}_3$ for $L_1 \cap L_2$ has at most $mn - 3(m+n) + 12$ states, where $|Q| = m-1$ and $|P| = n-1$, as shown in [HS08] (see Table 3.1). This bound is the result of:

1. The states (q_0, p), such that $p \in P \cup \{\Omega_2\}$ and $p \neq p_0$, and (q, p_0), such that $q \in Q \cup \{\Omega_1\}$ and $q \neq q_0$, are not reachable from the initial state (q_0, p_0), thus $m+n-2$ states can be saved;

2. The states (Ω_1, p), such that $p \in P \setminus \{p_0\}$, and (q, Ω_2), such that $q \in Q \setminus \{q_0\}$, can be merged with (Ω_1, Ω_2), thus $m+n-4$ states can be saved;

3. The states (q_f, p), such that $p \in P \setminus \{p_0, q_f\}$, and (q, q_f), such that $q \in Q \setminus \{q_0, q_f\}$, can also be merged with (Ω_1, Ω_2), as (q_f, p_f) is the only final state, thus $m+n-6$ states can be saved.

Thus, the number of remaining states is

$$mn - (m+n-2) - (m+n-4) + (mn-6) = mn - 3(m+n) + 12.$$

However, for block languages more states can be saved since a state (q, p) of $\mathcal{A}_3$ is both accessible and co-accessible if, and only if, $\mathsf{rank}(q) = \mathsf{rank}(p)$, for every $q \in Q$, $p \in P$. Let Q_i denote the set of states of rank i in $\mathcal{A}_1$, and $m_i = \mathsf{width}(i) = |Q_i|$. Analogously, let P_i denote the set of states of rank i in $\mathcal{A}_2$, and $n_i = \mathsf{width}(i) = |P_i|$. Also, we have $m = 1 + \sum_{i=0}^{\ell} m_i$ and $n = 1 + \sum_{i=0}^{\ell} n_i$, since the sink-states Ω_j do not belong to any rank, for $j \in \{1, 2\}$. We have that:

Lemma 5.22. *Given two minimal DFAs $\mathcal{A}_1$ and $\mathcal{A}_2$ for block languages L_1 and L_2, respectively, a DFA with*

$$\sum_{i=0}^{\ell} m_i n_i + 1$$

states is sufficient to recognise the intersection of L_1 and L_2, where m_i and n_i are the widths of rank i in $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively, for $i \in [\ell]$.

Proof. Given the above considerations, the set of states of the DFA resulting from trimming $\mathcal{A}_3$ are, in the worst-case, $\bigcup_{i=0}^{\ell} Q_i \times P_i$ and a single sink-state is needed. □

Let us show that this bound is tight for an alphabet of fixed size, as opposed to the general case of finite languages where a growing alphabet is required [HS08]. Consider the following family of languages, defined over an alphabet Σ of size $k \geqslant 2$, and let $d \geqslant 0$ and $x \in \{0, 1\}$:

$$L_{k,d,x} = \{\, a_1 a_2 \cdots a_{2d} \in \Sigma^{2d} \mid (\forall i \in [1, d]) : (i \equiv x \pmod 2) \implies a_i = a_{2d-i} \,\}.$$

Informally, it contains the words that can be split into two halves of size d, where, if $x = 0$ ($x = 1$, resp.), then the symbols in even (odd, resp.) positions of the first half are equal to their symmetric position in the second half.

Lemma 5.23. *Let $k \geqslant 2$, $d \geqslant 0$, and $x \in \{0, 1\}$. Also, let $\mathcal{A}$ be the minimal DFA for $L_{k,d,x}$ over a k-letter alphabet Σ and let m_i be the width of $\mathcal{A}$, for $i \in [2d]$. Then, for $i \in [d, 2d]$ we have:*

$$m_i = \begin{cases} k^{\lceil \frac{2d-i}{2} \rceil}, & \text{if } x = 0; \\ k^{\lfloor \frac{2d-i}{2} \rfloor}, & \text{if } x = 1, \end{cases}$$

and for $i \in [d]$ we have $m_i = m_{2d-1}$.

Proof. Let us prove for $x = 0$.

1. $(\forall i \in [d, 2d]) : m_i = k^{\lceil \frac{2d-i}{2} \rceil}$:
   Let $w_1, w_2 \in \Sigma^{2d-i}$ such that they differ at least in one even position. Now, let $w_3 = \sigma^{2(i-d)} w_1^R$, for some $\sigma \in \Sigma$. It is easy to see that $w_1 w_3 \in L_{k,d,0}$ but $w_2 w_3 \notin L_{k,d,0}$, so w_1 and w_2 have different quotients, and so they have to reach different states. Therefore, the number of states on rank i of $\mathcal{A}$ is given by $k^{\lceil \frac{2d-i}{2} \rceil}$, where the exponent is the number of odd integers between i and $2d$.

2. $(\forall i \in [d]) : m_i = m_{2d-i}$:
   It is easy to see that the proposition holds, as $L_{k,d,x} = L_{k,d,x}^R$, following a similar argument to the one provided in the proof for Lemma 5.10.

For $x = 1$, $k^{\lfloor \frac{2d-i}{2} \rfloor}$ is the number of even integers between i and $2d$, so the proof is similar. Refer to Figure 5.3 for support. □

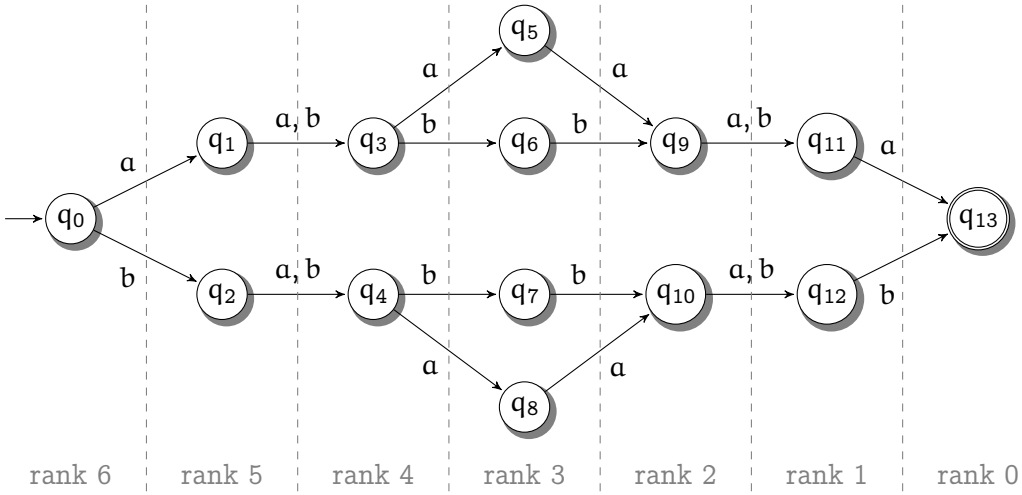

Figure 5.3: The minimal DFA for $L_{k,d,x}$ with $k = 2$, $d = 3$ and $x = 1$. The sink-state is omitted, as well as all transitions from and to it.

Lemma 5.24. *For some $d \geqslant 0$ and $k \geqslant 2$, let $\mathcal{A}_1$ and $\mathcal{A}_2$ be DFAs that accept $L_{k,d,0}$ and $L_{k,d,1}$, respectively. A DFA that recognises $L_{k,d,0} \cap L_{k,d,1}$ needs at least $\sum_{i=0}^{2d} m_i n_i + 1$ states, where m_i and n_i correspond to the widths of rank $i \in [2d]$ in $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively.*

Proof. It is easy to see that

$$L_{k,d,0} \cap L_{k,d,1} = \{ ww^R \mid w \in \Sigma^d \} = L_{k,d}$$

is the family of languages provided as witness for the maximal nondeterministic state complexity of a block language in Section 5.2. In particular, we concluded that a minimal NFA for $L_{k,d}$ must be of maximal size (Lemma 5.10), and that the NFA must be deterministic (Corollary 5.11). Let $\mathcal{A}_3$ be that DFA with set of states $S = S_0 \cup S_1 \cup \ldots \cup S_{2d}$, such that S_i corresponds to the set of states in rank $i \in [2d]$ of $\mathcal{A}_3$. As $\mathcal{A}_3$ is an NFA of maximal size, $|S_i| = |S_{2d-i}| = k^{2d-i}$, and, as seen in the proof of Lemma 5.23, we have $m_i = m_{2d-i} = k^{\lceil \frac{2d-i}{2} \rceil}$ and $n_i = n_{2d-i} = k^{\lfloor \frac{2d-i}{2} \rfloor}$, for $i \in [d, 2d]$. In fact,

$$|S_i| = m_i n_i = k^{\lceil \frac{2d-i}{2} \rceil} k^{\lfloor \frac{2d-i}{2} \rfloor} = k^{2d-i}.$$

□

From Lemmas 5.22 and 5.24 we have:

Theorem 5.25. *Given two block languages $L_1, L_2 \subseteq \Sigma^\ell$, for $\ell > 0$, with minimal DFAs $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively, we have*

$$\mathrm{sc}(L_1 \cap L_2) \leqslant \sum_{i=0}^{\ell} m_i n_i + 1,$$

and the bound is tight for alphabets of size at least 2, where m_i *and* n_i *are the widths of rank* i *in* $\mathcal{A}_1$ *and* $\mathcal{A}_2$*, respectively, for* $i \in [\ell]$.

For the nondeterministic state complexity, the bounds are the same as in the deterministic case except that the sink-state is not considered. In fact, the family of witnesses for the tightness of deterministic state complexity is also a witness for the nondeterministic one.

Theorem 5.26. *Let* $\mathcal{A}_1$ *be an* m*-state minimal NFA for a block language* $L_1 \subseteq \Sigma^\ell$*, and* $\mathcal{A}_2$ *be an* n*-state minimal NFA for* $L_2 \subseteq \Sigma^\ell$*, for some* $\ell \geqslant 0$*. Also, let* m_i *and* n_i *refer to the widths of rank* i *of* $\mathcal{A}_1$ *and* $\mathcal{A}_2$*, respectively. Then, an NFA with* $\sum_{i=0}^{\ell} m_i n_i$ *states is sufficient to recognise the intersection of* L_1 *and* L_2 *and the bound is tight for* $k > 1$.

Proof. That $\sum_{i=0}^{\ell} m_i n_i$ states are sufficient follows from the previous discussions. That are necessary results from considering again the languages $L_{k,d,x}$, for $k > 1$, $d > 0$ and $x \in \{0, 1\}$. Similar arguments to the ones used for Corollary 5.11 can be given to justify that the minimal NFAs for $L_{k,d,x}$ are, in fact, deterministic. Then, we have $\mathrm{sc}(L_{k,d,x}) - 1 = \mathrm{nsc}(L_{k,d,x})$. The rest of the proof follows similarly to the proof of Lemma 5.24. □

5.3.4 Union

Let $L_1, L_2 \subseteq \Sigma^\ell$ be two block languages, for some $\ell > 0$ and $|\Sigma| = k$, and their respective bitmaps $B(L_1), B(L_2)$. The bitmap of the union of L_1 and L_2 is $B(L_1) \vee B(L_2)$.

Let $\mathcal{A}_1 = \langle Q \cup \{\Omega_1\}, \Sigma, \delta_1, q_0, \{q_f\}\rangle$ and $\mathcal{A}_2 = \langle P \cup \{\Omega_2\}, \Sigma, \delta_2, p_0, \{p_f\}\rangle$ be the minimal DFAs for L_1 and L_2, respectively, with $|Q| = m - 1$ and $|P| = n - 1$. Again, as L_1 and L_2 are both finite languages, a DFA $\mathcal{A}_3$ for $L_1 \cup L_2$ has at most $mn - (m + n)$ states, as shown in [HS08] (see Table 3.1). This bound is the result of:

1. The states (q_0, p), such that $p \in P \cup \{\Omega_2\}$ and $p \neq p_0$, and (q, p_0), such that $q \in Q \cup \{\Omega_1\}$ and $q \neq q_0$, are not reachable from the initial state (q_0, p_0), thus $m + n - 2$ states can be saved;

2. The final states (q_f, Ω_2), (Ω_1, p_f), and (q_f, p_f) can be merged into a single final state, so 2 extra states can be saved.

However, again, one only needs to consider pairs of states (q, p) such that $\mathrm{rank}(q) = \mathrm{rank}(p)$, for $q \in Q, p \in P$. Let Q_i denote the set of states of rank i in $\mathcal{A}_1$, and $m_i = \mathrm{width}(i) = |Q_i|$. Analogously, let P_i denote the set of states of rank i in $\mathcal{A}_2$, and $n_i = \mathrm{width}(i) = |P_i|$. Also, we

have $m = 1 + \sum_{i=0}^{\ell} m_i$ and $n = 1 + \sum_{i=0}^{\ell} n_i$, since the sink-states Ω_j do not belong to any rank, for $j \in \{1, 2\}$. We have that:

Lemma 5.27. *Given two DFAs $\mathcal{A}_1$ and $\mathcal{A}_2$ for block languages L_1 and L_2, respectively, a DFA with*

$$\sum_{i=1}^{\ell-1} (m_i n_i + m_i + n_i) + 3$$

states is sufficient to recognise the union of L_1 and L_2, where m_i and n_i are the widths of rank i in $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively, for $i \in [1, \ell - 1]$.

Proof. Let $\mathcal{A}_3$ be the product automaton from $\mathcal{A}_1$ and $\mathcal{A}_2$. As mentioned above, the final states (q_f, Ω_2) and (Ω_1, p_f) can be merged with (q_f, p_f), and a state $(p, q) \in Q \times P$ is only accessible from the initial state if $\mathsf{rank}(p) = \mathsf{rank}(q)$. Therefore, the DFA resulting from trimming $\mathcal{A}_3$ has a single initial state, a final state and a sink-state, and also the states $(Q_i \times P_i) \cup (Q_i \times \{\Omega_2\}) \cup (\{\Omega_1\} \times P_i)$, at each rank $i \in [1, \ell - 1]$. Thus, the sufficient number of states follows. □

In fact, the bound is tight for an alphabet with size at least 3.

Lemma 5.28. *Given two DFAs $\mathcal{A}_1$ and $\mathcal{A}_1$ for block languages L_1 and L_2 over Σ^ℓ, respectively, a DFA with $\sum_{i=1}^{\ell-1}(m_i n_i + m_i + n_i) + 3$ states is necessary to recognise the union of L_1 and L_2, where m_i and n_i are the widths of rank i in $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively, for $i \in [1, \ell - 1]$ and $|\Sigma| > 2$.*

Proof. Since $\mathcal{A}_1$ and $\mathcal{A}_2$ are deterministic, $m_{\ell-1}$ and $n_{\ell-1}$, the width of rank $\ell - 1$ in $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively, are bounded by k and not equal to 0. Analogously, the width of the rank $\ell - 1$ of the DFA for the union of $\mathcal{L}(\mathcal{A}_1)$ and $\mathcal{L}(\mathcal{A}_2)$ is also at most k. When $k = 2$, it is easy to see that the inequality $0 < m_{\ell-1} n_{\ell-1} + m_{\ell-1} + n_{\ell-1} \leqslant k$ has no solutions.

Now, consider the languages $L_{1,\ell} = (a + c)^\ell$ and $L_{2,\ell} = (b + c)^\ell$, and let $\mathcal{A}_1$ and $\mathcal{A}_2$ be the minimal DFAs that recognise them, respectively, for some $\ell > 0$ and $\Sigma = \{a, b, c\}$. It is easy to see that $\mathsf{sc}(L_{1,\ell}) = \mathsf{sc}(L_{2,\ell}) = \ell + 2$ and $m_i = n_i = 1$, for every $i \in [\ell]$. The minimal DFA that recognises the language $L_{1,\ell} \cup L_{2,\ell}$ requires 3 states at each rank $i \in [1, \ell - 1]$: one state when an a has already been read, so the word is in $L_{1,\ell}$; one state when an b has already been read, so the word is in $L_{2,\ell}$; and one state for when only c's have been read (refer to Figure 5.4). Then,

$$\mathsf{sc}(L_{1,\ell} \cup L_{2,\ell}) = \sum_{i=1}^{\ell-1} (n_i m_i + n_i + m_i) + 3 = 3\ell.$$

□

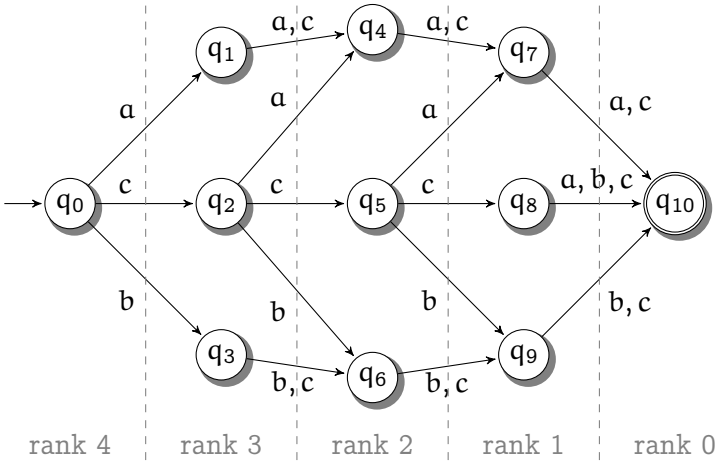

Figure 5.4: The minimal DFA for $L_{1,\ell} \cup L_{2,\ell} = (a+c)^\ell + (b+c)^\ell$, with $\ell = 4$. The sink-state is omitted as well as all transitions from and to it.

From Lemmas 5.27 and 5.28 we have:

Theorem 5.29. *Given two block languages $L_1, L_2 \subseteq \Sigma^\ell$, for $\ell > 0$, with minimal DFAs $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively, we have*

$$\mathsf{sc}(L_1 \cup L_2) \leqslant \sum_{i=1}^{\ell-1} (m_i n_i + m_i + n_i) + 3,$$

and the bound is tight for alphabets of size at least 3, where m_i and n_i are the widths of rank i in $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively, for $i \in [1, \ell-1]$.

For the nondeterministic state complexity, the upper bound is the same as for finite languages over the same alphabet size.

Theorem 5.30. *Let $L_1, L_2 \subseteq \Sigma^\ell$ with $\ell > 0$ and $|\Sigma| = k$, such that $\mathsf{nsc}(L_1) = m$ and $\mathsf{nsc}(L_2) = n$. Then, $\mathsf{nsc}(L_1 \cup L_2) \leqslant m + n - 2$, and this bound is reached.*

Proof. Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be minimal NFAs for L_1 and L_2 with m and n states, respectively. An NFA for $L_1 \cup L_2$ can be constructed by merging the initial states of $\mathcal{A}_1$ and $\mathcal{A}_2$, as well as the final states, so 2 states can be saved. In fact, this bound is tight, as the following languages suggest. Let $L_{1,\ell} = \{a^\ell\}$ and $L_{2,\ell} = \{b^\ell\}$, for some $\ell > 0$ and $\Sigma = \{a, b\}$. We have that $\mathsf{nsc}(L_{1,\ell}) = \mathsf{nsc}(L_{2,\ell}) = \ell + 1$ and $\mathsf{nsc}(L_{1,\ell} \cup L_{2,\ell}) = 2\ell$. □

5.3.5 Concatenation

Consider two languages $L_1 \subseteq \Sigma^{\ell_1}$ and $L_2 \subseteq \Sigma^{\ell_2}$, for some $\ell_1, \ell_2 > 0$ and $|\Sigma| = k$, with bitmaps $B(L_1)$ and $B(L_2)$, respectively. It is not difficult to see that replacing each 1 in $B(L_1)$

by $B(L_2)$ and each 0 by $0^{k^{\ell_2}}$ results in a bitmap for the language $B(L_1L_2)$. This ensures that each word of L_1L_2 is obtained by concatenating a word of L_1 with a word of L_2, and for each word obtained in such way the correspondent bit in $B(L_1L_2)$ is set to 1.

The deterministic state complexity of the concatenation for block languages coincide with the one for the finite languages when the first operand, L_1, has a single final state in its minimal DFA (see Table 3.1). Therefore, we have the following exact upper bound:

Theorem 5.31. *Let* $L_1 \subseteq \Sigma^{\ell_1}$ *and* $L_2 \subseteq \Sigma^{\ell_2}$*, for some* $\ell_1, \ell_2 > 0$*, be two block languages over a* k*-letter alphabet, where* $sc(L_1) = m$ *and* $sc(L_2) = n$*. Then,* $sc(L_1L_2) = m + n - 2$.

Proof. Let $B(L_1)$ and $B(L_2)$ be the bitmaps of these languages. Also, let $\mathcal{A}_1$ and $\mathcal{A}_2$ be the minimal DFAs for L_1 and L_2, respectively, and let $\mathcal{A}_3$ be the minimal DFA for L_1L_2. The width of the rank i of $\mathcal{A}_3$ is $|\mathcal{B}(L_2))_i|$, if $i \in [\ell_2]$, or is $|\mathcal{B}(L_1))_{i-\ell_2}|$, if $i \in [\ell_2 + 1, \ell_1 + \ell_2]$. Then, $\mathcal{A}_3$ saves 2 states by reusing the final state of $\mathcal{A}_1$ for the initial state of $\mathcal{A}_2$ (alternatively, reusing the initial state of $\mathcal{A}_2$ for the final state of $\mathcal{A}_1$) and also by eliminating one of the sink states. □

For the nondeterministic state complexity, the same result as the deterministic case is expected, which coincides with the state complexity for the finite languages.

Theorem 5.32. *Let* $L_1 \subseteq \Sigma^{\ell_1}$ *and* $L_2 \subseteq \Sigma^{\ell_2}$*, for some* $\ell_1, \ell_2 > 0$*, be two block languages over a* k*-letter alphabet, where* $nsc(L_1) = m$ *and* $nsc(L_2) = n$*. Then,* $nsc(L_1L_2) = m + n - 1$.

Proof. Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two minimal NFAs for two block languages L_1 and L_2. An NFA for L_1L_2 can be constructed by merging the final state of $\mathcal{A}_1$ with the initial state with $\mathcal{A}_2$, so a single state is saved. □

In fact, any two non-empty block languages L_1 and L_2 result in a family of witness languages, as this operation preserves the ranks of the automata of the operands, both on the deterministic and nondeterministic case.

Example 5.33. *Let* $L_{1,\ell_1} = \{a^{\ell_1}\}$ *and* $L_{2,\ell_2} = \{a^{\ell_2}\}$*, for* $\ell_1, \ell_2 > 0$*, over* $\Sigma = \{a\}$*. For the deterministic state complexity, we have* $sc(L_{1,\ell_1}) = \ell_1 + 2$*,* $sc(L_2) = \ell_2 + 2$*, and* $sc(L_{1,\ell_1}L_{2,\ell_2}) = \ell_1 + \ell_2 + 2$*. For the nondeterministic state complexity, we have* $nsc(L_{1,\ell_1}) = \ell_1 + 1$*,* $nsc(L_{2,\ell_2}) = \ell_2 + 1$*, and* $nsc(L_{1,\ell_1}L_{2,\ell_2}) = \ell_1 + \ell_2 + 1$.

5.3.6 Block Complement

Consider a language $L \subseteq \Sigma^{\ell}$, for some $\ell > 0$ and alphabet of size $k > 0$, and let B be its bitmap. We now consider the block complement language, namely $\Sigma^{\ell} \setminus L$, which we will denote by $\overline{L}^{\ell}$.

The bitmap of the language $\overline{L}^{\ell}$, namely $\overline{B}$, is given by complementing every bit of B.

Theorem 5.34. *Let* $L \subseteq \Sigma^{\ell}$*, with* $\ell > 0$*, be a block language with* $|\Sigma| = k$*, such that* $\mathrm{sc}(L) = m$*. Then,* $m - (\ell - 1) \leqslant \mathrm{sc}(\overline{L}^{\ell}) \leqslant m + (\ell - 1)$.

Proof. The number of states on a rank $i \in [\ell]$ of the minimal DFA for $\overline{L}^{\ell}$ is given by the cardinality of $\overline{\mathcal{B}}_i$, the set of the non-null segments of length k^i on the bitmap $\overline{B}$. If, for some $j \in [1, k^{\ell - i}]$, we have that $s_j^i = 00\cdots0$, which by definition implies that $s_j^i \notin \mathcal{B}_i$, then $\overline{s_j^i} = 11\cdots1$ and so $\overline{s_j^i} \in \overline{\mathcal{B}}_i$. On the other hand, if $s_j^i = 11\cdots1$ then $\overline{s_j^i} = 00\cdots0$, and so $s_j^i \in \mathcal{B}_i$ but $\overline{s_j^i} \notin \overline{\mathcal{B}}_i$. If s_j^i contains both zeros and ones, then both $s_j^i \in \mathcal{B}_i$ and $\overline{s_j^i} \in \overline{\mathcal{B}}_i$, so the operation does not alter the cardinality of the set. Therefore, $\big||\mathcal{B}_i| - |\overline{\mathcal{B}}_i|\big| \leqslant 1$, and so the upper bound follows.
As a family of languages that achieve the upper bound, consider $L_\ell = \{a^\ell\}$, for $\ell > 0$, over the alphabet $\{a, b\}$. It is easy to see that $\mathrm{sc}(L_\ell) = \ell + 2$. We have that $\overline{L}_\ell^{\ell} = (a + b)^\ell \setminus \{a^\ell\}$ and, as we saw in the proof of Theorem 5.21, $\mathrm{sc}(\overline{L}_\ell^{\ell}) = 2\ell + 1$. □

For the nondeterministic state complexity of the block complement operation, the bound meets the cost of the determinisation of NFAs for block languages. This bound is asymptotically tight for alphabets of size at least 2.

Lemma 5.35. *Let* $L \subseteq \Sigma^{\ell}$ *be a block language with* $|\Sigma| = k$*, such that* L *is accepted by an* m*-state NFA. Then,* $2^{O(\sqrt{m})}$ *states are sufficient for any NFA that recognises* $\overline{L}^{\ell}$.

Proof. Let $\mathcal{A}_1$ be an NFA for L with m states. The minimal DFA $\mathcal{A}_2$ for L will have at most $2^{O(\sqrt{m})}$ states [KO14]. Furthermore, the minimal DFA $\mathcal{A}_3$ for $\overline{L}^{\ell}$ will have at most $\ell + 1$ more states then $\mathcal{A}_2$, as shown in Theorem 5.34. As previously mentioned, the nondeterministic state complexity is bounded by the deterministic state complexity, that is, $\mathrm{nsc}(L) \leqslant \mathrm{sc}(L)$, so the sufficient number of states follows. □

Consider the following family presented by Karhumäki and Okhotin [KO14]:

$$L_{k,d} = \{\, a_1 a_2 \cdots a_{2d} \mid (\exists i \in [1, d]) : a_i = a_{i+d} \in \Sigma \setminus \{\sigma_k\} \}$$

defined over a k-ary alphabet $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_k\}$. Informally, this language contains words that can be split into two halves of size d, such that there is at least one position in the first half that matches its counterpart in the second one, and it is different than the *prohibited symbol* σ_k. The authors also showed the following result:

Proposition 5.36 ([KO14], Lemma 3)**.** *For each* $k \geqslant 2$ *and* $d \geqslant 2$*, the language* $L_{k,d}$ *is recognised by an NFA with* $(k-1)d^2 + 2d$ *states.*

Let us now focus on the block complement of these languages, in particular, in this following result:

Lemma 5.37. *For each* $k \geqslant 2$ *and* $d \geqslant 2$*, the language* $\overline{L}^{2d}_{k,d}$*, defined over a* k*-letter alphabet* Σ*, requires at least* k^d *states on the* d*-th rank.*

Proof. First, notice that $\overline{L}^{2d}_{k,d}$, the block complement of the language $\overline{L}_{k,d}$, can be formally defined as

$$\overline{L}^{2d}_{k,d} = \{\, a_1 a_2 \cdots a_{2d} \mid (\forall i \in [1, d]) : a_i \neq a_{i+d} \text{ or } a_i = \sigma_k \,\}.$$

Let u and v be two distinct words in Σ^d. It is easy to see that $u^{-1}\overline{L}^{2d}_{k,d} \neq v^{-1}\overline{L}^{2d}_{k,d}$. Moreover, there is at least a word $w \in u^{-1}\overline{L}^{2d}_{k,d}$ such that $w \notin v^{-1}\overline{L}^{2d}_{k,d}$, and vice versa, thus neither set is a subset of the other. As a consequence, one state in rank d is needed for each word in Σ^d. □

With these results, it is possible to determine that the nondeterministic state complexity for the block complement operation given in Lemma 5.35 is tight.

Theorem 5.38. *Let* $m \geqslant 2$ *and* Σ *an alphabet of size* $k \geqslant 2$*. Then, there exists a language* $L \subseteq \Sigma^\ell$*, for some* $\ell > 0$*, such that* $\mathsf{nsc}(L) = m$ *and* $\mathsf{nsc}(\overline{L}^\ell) = 2^{\Omega(\sqrt{m})}$.

Proof. Consider d as the largest integer for which $(k-1)d^2 + 2d \leqslant m$. As shown in [KO14], we have that

$$d = \left\lfloor \sqrt{\frac{m}{k-1} + \frac{1}{(k-1)^2}} - \frac{1}{k-1} \right\rfloor \geqslant \sqrt{\frac{m}{k-1}} - 2.$$

Then, $L_{k,d}$ is a language recognised by an-m-state NFA, while every NFA for $\overline{L}^\ell_{k,d}$ requires, by Lemma 5.37, at least

$$k^d \geqslant k^{\sqrt{\frac{m}{k-1}} - 2} = 2^{\Omega(\sqrt{m})}$$

states, as required. □

5.3.7 Kleene Star and Plus

Let $L \subseteq \Sigma^\ell$, for some $\ell \geqslant 0$, and B its bitmap representation. From B, one can obtain the minimal DFA for L, namely $\mathcal{A}_1 = \langle Q, \Sigma, \delta_1, q_0, \{q_f\}\rangle$. A DFA $\mathcal{A}_2 = \langle Q \setminus \{q_f\}, \Sigma, \delta_2, q_0, \{q_0\}\rangle$, recognises the language $L^\star$ if:

1. $\delta_2(q, \sigma) = q_0$, for all $q \in Q$ and $\sigma \in \Sigma$, such that $\delta_1(q, \sigma) = q_f$;

2. $\delta_2(q, \sigma) = \delta_1(q, \sigma)$, for all the remaining pairs $(q, \sigma) \in Q \times \Sigma$.

That is, the DFA for $L^\star$ is given by substituting all the transitions with final state as the target state to transitions to the initial state, and setting the initial state also as accepting. The same applies for a NFA for $L^\star$, as the following theorem states.

Theorem 5.39. *Let* $L \subseteq \Sigma^\ell$, *with* $\ell \geqslant 0$, *be a block language with* $\mathsf{sc}(L) = n$ *and* $\mathsf{nsc}(L) = m$. *Then,* $\mathsf{sc}(L^\star) = n - 1$ *and* $\mathsf{nsc}(L^\star) = m - 1$.

Continuing with the above setting, a DFA $\mathcal{A}_3 = \langle Q, \Sigma, \delta_3, q_0, \{q_f\}\rangle$ recognises the language L^+ if $\delta_3(q_f, \sigma) = \delta_1(q_0, \sigma)$, for all $\sigma \in \Sigma$, and if it behaves as δ_1 for the remaining states. Again, the NFA for L^+ is given by applying the same changes to the minimal NFA for L.

Theorem 5.40. *Let* $L \subseteq \Sigma^\ell$, *with* $\ell \geqslant 0$. *Then,* $\mathsf{sc}(L^+) = \mathsf{sc}(L)$ *and* $\mathsf{nsc}(L^+) = \mathsf{nsc}(L)$.

Chapter 6

Conclusions

Formal languages, defined by specific sets of words over a given alphabet, play an important role in various fields as in pattern recognition, programming languages, compiler design, software verification, and others. On the other hand, automata theory studies abstract machines for recognising and processing these languages. This thesis explores these concepts, focusing on sets of languages that share a uniform length, namely block languages, with applications in image processing and code theory. In particular, we propose the use of a new representation for these languages, that we call bitmaps, which turns out to be a good tool for the investigation of several properties of this class of language as, for example, the maximal size of the machines that represent them.

The initial part of the thesis consists of the study of the state of the art of the general case of finite languages, along with the existing results on the representation of block languages. Building from this knowledge, we present the representation of a block language as a binary word, such that each bit indicates whether the corresponding word, according to the lexicographical order, belongs, or not, to the language. Subsequently, we studied the conversion of these representations into machines that recognise the respective language, specifically deterministic finite automata (DFAs) and nondeterministic finite automata (NFAs). Apparently, the mapping of bitmaps to minimal DFAs is a straightforward task, as there is a direct correlation between bitmaps of a language and its quotients w.r.t. a word. Consequently, the Myhill-Nerode Theorem can be invoked to justify the minimality of the resulting DFAs. On the other hand, the construction of minimal NFAs from bitmaps is not trivial and was proven to be NP-COMPLETE. The problem is characterised by finding, for each rank of the NFA, the minimal set that covers the set of states of the same rank of the DFA. While representing languages by bitmaps does not directly reduce the complexity of this problem, it enable us to encode language operations as logic operations, thereby simplifying the implementation of our construction. The implementation of these algorithms can be found in the FAdo system [RM02].

Block Languages				
	sc	$\|\Sigma\|$	nsc	$\|\Sigma\|$
$L_1 \cup L_2$	$\sum_{i=1}^{\ell-1}(m_i n_i + m_i + n_i) + 3$	3	$m + n - 2$	2
$L_1 \cap L_2$	$\sum_{i=0}^{\ell} m_i n_i + 1$	2	$\sum_{i=0}^{\ell} m_i n_i$	2
$L_1 L_2$	$m + n - 2$	1	$m + n - 1$	1
$\Sigma^\ell \setminus L$	$m + \ell - 1$	2	$O(2^{\sqrt{m}})$	2
$L \cup \{w\}$	$m + \ell - 1$	2	$m + \ell - 1$	2
$L \setminus \{w\}$	$m + \ell - 1$	2	$m + \ell - 1$	2
L^*	$m - 1$	1	$m - 1$	1
L^+	m	1	m	1
L^R	$2^{\Theta(\sqrt{m})}$	2	m	1

Table 6.1: State complexity and nondeterministic state complexity for basic operations on block languages.

The second part of our work corresponds to the analysis of the state complexity of block languages. Initially, we establish bounds on the maximal size of minimal NFAs for block languages. The maximal size for minimal DFAs have already been studied before, however we extend this analysis by estimating one of the parameters within this bound. Then, we examine the operational state complexity of standard operations applied to block languages. Let L_1 be a block language over an alphabet of size $k > 0$, a block length of $\ell > 0$ and $\mathrm{sc}(L_1) = m$ ($\mathrm{nsc}(L_1) = m$). Also, we denote by m_i the width of rank i of the minimal DFA (a minimal NFA) for L_1. Analogously, let L_2 be a block language over the same alphabet and block size as L_1 and $\mathrm{sc}(L_2) = n$ ($\mathrm{nsc}(L_2) = n$). Moreover, we denote by n_i the width of rank i of the minimal DFA (a minimal NFA) for L_2. Table 6.1 summarises the obtained results, and one can compare these results with the ones for finite languages summarised in Table 3.1. For Boolean operations, the bounds are given using the rank widths and are smaller than the ones for finite languages. For the deterministic state complexity of concatenation and Kleene star, the bounds correspond to special cases of the ones for finite languages. For reversal, the results are analogous to the ones for finite languages, but here considering the bounds known for the determinisation of block languages. The remaining results for nondeterministic state complexity meet the values known for finite languages except for the specific operations for block languages (block complement, word addition, and word removal).

The main contributions of this thesis were documented in two submitted (and accepted) conference papers. The first focuses on the bitmap representation of block languages, the construction of minimal finite automata from bitmaps, and the maximal size of these machines. The second paper primarily addresses the operational state complexity of block languages.

6.1 Future Research Directions

As future work, we would like to extend bitmaps to the general case of finite languages. A finite language can be seen as a finite union of block languages. Therefore, the central focus of this research would be to determine how to obtain the minimal finite automaton for the union of two block languages with different block lengths by examining their bitmap representations.

Another possible line of work would be to express the bounds given for the deterministic state complexity of Boolean operations on block languages as a function of the number of states of the operands, as it is usually done. The bound that we propose is given as function of the rank widths of the automata, where one can easily conclude that it is smaller than the one given for finite languages, but it would be interesting to estimate this difference.

Moreover, it would be interesting to analyse the operation state complexity for the general complement operation on block languages. Unlike the one we studied, this operation results in an infinite language. Even though for the deterministic analysis it is trivial (turn every non-final state, including the sink state, into a final state in the original automaton, and convert the final state into a non-final state), for the nondeterministic case it seems to be a more difficult task.

This page is intentionally left blank.

References

[AMR08] Marco Almeida, Nelma Moreira, and Rogério Reis. Exact generation of minimal acyclic deterministic finite automata. 19(4):751–765, 2008.

[AMR12] Marco Almeida, Nelma Moreira, and Rogério Reis. Finite automata minimization algorithms. In Jiacun Wang, editor, *Handbook of Finite State Based Models and Applications*, pages 145–170. CRC Press, 2012.

[BJ63] J. A. Brzozowski and E. J. McCluskey Jr. Signal flow graph techniques for sequential circuit state diagrams. *IEEE Trans. on Electronic Computers*, EC-12(2):67–76, 1963.

[BK09] Janusz Brzozowski and Stavros Konstantinidis. State-complexity hierarchies of uniform languages of alphabet-size length. *Theoretical Computer Science*, 410(35):3223–3235, 2009. Descriptional Complexity of Formal Systems.

[Brz62] Janusz A. Brzozowski. Canonical regular expressions and minimal state graphs for definite events. 1962.

[BT13] Janusz A. Brzozowski and Hellis Tamm. Minimal nondeterministic finite automata and atoms of regular languages. *CoRR*, abs/1301.5585, 2013.

[CCSY01] Cezar Câmpeanu, Karel Culik II, Kai Salomaa, and Sheng Yu. State complexity of basic operations on finite languages. In Oliver Boldt and Helmut Jürgensen, editors, *4th WIA'99*, volume 2214, pages 60–70, 2001.

[CH04] Cezar Câmpeanu and Wing Hong Ho. The maximum state complexity for finite languages. 9(2-3):189–202, 2004.

[Cho56] N. Chomsky. Three models for the description of language. *IRE Transactions on Information Theory*, 2(3):113–124, 1956.

[DK12] Krystian Dudzinski and Stavros Konstantinidis. Formal descriptions of code properties: Decidability, complexity, implementation. *Int. J. Found. Comput. Sci.*, 23(1):67–85, 2012.

[EoWDoCS02] K. Ellul and University of Waterloo. Department of Computer Science. *Descriptional Complexity Measures of Regular Languages*. University of Waterloo, 2002.

[Glu61] V M Glushkov. The abstract theory of automata. *Russian Mathematical Surveys*, 16(5):1, oct 1961.

[GMRY17] 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.

[HK03a] Markus Holzer and Martin Kutrib. State complexity of basic operations on nondeterministic finite automata. In Jean-Marc Champarnaud and Denis Maurel, editors, *Implementation and Application of Automata*, pages 148–157, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[HK03b] Markus Holzer and Martin Kutrib. State complexity of basic operations on nondeterministic finite automata. In Jean-Marc Champarnaud and Denis Maurel, editors, *7th CIAA 2002*, volume 2608, pages 148–157, 2003.

[Hop71] John Hopcroft. An n log n algorithm for minimizing states in a finite automaton. In Zvi Kohavi and Azaria Paz, editors, *Theory of Machines and Computations*, pages 189–196. Academic Press, 1971.

[HS08] Yo-Sub Han and Kai Salomaa. State complexity of union and intersection of finite languages. *Int. J. Found. Comput. Sci.*, 19(3):581–595, 2008.

[HU79] John E. Hopcroft and Jeffrey D. Ullman. Introduction to automata theory, languages and computation. 1979.

[IY03] L. Ilie and S. Yu. Reducing NFAs by invariant equivalences. *Theoret. Comput. Sci.*, 306(1-3):373–390, 2003.

[Jir05] Galina Jiraskova. Jirásková, g.: State complexity of some operations on binary regular languages. theoret. comput. sci. 330, 287-298. *Theoretical Computer Science*, 330:287–298, 02 2005.

[JJS05] Jozef Jirásek, Galina Jirásková, and Alexander Szabari. State complexity of concatenation and complementation of regular languages. In Michael Domaratzki, Alexander Okhotin, Kai Salomaa, and Sheng Yu, editors, *Implementation and Application of Automata*, pages 178–189, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[KK21] Juhani Karhumäki and Jarkko Kari. Finite automata, image manipulation, and automatic real functions. In Jean-Éric Pin, editor, *Handbook of Automata Theory*, pages 1105–1143. European Mathematical Society, 2021.

[KL19] Bjørn Kjos-Hanssen and Lei Liu. The number of languages with maximum state complexity. In T. V. Gopal and Junzo Watada, editors, *15th TAMC*, volume 11436 of *LNCS*, pages 394–409. Springer, 2019.

[Kle56] S. C. Kleene. *Representation of Events in Nerve Nets and Finite Automata*, pages 3–42. Princeton University Press, Princeton, 1956.

[KMR18] Stavros Konstantinidis, Nelma Moreira, and Rogério Reis. Randomized generation of error control codes with automata and transducers. *RAIRO*, 52:169–184, 2018.

[KO14] Juhani Karhumäki and Alexander Okhotin. On the determinization blowup for finite automata recognizing equal-length languages. In R. Freivalds C. S. Calude and K. Iwama, editors, *Computing with New Resources - Essays Dedicated to Jozef Gruska*, volume 8808 of *LNCS*, pages 71–82. Springer, 2014.

[KPR03] Juhani Karhumäki, Wojciech Plandowski, and Wojciech Rytter. The complexity of compressing subsegments of images described by finite automata. *Discrete Applied Mathematics*, 125(2):235–254, 2003.

[KS16] Daniel Kroening and Ofer Strichman. *Decision Procedures:An Algorithmic Point of View*. Springer, 2016.

[MF71] Albert R. Meyer and Michael J. Fischer. Economy of description by automata, grammars, and formal systems. In *12th Annual Symposium on Switching and Automata Theory*, pages 188–191, Los Alamitos, 1971. IEEE.

[Moo56] Edward F. Moore. *Gedanken-Experiments on Sequential Machines*, pages 129–154. Princeton University Press, Princeton, 1956.

[Moo71] F.R. Moore. On the bounds for state-set size in the proofs of equivalence between deterministic, nondeterministic, and two-way finite automata. *IEEE Transactions on Computers*, C-20(10):1211–1214, 1971.

[MS72] A. R. Meyer and L. J. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *13th Annual Symposium on Switching and Automata Theory (swat 1972)*, pages 125–129, 1972.

[Ner58] A. Nerode. Linear automaton transformations. *Procedings of the American Mathematical Society*, pages 541–544, 1958.

[Pin21] Jean-Éric Pin, editor. *Handbook of Automata Theory*. European Mathematical Society Publishing House, Zürich, Switzerland, 2021.

[PLW83] Diaconis Persi, Graham R. L., and Kantor William.M. The mathematics of perfect shuffles. *Advances in Applied Mathematics*, 4:175–196, 1983.

[Rev92] Dominique Revuz. Minimisation of acyclic deterministic automata in linear time. *Theoret. Comput. Sci.*, 92(1):181–189, 1992.

[RM02] Rogério Reis and Nelma Moreira. Fado: tools for finite automata and regular expressions manipulation. 11 2002.

[RS59] M. O. Rabin and D. Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3(2):114–125, 1959.

[SDDR03] Priti Shankar, Amitava Dasgupta, Kaustubh Deshmukh, and B. Sundar Rajan. On viewing block codes as finite automata. *Theor. Comput. Sci.*, 290(3):1775–1797, 2003.

[SM73] L.J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time: Preliminary report. In *5th Annual ACM Symposium on Theory of Computing*, pages 1–9. ACM, 1973.

[Sto76] L. Stockmeyer. Set basis problem is NP-complete. Technical Report Report No. RC-5431, IBM Research Center, 1976.

[SY97] Kai Salomaa and Sheng Yu. NFA to DFA transformation for finite languages over arbitrary alphabets. 2(3):177–186, 1997.

[Tho68] Ken Thompson. Programming techniques: Regular expression search algorithm. *Commun. ACM*, 11(6):419–422, jun 1968.

[YZS94] Sheng Yu, Qingyu Zhuang, and Kai Salomaa. The state complexities of some basic operations on regular languages. *Theoretical Computer Science*, 125(2):315–328, 1994.