Block Languages and their Bitmap Representations

Guilherme Duarte[0000−0002−4119−0694]1, Nelma Moreira[0000−0003−0861−0105]1, Luca Prigioniero[0000−0001−7163−4965]2, and Rogério Reis[0000−0001−9668−0917]1⋆

[1] CMUP & DCC, Faculdade de Ciências da Universidade do Porto
Rua do Campo Alegre, 4169-007 Porto, Portugal
{guilherme.duarte,nelma.moreira,rogerio.reis}@fc.up.pt
[2] Department of Computer Science, Loughborough University, UK
L.Prigioniero@lboro.ac.uk

Abstract. In this paper we consider block languages, namely sets of words having the same length, and we propose a new representation for these languages. 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. Finally, we give an analysis of the maximum number of states sufficient to accept every block language in the deterministic and nondeterministic case.

1 Introduction

In the area of formal languages and automata theory, the class of regular languages is one of the most investigated. Classical recognizers for this class are finite automata, in both deterministic and nondeterministic variants. The capabilities 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 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

⋆ This work was partially supported by CMUP, member of LASI, which is financed by national funds through FCT – Fundação para a Ciência e a Tecnologia, I.P., under the projects with reference UIDB/00144/2020 and UIDP/00144/2020.

minimization of these machines [2]. The situation in the nondeterministic case is more challenging as 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 [17]. In this paper 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 [4,9] and image processing [6,7]. A typical problem in code theory is the construction of (maximal) block languages (codes) capable of detecting and correcting errors. Several properties of block codes using automata theory have also been studied, e.g. [15]. On the other hand, an image can be represented by a set of words of a same length (pixels). Then, 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 minimization of deterministic finite automata can be done in linear time in the case of finite (and hence also block) languages [13]. 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 [7], which is smaller than the general case, for which the cost in size is $2^{\Theta(n)}$ [11,14]. The maximum number of states of minimal deterministic finite automata for finite and block languages were studied by Câmpeanu and Ho [3], and Hanssen and Liu determined the number of block languages that attain the maximum state complexity [8]. Minimal deterministic finite automata for finite languages were enumerated by Almeida et al. [1]. Asymptotic estimates and exact formulae for the number of n-state minimal deterministic finite automata accepting finite languages over alphabets of size k were obtained by J. Priez [12] and by Price et al. [5].

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 symbol (or *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). We use this representation as a tool to investigate several properties of block languages. More precisely, in Sections 4 and 5 we show how to convert bitmaps into deterministic and nondeterministic finite automata, respectively. It is important to notice that the devices yielded by such 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. For the deterministic case, we also refine the analysis of the state complexity of block languages given by Câmpeanu and Ho [3], and we present a family of languages that witnesses the optimality of such costs (Section 4.1). On the other hand, for nondeterministic finite automata, we determine the sufficient number of states to accept every block language (Section 5.1).

2 Preliminaries

In this section we review some basic definitions about finite automata and languages and fix notation. Given an *alphabet* Σ, a *word* w is a sequence of symbols, and a *language* $L \subseteq \Sigma^\star$ is a set of words on Σ. The empty word is represented by ε. The *(left) quotient* of a language L by a word $w \in \Sigma^\star$ refers to the set $w^{-1}L = \{w' \in \Sigma^\star \mid ww' \in L\}$. The *reversal* of a word $w = \sigma_0\sigma_1\cdots\sigma_{n-1}$ is denoted as w^{R} and is obtained by reversing the order of the symbols of w, that is $w^{\mathrm{R}} = \sigma_{n-1}\sigma_{n-2}\cdots\sigma_0$. Given two integers i, j with $i < j$, let $[i, j]$ denote the range from i to j, including both i and j, namely $\{i, \ldots, j\}$. Moreover, we shall omit the left bound if it is equal to 0, thus $[j] = \{0, \ldots, j\}$.

A *nondeterministic finite automaton* (NFA) is a five-tuple $\mathcal{A} = \langle Q, \Sigma, \delta, I, F\rangle$ where Q is a finite set of states, Σ is a finite alphabet, $I \subseteq Q$ is the set of initial states, $F \subseteq Q$ is the set of final states, and $\delta : Q \times \Sigma \to 2^Q$ is the transition function. We consider the *size* of an NFA as its number of states. The transition function can be extended to words and sets of states in the natural way. When $I = \{q_0\}$, we use $I = q_0$. An NFA accepting a non-empty language is *trim* if every state is accessible from an initial state and every state leads to a final state. Given a state $q \in Q$, the *right language* of q is $\mathcal{L}_q(\mathcal{A}) = \{\, w \in \Sigma^\star \mid \delta(q, w) \cap F \neq \emptyset \,\}$, and the *left language* is $\overleftarrow{\mathcal{L}}_q(\mathcal{A}) = \{\, w \in \Sigma^\star \mid q \in \delta(I, w) \,\}$. The *language accepted* by $\mathcal{A}$ is $\mathcal{L}(\mathcal{A}) = \bigcup_{q\in I} \mathcal{L}_q(\mathcal{A})$. An NFA $\mathcal{A}$ is *minimal* if it has the smallest number of states among all NFAs that accept $\mathcal{L}(\mathcal{A})$. An NFA is *deterministic* (DFA) if $|I| = 1$ and $|\delta(q, \sigma)| \leq 1$, for all $(q, \sigma) \in Q \times \Sigma$. We can convert an NFA into an equivalent DFA by using the well-known subset construction. Two states q_1, q_2 of an automaton $\mathcal{A}$ are *equivalent* (or *indistinguishable*) if $\mathcal{L}_{q_1}(\mathcal{A}) = \mathcal{L}_{q_2}(\mathcal{A})$. If a DFA is *minimal* it has no equivalent states and it is unique up to renaming of states. If $\mathcal{A}$ is the minimal DFA for L, then, for each state q, $\mathcal{L}_q(\mathcal{A}) = w_q^{-1}L$ for some $w_q \in \Sigma^\star$, and if $q \neq q'$ then $w_q^{-1}L \neq w_{q'}^{-1}L$. The *state complexity* of a language L, denoted $sc(L)$, is the size of the minimal DFA accepting L. The *nondeterministitic state complexity* of a language L, denoted $nsc(L)$, is defined analogously.

A trim NFA $\mathcal{A} = \langle Q, \Sigma, \delta, I, F\rangle$ for a non-empty finite language, whose longest word is $\ell \geq 0$, is acyclic and ranked, i.e., the set of states Q can be partitioned into $Q_0 \cup Q_1 \cup \cdots \cup Q_\ell$ such that for every state q in rank Q_r, $\mathcal{A}$ reaches a final state by words of length at most r ($Q_r = \{q \in Q \mid \forall w \in \Sigma^\star, \delta(q, w) \in F \implies |w| \leq r\}$) and all transitions from states in rank Q_r lead only to states in rank Q_s, with $s < r$. We define the *width* of a rank Q_r as the cardinality of Q_r, the *width* of $\mathcal{A}$ to be the maximal width of all ranks. In the following, for the ease of notation, we shall denote the ranks by their indices, e.g., we refer to rank Q_r as rank r.

A DFA for a finite language is also ranked but it may have a *dead state* Ω, which is the only cyclic state, is usually omitted, and has no rank. Formally, $\mathcal{A} = \langle Q \cup \{\Omega\}, \Sigma, \delta, q_0, F\rangle$, with $F \subseteq Q$, $q_0 \neq \Omega$, and, for each symbol $\sigma \in \Sigma$, $\delta(\Omega, \sigma) = \Omega$. Moreover, for each nonempty word w and each state $q \in Q$, $\delta(q, w) \neq q$. In a trim acyclic automaton, two states q and q' are equivalent if they are both in the same rank, either final or not final, and their transition

functions lead to equivalent states, i.e., $\delta(q, w) \in F \iff \delta(q', w) \in F$, for each word $w \in \Sigma^\star$. An acyclic DFA can be minimized by merging equivalent states and the resulting algorithm runs in linear time in the size of the automaton (Revuz algorithm, [1,13]).

In the following we shall consider sequences of Boolean values, $\mathsf{B} \in \{0,1\}^n$ that we denote by *bitmaps*. Given two bitmaps $\mathsf{B}_1, \mathsf{B}_2 \in \{0,1\}^n$, $\mathsf{B}_1 \circ \mathsf{B}_2$ represents the bitmap obtained by carrying out the bitwise operation $\circ \in \{\vee, \wedge\}$, and $\overline{\mathsf{B}}_1$ the bitwise complement of B_1.

3 Block Languages and Bitmaps

Given an alphabet $\Sigma = \{\sigma_0, \ldots, \sigma_{k-1}\}$ of size $k > 0$ and an integer $\ell \geq 0$, a *block language* $L \subseteq \Sigma^\ell$ is a set of words of length ℓ over Σ. The language L can be characterized by a word in $\{0,1\}^{k^\ell}$ that we call *bitmap* and denote as

$$\mathsf{B}(L) = b_0 \cdots b_{k^\ell - 1},$$

where $b_i = 1$ if the word w is in L, and $i \in [k^\ell - 1]$ is the index of w in the lexicographical ordered list of all the words of Σ^ℓ. We will denote the bitmap of a language as B when it is unambiguous to which language the bitmap refers to. Moreover, each bitmap $\mathsf{B} \in \{0,1\}^{k^\ell}$ represents a block language of length ℓ over a k-ary alphabet, thus one can use any alphabet of size k. Boolean bitmap bitwise operations trivially correspond to boolean set operations on block languages of the same length.

Example 1. Let $L = \{aaaa, aaba, aabb, abab, abba, abbb, babb, bbaa, bbab, bbba\}$ be a language over $\{a, b\}$ and $\ell = 4$. The bitmap of L is $\mathsf{B}(L) = 1011011100011110$.

A bitmap $\mathsf{B} \in \{0,1\}^{k^\ell}$ can be split into factors of length k^r, for $r \in [\ell]$. Let $s_i^r = b_{ik^r} \cdots b_{(i+1)k^r - 1}$ denote the i-th factor of length k^r, for $i \in [k^{\ell - r} - 1]$. Since each factor of length k^r can also be split into k factors, s_i^r is inductively defined as:

$$s_i^r = \begin{cases} b_i, & \text{if } r = 0, \\ s_{ik}^{r-1} \cdots s_{(i+1)k-1}^{r-1}, & \text{otherwise.} \end{cases}$$

The following lemma formally introduces the observation that each factor s_i^r, with $r \in [\ell]$ and $i \in [k^{\ell - r} - 1]$, represents the bitmap of a quotient of L.

Lemma 1. *Let $L \subseteq \Sigma^\ell$ be a block language, $|\Sigma| = k$, $\ell \geq 0$, and B the bitmap of L. Let $r \in [\ell]$, $i \in [k^{\ell - r} - 1]$, and $w \in \Sigma^{\ell - r}$ be the word of index i of size $\ell - r$, in lexicographic order. Then, s_i^r corresponds to the bitmap of $w^{-1}L$.*

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

- *Base case* $r = 0$: by definition, $s_i^0 = b_i$. Additionally, we have that $b_i = 1$ if the word w is the i-th word in Σ^ℓ and $w \in L$. Since $|w| = \ell$ either $w^{-1}L = \{\varepsilon\}$ or $w^{-1}L = \emptyset$, according to the membership of w in L.

– *Inductive step*: since $r > 0$, we have that $s_i^r = s_{ik}^{r-1} \cdots s_{(i+1)k-1}^{r-1}$. By hypothesis, we have that s_{ik+j}^{r-1} corresponds to the bitmap of the language $w_j^{-1}L$, where w_j denotes the $(ik+j)$-th word of length $\ell-(r-1)$ over Σ, for each $j \in [k-1]$. One can observe that the words $(w_j)_{j\in[k-1]}$ are all equal on the first $\ell - r$ symbols corresponding to the i-th word of size $\ell - r$. Thus, s_i^r corresponds to the bitmap of the quotient of L by the i-th word of length $\ell - r$. □

Example 2. Recall the Example 1, where $\mathsf{B} = 1011011100011110$, $k = 2$ and $\ell = 4$. We have that $s_0^2 = 1011$ is the bitmap of $(aa)^{-1}L = \{aa, ba, bb\}$, $s_1^3 = 00011110$ the bitmap of $b^{-1}L = \{abb, baa, bab, bba\}$, and $s_0^4 = \mathsf{B}$ the bitmap of L.

Given a bitmap $\mathsf{B} \in \{0,1\}^{k^\ell}$, let $\mathcal{B}_r$ be the set of factors of B of length k^r, for $r \in [\ell]$, in which there is at least one bit different than zero, that is,

$$\mathcal{B}_r = \{\, s \in \{0,1\}^{k^r} \mid \exists i \in [k^{\ell-r} - 1] : s = s_i^r \text{ and } s_i^r \neq 0^{k^r} \,\}.$$

Example 3. Consider the bitmap of Example 1, $\mathsf{B} = 1011011100011110$, with $k = 2$ and $\ell = 4$. We have $\mathcal{B}_0 = \{1\}$, which contains the only factor of length 1 in B different than 0, $\mathcal{B}_1 = \{01, 10, 11\}$, which contains the factors of length 2 different than 00 occurring in even positions of B, $\mathcal{B}_2 = \{0001, 0111, 1011, 1110\}$, which contains the factors of length 4 in positions multiple of 4 in B. Analogously, we have $\mathcal{B}_3 = \{00011110, 10110111\}$ and $\mathcal{B}_4 = \{\mathsf{B}\}$.

The size of $\mathcal{B}_r$ is bounded by the number of factors with size k^r. Additionally, each factor is a composition of factors from the previous set. These two conditions are formally stated in the following lemma.

Lemma 2. *Let $L \subseteq \Sigma^\ell$ be a block language of words of length $\ell \geq 0$ over an alphabet Σ of size $k > 0$, with a correspondent bitmap B. Then, the cardinality of $\mathcal{B}_r$ is bounded by:*

$$|\mathcal{B}_r| \leq \begin{cases} 1, & \text{if } r = 0, \\ \min(k^{\ell-r}, (|\mathcal{B}_{r-1}|+1)^k - 1), & \text{otherwise.} \end{cases}$$

Proof. The case $r = 0$ is trivial, as $\mathcal{B}_0$ contains at most the factor 1. This happens when L is not empty. Since there are at most $k^{\ell-r}$ unique factors of size k^r in a bitmap of size k^ℓ, for $r \in [\ell]$, then $|\mathcal{B}_r| \leq k^{\ell-r}$. Now, let $s \in \mathcal{B}_r$, for some $r \in [1, \ell]$. By definition, $s = s_0 \cdots s_{k-1}$ (with $|s_j| = k^{r-1}$), and either $s_j \in \mathcal{B}_{r-1}$ or it is composed only by zeros, for every $j \in [k-1]$. Since $s \in \mathcal{B}_r$, it must have at least one bit equal to 1. Therefore, $|\mathcal{B}_r| \leq (|\mathcal{B}_{r-1}|+1)^k - 1$. □

The sets $\mathcal{B}_r$ are related to the states of the finite automata representing the block language with bitmap B, as described in the next sections. A finite automaton for a block language is, of course, also acyclic and ranked. If two states belong to the same rank, their right languages contain only words of the same length. All final states belong to Q_0 and, therefore, can be merged. Additionally, if an NFA for a block language has multiple initial states, they can also be merged.

4 Bitmaps for Block Languages and Minimal DFAs

In this section we relate the bitmap of a block language to its minimal DFA. Given a bitmap B associated with a block language $L \subseteq \Sigma^\ell$, with $|\Sigma| = k$ and $\ell \geq 0$, one can directly build a minimal DFA $\mathcal{A}$ for L. Let $Q = \bigcup_{r \in [\ell]} \mathcal{B}_r$ be the set of states of $\mathcal{A}$, and the transition function mapping the states in $\mathcal{B}_r$ with the ones in $\mathcal{B}_{r-1}$, $r \in [1, \ell]$. We will now detail this construction. We start by the final state, that is the factor $1 \in \mathcal{B}_0$, as well as the dead state corresponding to the factor 0. Then, for each rank $r = 1, \ldots, \ell$, we consider every factor $s \in \mathcal{B}_r$ as a state in rank r. As stated in Lemma 1, every factor 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 decomposition of s into $s_0 \cdots s_{k-1}$, where $|s_i| = k^{r-1}$, for every $i \in [k-1]$. Then, we set $\delta(s, \sigma_i) = s_i$, where $s_i \in \mathcal{B}_{r-1}$. Note that, if the language L is not empty, this construction creates exactly one initial and one final state, since $|\mathcal{B}_0| = |\mathcal{B}_\ell| = 1$.

Lemma 3. *Let $L \subseteq \Sigma^\ell$ be a block language with bitmap B, where $\ell \geq 0$. Then, the DFA $\mathcal{A}$ obtained by applying the above construction to B accepts 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 also a factor which, by Lemma 1, 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 $w_1^{-1} L = \emptyset$. Then, since for every word x, $x^{-1}\emptyset = \emptyset$, we get $w^{-1}L = (w_1 w_2)^{-1} L = {w_2}^{-1}(w_1^{-1} L) = \emptyset$. The empty language corresponds to the bitmap associated with the dead state, therefore $w \notin \mathcal{L}(\mathcal{A})$.
- $L \subseteq \mathcal{L}(\mathcal{A})$: Let $w \in L$. Then, $w^{-1}L = \{\varepsilon\}$, whose bitmap is the final state, therefore $w \in \mathcal{L}(\mathcal{A})$. □

Lemma 4. *Let $L \subseteq \Sigma^\ell$ be a block language with bitmap B, where $\ell \geq 0$. Then, the DFA $\mathcal{A}$ obtained by applying the above construction to B is minimal.*

Proof. Let s_1, s_2 be two distinct states of $\mathcal{A}$. It can be noticed that if s_1 and s_2 do not belong to the same rank, they are distinguishable. Otherwise, if they belong to the same rank, by construction, $s_1 \neq s_2$, and consequently they have distinct right languages. Therefore, s_1 and s_2 are distinguishable. □

Combining the results of Lemmas 3 and 4, we obtain:

Theorem 1. *Let $L \subseteq \Sigma^\ell$ be a block language. The above construction of a DFA from the bitmap $\mathsf{B}(L)$ yields the minimal DFA for L.*

Example 4. Let $L \subseteq \{a, b\}^4$ be the language of Example 1 with bitmap $\mathsf{B}(L) =$ 1011011100011110. The correspondent minimal DFA is depicted in Fig. 1. The final state (in rank 0) labeled by s_0^0 represents the bitmap factor 1 and the dead state is omitted as well as all transitions from and to it. States in rank

1 correspond to 2-bit factors, in this case: $s_0^1 = 10$, $s_1^1 = 11$, and $s_2^1 = 01$. We have $\delta(s_0^1, a) = s_0^0$, $\delta(s_2^1, b) = s_0^0$, etc. States in rank 2 correspond to $2^2 = 4$-bit words: $s_0^2 = 1011$, $s_1^2 = 0111$, $s_2^2 = 0001$ and $s_3^2 = 1110$. And we have, for instance, $\delta(s_0^2, a) = s_0^1$ and $\delta(s_0^2, b) = s_1^1$. Similarly for ranks 3 and 4. The initial state corresponds to B.

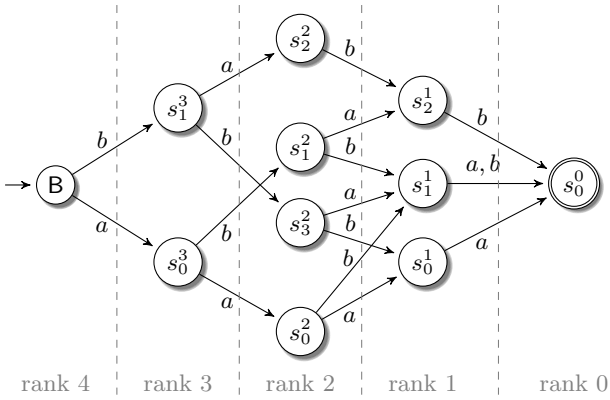

Fig. 1. Minimal DFA accepting the language of Example 1, where $s_0^0 = 1$, $s_0^1 = 10$, $s_1^1 = 11$, $s_2^1 = 01$, $s_0^2 = 1011$, $s_1^2 = 0111$, $s_2^2 = 0001$, $s_3^2 = 1110$, $s_0^3 = 10110111$, $s_1^3 = 00011110$, and $\mathsf{B} = 1011011100011110$.

4.1 Maximal Size of Minimal DFAs for Block Languages

Câmpeanu and Ho [3] showed that the number of states of a DFA accepting a block language $L \subseteq \Sigma^\ell$, over an alphabet of size k and $\ell \geq 0$, is at most $\frac{k^{\ell-r}-1}{k-1} + \sum_{n=0}^{r-1}(2^{k^n} - 1) + 1$, where $r = \min\{n \in [\ell] \mid k^{\ell-n} \leq 2^{k^n} - 1\}$. In the next result we give an estimation of the value of r.

Theorem 2. *Let $\ell > 0$, $k > 1$, and $r = \min\{n \in [\ell] \mid k^{\ell-n} \leq 2^{k^n} - 1\}$. Then, $r = \lfloor \log_k \ell \rfloor + 1 + x$, for some $x \in \{-1, 0, 1\}$.*

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

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

While, from the second inequality, we get:

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

□

We now present a family of *witness languages* recognized by minimal DFAs of maximal size, according to the bounds given in Theorem 2. The bitmaps of these languages correspond to sequences of binary representations of the first m positive integers, as we will see in Lemma 5. Given an integer i, let us denote by $i_{[2]}$ its binary representation, and let $\mathsf{pad}(s, t)$ be the function that adds leading zeros to a binary string s until its length equals t. Moreover, to indicate that the j-th least significant bit of $i_{[2]}$ is 1, we will use the notation $i \wedge 2^j \neq 0$. 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 Σ^ℓ.

Let $\ell > 0$, $k = 2$, and $r = \min\{n \in [\ell] \mid 2^{\ell-n} \leq 2^{2^n} - 1\}$ as in Theorem 2. In a minimal DFA with maximal size, the rank having the largest size is either r or $r - 1$, depending on whether $2^{\ell-r} > 2^{2^{r-1}} - 1$ or not, respectively. Let r_ℓ be that rank and $m = \max(2^{\ell-r}, 2^{2^{r-1}} - 1)$ its width. Then, we consider the following family of witnesses, defined for every $\ell > 0$:

$$\begin{aligned} \mathsf{MAX}_\ell = \{w_1 w_2 \mid & w_1 \in \Sigma^{\ell-r_\ell}, w_2 \in \Sigma^{r_\ell}, \\ & i = \mathsf{ind}(w_1), j = \mathsf{ind}(w_2), (i+1) \wedge 2^j \neq 0\}. \end{aligned}$$

Example 5. For $\ell = 5$, we have $r = \min\{n \in [5] \mid 2^{5-n} \leq 2^{2^n} - 1\} = 2$. Moreover, $m = \max(2^{5-2}, 2^{2^{2-1}} - 1) = \max(8, 3) = 8$, implying that $r_\ell = r$. Then, consider for $\Sigma = \{a, b\}$,

$$\begin{aligned} \mathsf{MAX}_5 = \{&aaaaa, aabab, abaaa, ababa, abbba, baaaa, \\ &baaba, babab, babba, bbaaa, bbaab, bbaba, bbbbb\}. \end{aligned}$$

For example, let $w_1 = baa$ where $i = \mathsf{ind}(w_1) = 4$ and $(i+1)_{[2]} = 101$. For $j = 0$ and $j = 2$, we have that $(i+1) \wedge 2^j \neq 0$, which correspond to the words aa and ba, respectively. Thus, $baaaa, baaba \in \mathsf{MAX}_5$.

Lemma 5. *Let r, r_ℓ, and m be defined as before for $\ell > 0$ and alphabet size $k = 2$. Let $P_{m,r_\ell} = \prod_{i=1}^{m} \mathsf{pad}(i_{[2]}, 2^{r_\ell})^{\mathrm{R}}$. Then, the bitmap of the language MAX_ℓ is given by*

$$\mathsf{B}(\mathsf{MAX}_\ell) = \begin{cases} P_{m,r_\ell}, & \text{if } m = 2^{\ell-r}, \\ P_{m,r_\ell} \cdot 0^{2^{r_\ell}}, & \text{if } m = 2^{2^{r-1}} - 1. \end{cases}$$

Proof. Let us show that the bitmap is correct, for either value of m.

1. $m = 2^{\ell-r}$:
   In this case $r_\ell = r$. Also, $|\,\mathsf{B}(\mathsf{MAX}_\ell)| = 2^\ell$. Let $w_1 \in \Sigma^{\ell-r_\ell}$, $i = \mathsf{ind}(w_1)$, $w_2 \in \Sigma^{r_\ell}$, and $j = \mathsf{ind}(w_2)$. We must prove that $w_2 \in w_1^{-1}\,\mathsf{MAX}_\ell$ if, and only if, the j-th most significant bit of $(i+1)^{\mathrm{R}}_{[2]}$ is set to 1, and vice versa. If $w_2 \in w_1^{-1}\,\mathsf{MAX}_\ell$ then, by definition, the j-th bit of the binary representation of $i+1$ is set to 1, that is, $(i+1) \wedge 2^j \neq 0$, thus the condition holds. In the other direction a similar argument applies.
2. $m = 2^{2^{r-1}} - 1$:
   Now, $r_\ell = r - 1$. Let us first prove that the size of $\mathsf{B}(\mathsf{MAX}_\ell)$ is 2^ℓ. It can be noticed that both $2^{2^{r-1}} - 1 > 2^{\ell-r}$ and $2^{2^{r-1}} - 1 < 2^{\ell-r+1}$ hold. These two conditions imply that $2^{r-1} = \ell - r + 1$. Then, $|\,\mathsf{B}(\mathsf{MAX}_\ell)| = 2^{r-1}2^{2^{r-1}} = 2^{\ell-r+1+r-1} = 2^\ell$.
   Since m is odd, we add a padding of 2^{r_ℓ} zeros to ensure that the length of the bitmap is 2^ℓ. By Lemma 1, these particular bits of the bitmap correspond to $w_1^{-1}\,\mathsf{MAX}_\ell$ such that $\mathsf{ind}(w_1) = 2^{2^{r-1}} - 1$. Thus, we need to prove that $w_1^{-1}\,\mathsf{MAX}_\ell = \emptyset$. By the definition of MAX_ℓ, for $(i+1) \wedge 2^j \neq 0$ to hold, j must be at least 2^{r-1}, but for every $w_2 \in \Sigma^{r_\ell}$ we have $\mathsf{ind}(w_2) \leq 2^{r-1} - 1$. Thus, $w_1^{-1}\,\mathsf{MAX}_\ell = \emptyset$. □

Example 6. According to Lemma 5, the bitmap of the language MAX_5 given in Example 5 is

$$\mathsf{B}(\mathsf{MAX}_5) = \prod_{i=1}^{8} \mathsf{pad}(i_{[2]}, 4)^{\mathrm{R}} = 1000010011000010101001011011100001.$$

To have a DFA of maximal size for a block language contained in Σ^ℓ, for some $\ell \geq 0$, the width of each rank $r' \in [\ell]$ must be either $2^{2^{r'}} - 1$, for $r' \in [r-1]$, or $2^{\ell-r'}$, for $r' \in [r, \ell]$, from which the result from Câmpeanu and Ho was established [3]. From this observation, it follows:

Lemma 6. *Let r, r_ℓ, and m 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 $Q = Q_0 \cup \ldots \cup Q_\ell$ be the set of states of the minimal DFA for MAX_ℓ, such that $q \in Q_{r'}$ is in rank $r' \in [\ell]$. Then, the DFA has maximal size if $|Q_{r'}| = 2^{2^{r'}} - 1$, for $r' \in [r-1]$, and if $|Q_{r'}| = 2^{\ell-r'}$, for $r' \in [r, \ell]$. To that aim, one analyses the cardinalities of the sets $\mathcal{B}_{r'}$, with $r' \in [\ell]$, for the possible values of m.

1. $m = 2^{\ell-r}$:
   (a) $(\forall r' \in [r-1])|\mathcal{B}_{r'}| = 2^{2^{r'}} - 1$: we have that $\mathcal{B}_{r'} = \{\mathsf{pad}(i_{[2]}, 2^{r'})^{\mathrm{R}} \mid \forall i \in [1, m]\}$. Since both $|\mathcal{B}_{r'}| \leq 2^{2^{r'}} - 1$ and $m \geq 2^{2^{r'}} - 1$, the proposition holds.

   (b) $(\forall r' \in [r, \ell])|\mathcal{B}_{r'}| = 2^{\ell - r'}$: clearly $|\mathcal{B}_r| = 2^{\ell - r}$. Let $x \in [\ell - r]$ and $r' = r + x$. The set $\mathcal{B}_{r'}$ is given by splitting $\mathsf{B}(\mathsf{MAX}_\ell)$ into factors of length $2^{r'}$. Of course, $|\mathcal{B}_{r'}| = 2^{\ell - r'}$.
2. $m = 2^{2^{r-1}} - 1$:
   (a) $(\forall r' \in [r-1])|\mathcal{B}_{r'}| = 2^{2^{r'}} - 1$: analogous to the first case (1a).
   (b) $(\forall r' \in [r, \ell])|\mathcal{B}_{r'}| = 2^{\ell - r'}$: it suffices to show that $|\mathcal{B}_r| = 2^{\ell - r}$, as we saw on the previous case (1b). By construction, $\mathsf{B}(\mathsf{MAX}_\ell)$ is composed by m different blocks of length 2^{r_ℓ} and a single block of zeros. Since m is odd, each element of $\mathcal{B}_r$, consisting of blocks of length 2^r, will be equal either to the binary representation of two consecutive numbers or the second number represented is zero. Therefore, $|\mathcal{B}_r| = 2^{2^{r-1}-1}$ and, as mentioned in proof of Lemma 5, $2^{r-1} = \ell - r + 1$. Then, $|\mathcal{B}_r| = 2^{\ell - r}$. □

5 Bitmaps for Block Languages and Minimal NFAs

We now show that the bitmap of a block language L can be used to obtain a minimal NFA for L. However, in this case the problem is NP-complete. Given a bitmap B associated with a block language $L \subseteq \Sigma^\ell$, 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. The main difference with the deterministic case is that the quotients of the language, corresponding to factors from the bitmap, are represented by sets of states, instead of single ones.

First, let us define what a *cover* is. 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, $\mathcal{C}$ covers s if and only if $(\exists\, m \in [1, |\,\mathcal{C}\,|])(\exists \{c_1, \ldots, c_m\} \subseteq \mathcal{C}\,)(\bigvee_{i=1}^{m} c_i = s)$.

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 smaller set that covers $\mathcal{B}$.

Example 7. 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 it is not unique since $\mathcal{C}' = \{1100, 0010, 0001\}$ also cover $\mathcal{B}$ and $|\,\mathcal{C}\,| = |\,\mathcal{C}'\,|$.

Let us now show how to obtain an NFA for a non-empty block language $L \subseteq \Sigma^\ell$, for some $\ell \geq 0$ and an alphabet $\Sigma = \{\sigma_0, \ldots, \sigma_{k-1}\}$ of size k, 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 $\rho : \{0,1\}^\star \to$

$2^{\{0,1\}^\star}$ which maps a factor into the set that covers it. First, let $\rho(1) = \{1\}$. Then, for each rank $i = 1, 2, \ldots, \ell$, we look for the minimal set $\mathcal{C}_i$ that covers the set $\mathcal{B}_i$, the collection of factors of B with length k^i with at least one bit set to 1.

The rank i in the NFA will be $\mathcal{C}_i$. Subsequently, for each factor $s \in \mathcal{B}_i$ we set $\rho(s) = \{c_0, \ldots, c_{m-1}\} \subseteq \mathcal{C}_i$, such that $\rho(s)$ covers s.

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 rank i, we split $c = c_0 \cdots c_{k-1}$, where $|c_j| = k^{i-1}$, for every $j \in [k-1]$, and set $\delta(c, \sigma_j) = \rho(c_j)$, if $c_j \neq 0^{k^{i-1}}$.

We must also guarantee that ρ is defined in c_j, i.e., 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 concatenation 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, $\mathcal{B}_\ell = \{\mathsf{B}\}$, so the minimal cover for $\mathcal{B}_\ell$ is itself. This result implies that B will be the single initial state at rank ℓ.

Lemma 7. *Let $L \subseteq \Sigma^\ell$ be a block language, for some $\ell \geq 0$, with bitmap B. Then, the NFA $\mathcal{A}$ given by the above construction applied to B accepts L, that is, $\mathcal{L}(\mathcal{A}) = L$.*

Proof. Recall the construction of the minimal DFA from a bitmap in Lemma 3. For $r \in [\ell]$, let $s \in \mathcal{B}_r$ be a state from the DFA. By construction, $\rho(s) = \{c_0, \ldots, c_{n-1}\}$, where c_i are states in $\mathcal{A}$ that exactly cover the bitmap s. As $\mathcal{L}(s) = \bigcup_{i \in [n-1]} \mathcal{L}(c_i)$, one concludes that $\mathcal{L}(\mathcal{A}) = L$. □

Lemma 8. *Let $L \subseteq \Sigma^\ell$ be a block language, for some $\ell \geq 0$, with bitmap B. Then, the NFA $\mathcal{A}$ given by the above construction applied to B is minimal.*

Proof. Let $w \in \Sigma^{\ell-r}$, for some $r \in [\ell]$, 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 states P_1 and P_2, respectively. Suppose that $s_1 = s_2$, so P_1 and P_2 cover the same bitmaps. Then, $\mathcal{C}_r \setminus P_2$ would also cover the set $\mathcal{B}_r$, hence $\mathcal{C}_r$ would not be minimal. □

From Lemmas 7 and 8, we obtain the following.

Theorem 3. *The construction of an NFA from a bitmap B of a block language $L \subseteq \Sigma^\ell$ results in a minimal NFA $\mathcal{A}$ such that $\mathcal{L}(\mathcal{A}) = L$.*

Example 8. Let $L \subseteq \{a, b\}^4$ be the language of Example 1 with bitmap $\mathsf{B} = 1011011100011110$. A correspondent NFA is depicted in Fig. 2. One has $\mathcal{B}_1 = \{01, 11, 10\}$ but $\mathcal{C}_1 = \{01, 10\}$ is a minimal cover. Thus, only two states are needed in rank 1 of the NFA. Then, $\mathcal{B}_2 = \{1011, 0111, 0001, 1110\}$, and let $\mathcal{C}_2 = \{1010, 0110, 0001\}$. We have $1011 = 0001 \vee 1010$, $0111 = 0110 \vee 0001$, and $1110 = 1010 \vee 0110$. In rank 3 two states are needed and rank 4 has only the initial state.

The problem of obtaining a minimal NFA from the bitmap is NP-COMPLETE, as proved in the following result.

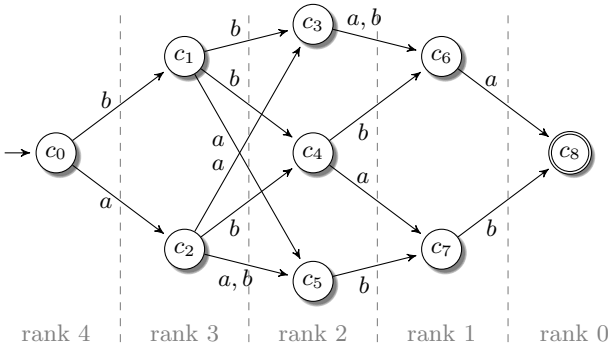

Fig. 2. A minimal NFA accepting the language of Example 1, where $c_8 = 1$, $c_7 = 01$, $c_6 = 10$, $c_5 = 0001$, $c_4 = 0110$, $c_3 = 1010$, $c_2 = 10110111$, $c_1 = 00011110$, and $c_0 = 1011011100011110$.

Theorem 4. *Let* B *be a bitmap of length* k^ℓ*. Given a set of factors* $\mathcal{B}_r$*, each with length* k^r *where* $r \in [\ell]$*, the problem of finding the minimal cover* $\mathcal{C}_r$ *for* $\mathcal{B}_r$ *is* NP-COMPLETE.

Proof. The problem we aim to solve is characterized as:

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

By Lemma 1, each bitmap factor corresponds to a quotient of the language L. Therefore, the bitwise disjunction over factors corresponds to the union over sets. Thus, this problem is the SET-BASIS problem and Stockmeyer proved that it is NP-COMPLETE by reduction to the VERTEX-COVER problem [16]. □

One can use an SMT-solver [10] to find a cover of size n of a set, wherein every factor of size k^r can be represented by a bit vector of the same size, and then use binary search on the solution to determine the minimal one.

5.1 Maximal Size of Minimal NFAs for Block Languages

The width of each rank of the NFA given by the construction described above is bounded by the width of the same rank on the DFA. Also, to cover factors of length k^r, we show that a cover of k^r elements is sufficient. These bounds are formally stated in the following lemma.

Lemma 9. *Let* $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, \{q_f\}\rangle$ *be an NFA for a block language* $L \subseteq \Sigma^\ell$ *over* Σ *of size* k *and* $\ell \geq 0$*, given by the construction of NFAs from bitmaps. Let* $Q = Q_0 \cup \ldots \cup Q_\ell$*, such that* Q_r *is the rank* r*, for* $r \in [\ell]$*. Then, the width of rank* Q_r *is bounded by* $|Q_r| \leq \min(k^{\ell-r}, k^r)$*, for all* $r \in [\ell]$*.*

Proof. The bound $|Q_r| \leq k^{\ell-r}$, for $r \in [\ell]$, refers to the size of $\mathcal{B}_r$, that is, the number of unique factors of length k^r. We showed in Lemma 2 that $|\mathcal{B}_r| \leq k^{\ell-r}$. On the other hand, we have that $|Q_r| \leq k^r$ since the set $\mathcal{B}_r$ can be covered by the set of unit factors $\{u_i\}_{i\in[k^r-1]}$ of size k^r, where u_i represents the factor filled with zeros, apart from the i-th position which is set to 1. □

Lemma 9 allows us to determine the exact maximal size of a minimal NFA for a block language, and easily prove the following theorem.

Theorem 5. *The maximal size of a minimal NFA for a block language $L \subseteq \Sigma^\ell$, with $\ell \geq 0$ and $|\Sigma| = k$, is $nsc(L) \leq 2 \cdot \frac{k^{\frac{\ell}{2}} - 1}{k-1} + k^{\frac{\ell}{2}}$ if ℓ is even, and $nsc(L) \leq 2 \cdot \frac{k^{\lceil \frac{\ell}{2} \rceil} - 1}{k-1}$, otherwise.*

Proof. If ℓ is even, there is an odd number of ranks and the width of the minimal NFA with maximal size is achieved by the rank $\frac{\ell}{2}$. So the maximal number of states is given by $2 \cdot \sum_{r=0}^{\frac{\ell}{2}-1} k^{\ell-r} + k^{\frac{\ell}{2}}$. If ℓ is odd, the width of the minimal NFA with maximal size is reached both in rank $\lceil \frac{\ell}{2} \rceil - 1$ and $\lceil \frac{\ell}{2} \rceil$. So the NFA has at most $2 \cdot \sum_{r=0}^{\lceil \frac{\ell}{2} \rceil - 1} k^{\ell-r}$ states. □

References

1. Almeida, M., Moreira, N., Reis, R.: Exact generation of minimal acyclic deterministic finite automata. Int. J. Found. Comput. S. **19**(4), 751–765 (2008). https://doi.org/10.1142/S0129054108005930
2. Almeida, M., Moreira, N., Reis, R.: Finite automata minimization algorithms. In: Wang, J. (ed.) Handbook of Finite State Based Models and Applications, pp. 145–170. CRC Press (2012)
3. Câmpeanu, C., Ho, W.H.: The maximum state complexity for finite languages. J. Autom. Lang. Comb. **9**(2-3), 189–202 (2004)
4. Dudzinski, K., Konstantinidis, S.: Formal descriptions of code properties: Decidability, complexity, implementation. Int. J. Found. Comput. Sci. **23**(1), 67–85 (2012). https://doi.org/10.1142/S0129054112400059
5. Elvey Price, A., Fang, W., Wallner, M.: Compacted binary trees admit a stretched exponential. J. Comb. Theory, Ser. A **177**, 105306 (2021). https://doi.org/10.1016/J.JCTA.2020.105306
6. Karhumäki, J., Kari, J.: Finite automata, image manipulation, and automatic real functions. In: Pin, J. (ed.) Handbook of Automata Theory, pp. 1105–1143. European Mathematical Society (2021). https://doi.org/10.4171/AUTOMATA-2/8
7. Karhumäki, J., Okhotin, A.: On the determinization blowup for finite automata recognizing equal-length languages. In: C. S. Calude, R.F., Iwama, K. (eds.) Computing with New Resources - Essays Dedicated to Jozef Gruska. LNCS, vol. 8808, pp. 71–82. Springer (2014). https://doi.org/10.1007/978-3-319-13350-8_6
8. Kjos-Hanssen, B., Liu, L.: The number of languages with maximum state complexity. In: Gopal, T.V., Watada, J. (eds.) 15th TAMC. LNCS, vol. 11436, pp. 394–409. Springer (2019). https://doi.org/10.1007/978-3-030-14812-6_24

9. Konstantinidis, S., Moreira, N., Reis, R.: Randomized generation of error control codes with automata and transducers. RAIRO **52**, 169–184 (2018)
10. Kroening, D., Strichman, O.: Decision Procedures:An Algorithmic Point of View. Springer (2016). https://doi.org/10.1007/978-3-662-50497-0
11. Meyer, A.R., Fischer, M.J.: Economy of description by automata, grammars, and formal systems. In: 12th Annual Symposium on Switching and Automata Theory. pp. 188–191. IEEE, Los Alamitos (1971)
12. Priez, J.B.: Enumeration of minimal acyclic automata via generalized parking functions. In: 27th FPSAC. DMTCS, vol. 2471 (2015), https://doi.org/10.46298/dmtcs.2471
13. Revuz, D.: Minimisation of acyclic deterministic automata in linear time. Theoret. Comput. Sci. **92**(1), 181–189 (1992)
14. Salomaa, K., Yu, S.: NFA to DFA transformation for finite languages over arbitrary alphabets. J. Autom. Lang. Comb. **2**(3), 177–186 (1997)
15. Shankar, P., Dasgupta, A., Deshmukh, K., Rajan, B.S.: On viewing block codes as finite automata. Theor. Comput. Sci. **290**(3), 1775–1797 (2003). https://doi.org/10.1016/S0304-3975(02)00083-X
16. Stockmeyer, L.: Set basis problem is NP-complete. Tech. Rep. Report No. RC-5431, IBM Research Center (1976)
17. Stockmeyer, L., Meyer, A.R.: Word problems requiring exponential time: Preliminary report. In: 5th Annual ACM Symposium on Theory of Computing. pp. 1–9. ACM (1973)