State Complexity of Binary Coded Regular Languages

Viliam Geffert(✉), Dominika Pališínová, and Alexander Szabari

Department of Computer Science, P. J. Šafárik University,
Jesenná 5, 04154 Košice, Slovakia
{viliam.geffert,alexander.szabari}@upjs.sk,
dominika.palisinova@student.upjs.sk

Abstract. For the given non-unary input alphabet Σ, a maximal prefix code h mapping strings over Σ to binary strings, and an optimal deterministic finite automaton (DFA) $\mathcal{A}$ with n states recognizing a language $\mathcal{L}$ over Σ, we consider the problem of how many states we need for an automaton $\mathcal{A}'$ that decides membership in $h(\mathcal{L})$, the binary coded version of $\mathcal{L}$. Namely, $\mathcal{A}'$ accepts binary inputs belonging to $h(\mathcal{L})$ and rejects binary inputs belonging to $h(\mathcal{L}^{\mathrm{C}})$, where $\mathcal{L}^{\mathrm{C}}$ is the complement of $\mathcal{L}$. The outcome on inputs that are not valid binary codes for any string in Σ^* can be arbitrary: $\mathcal{A}'$ may accept, reject, or halt in a "don't care" state. We show that any optimal deterministic don't care finite automaton (dcDFA) $\mathcal{A}'$ solving this promise problem uses at most $(\|\Sigma\| - 1)\cdot n$ states but at least n states. We also show that, for each non-unary input alphabet Σ, there exists a maximal binary prefix code h such that, for each $n \geq 2$ and for each N in range from n to $(\|\Sigma\| - 1)\cdot n$, there exists a language $\mathcal{L}$ over Σ such that the optimal DFA recognizing $\mathcal{L}$ uses exactly n states and any optimal dcDFA for solving the above promise problem uses exactly N states. Thus, we have the complete state hierarchy for deciding membership in the binary coded version of $\mathcal{L}$, with no magic numbers in between the lower and upper bounds.

Keywords: state complexity · finite automata · don't care automata · prefix codes · promise problems

1 Introduction

One of the earliest results in automata theory is the subset construction [16]: every n-state nondeterministic finite automaton (NFA) can be replaced by an equivalent deterministic finite automaton (DFA) using at most 2^n states. This raised later the question of whether it is possible, for a given number n, to find some $N \in \{n, \ldots, 2^n\}$ such that there is no optimal DFA with exactly N states, equivalent to some optimal NFA with exactly n states [10]; such numbers were named "magic". The problem was solved in [11], showing that there are no magic numbers for ternary languages, contrary to the unary languages [5]. Since then,

Supported by the Slovak grant contract VEGA 1/0177/21.

Published by Springer Nature Switzerland AG 2022
Y.-S. Han and G. Vaszil (Eds.): DCFS 2022, LNCS 13439, pp. 72–84, 2022.
https://doi.org/10.1007/978-3-031-13257-5_6

the magic numbers were studied for language operations, e.g., in [9], it was shown that, for the intersection of two languages, given by two DFAs with n and m states, we have no magic numbers in $\{1, \ldots, n \cdot m\}$. Such state hierarchies were studied for other operations as well [4,7,9,11].

From a different starting point, we are going to land in yet another complete state hierarchy with no magic numbers. Our initial motivation was the fact that most present-day computers store data in a binary coded form. This raises the following natural question: given a standard DFA $\mathcal{A}$ with n states for a regular language $\mathcal{L}$ over an input alphabet Σ, how many states we need to recognize $h(\mathcal{L})$, the binary coded version of $\mathcal{L}$? Clearly, the answer depends also on $h : \Sigma^* \to \{0,1\}^*$, the binary code in use. In most cases, we can work with the assumption that the *code* is a homomorphism, such that $h(\alpha_1) = h(\alpha_2)$ implies $\alpha_1 = \alpha_2$, so that each encoded string can be unambiguously decoded back. Since it is well known that regular languages are closed under *any* homomorphism (not necessarily a code—see e.g. [8, Sect. 4.2.3]), the situation seems clear at first glance:[1] construct an optimal DFA for $h(\mathcal{L})$.

However, if the automaton for $h(\mathcal{L})$ receives only inputs that are valid binary images of strings in Σ^*, the outcome on inputs that are not valid images can be quite arbitrary, which allows us to save some states. This brings us to a modified problem: given a code $h : \Sigma^* \to \{0,1\}^*$ and a standard DFA $\mathcal{A}$ with n states for $\mathcal{L} \subseteq \Sigma^*$, how many states we need for an automaton $\mathcal{A}'$ that accepts each $\beta \in h(\mathcal{L})$ and rejects each $\beta \in h(\mathcal{L}^{\mathrm{C}})$? Here $\mathcal{L}^{\mathrm{C}}$ denotes the complement of $\mathcal{L}$.

This approach is not completely new: in general, we are given a pair of disjoint languages $\langle \mathcal{L}^{\oplus}, \mathcal{L}^{\ominus} \rangle$ over the same alphabet Σ, called a *promise problem*, and we decide whether $w \in \mathcal{L}^{\oplus}$ or $w \in \mathcal{L}^{\ominus}$ by the use of a *don't care deterministic finite automaton* (dcDFA) which, besides accepting and rejecting states, may also use *neutral* or "don't care" states, otherwise it behaves like a standard DFA (see e.g. [6,13]). In our settings, $\mathcal{L}^{\oplus} = h(\mathcal{L})$ and $\mathcal{L}^{\ominus} = h(\mathcal{L}^{\mathrm{C}})$, where $h : \Sigma^* \to \{0,1\}^*$ is a code. We shall concentrate on the most common binary codes used in practice, that allow decoding by one-way deterministic finite-state transducers in real time and minimize $\sum_{a \in \Sigma} |h(a)|$, the sum of lengths of codewords. Such codes are called *maximal prefix codes* in literature [1,3]. (See Definition 1.)

This paper shows that, for each maximal prefix code $h : \Sigma^* \to \{0,1\}^*$ and each optimal DFA $\mathcal{A}$ with n states recognizing some $\mathcal{L}$ over the alphabet Σ, the binary promise problem $\langle h(\mathcal{L}), h(\mathcal{L}^{\mathrm{C}}) \rangle$ can be solved by a dcDFA $\mathcal{A}'$ using at most $(\|\Sigma\| - 1) \cdot n$ states, but at least n states.[2] We also show that, for each non-unary input alphabet Σ, there exists a maximal binary prefix code h such that, for each $n \geq 2$ and each $N \in \{n, \ldots, (\|\Sigma\| - 1) \cdot n\}$, there exists a language $\mathcal{L} \subseteq \Sigma^*$ such that the optimal DFA recognizing $\mathcal{L}$ uses exactly n states and any optimal dcDFA for solving $\langle h(\mathcal{L}), h(\mathcal{L}^{\mathrm{C}}) \rangle$ uses exactly N states.

[1] State complexity of homomorphisms depends on the length of the images of symbols and is somewhat difficult to define in the general case. Perhaps the only existing related result is the state complexity of projections (that is, homomorphisms mapping each symbol either to itself or to ε), which was determined to be $3/4 \cdot 2^n - 1$ in [12].

[2] Throughout the paper, $\|X\|$ denotes the cardinality of the set X.

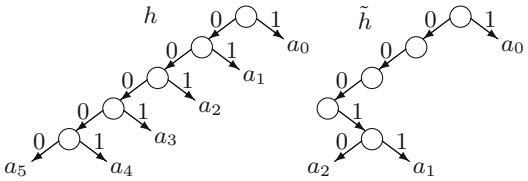

Fig. 1. Examples of homomorphisms establishing some binary prefix codes. Each homomorphism is displayed as a tree in which each leaf represents some a_i, a letter of the original input alphabet; the edges are labeled so that the path from the root to a_i gives the corresponding string $h(a_i)$. Internal nodes of the tree are related to prefixes of strings in $\{h(a) : a \in \Sigma\}$. The code h (left), defined by $h(a_0) = 1$, $h(a_1) = 01$, $h(a_2) = 001$, $h(a_3) = 0001$, $h(a_4) = 00001$, and $h(a_5) = 00000$, is maximal, while the code $\tilde{h}$ (right), with $\tilde{h}(a_0) = 1$, $\tilde{h}(a_1) = 00011$, and $\tilde{h}(a_2) = 00010$, is not—it can be extended, e.g., by defining $\tilde{h}(a_3) = 01$.

2 Preliminaries

Here we shall fix some basic definitions, notation, and preliminary properties. For more details, we refer the reader to [6,8], or any other standard textbooks.

Definition 1. *A* homomorphism *between strings over two alphabets is a mapping* $h : \Sigma_1^* \to \Sigma_2^*$ *preserving concatenation, i.e.,* $h(\alpha_1 \cdot \alpha_2) = h(\alpha_1) \cdot h(\alpha_2)$*, for each* $\alpha_1, \alpha_2 \in \Sigma_1^*$*. The image of a language* $L \subseteq \Sigma_1^*$ *is* $h(L) = \{h(\alpha) : \alpha \in L\} \subseteq \Sigma_2^*$*.*

If $h(\alpha_1) = h(\alpha_2)$ *implies that* $\alpha_1 = \alpha_2$*, then* h *is called a* code*.*

h *is a* prefix code*, if no string in* $h(\Sigma_1) = \{h(a) : a \in \Sigma_1\}$ *is a proper prefix of another one. The code* h *is* maximal*, if there is no other code* $h' : \Sigma_1'^* \to \Sigma_2^*$ *(for some* $\Sigma_1' \supseteq \Sigma_1$*) such that* $h'(\Sigma_1')$ *is a proper superset of* $h(\Sigma_1)$*.*

Each homomorphism h is completely determined by the strings in $h(\Sigma_1)$, since $h(a_1 \cdots a_\ell) = h(a_1) \cdots h(a_\ell)$, for each $a_1, \ldots, a_\ell \in \Sigma_1$. In addition, if h is a code, each $\beta \in h(\Sigma_1^*)$ has a unique factorization into $\beta = \beta_1 \cdots \beta_\ell$, where $\ell \geq 0$ and $\beta_1, \ldots, \beta_\ell \in h(\Sigma_1)$. For examples of codes, see Fig. 1.

Prefix codes allow easy decoding by a one-way deterministic finite-state machine such that, for the given $\beta \in h(\Sigma_1^*)$, it computes the factorization of β into $h(a_1) \cdots h(a_\ell)$ and prints $a_1 \cdots a_\ell$ on the output in real time [1, Prop. 5.1.6].

Maximal codes minimize $\sum_{a \in \Sigma} |h(a)|$ and do not have "gaps" in images: each $\beta \in \Sigma_2^*$ can be extended to an image of some $\alpha \in \Sigma_1^*$, that is, to $\beta \cdot \beta' = h(\alpha)$, for some $\beta' \in \Sigma_2^*$ and some $\alpha \in \Sigma_1^*$, not excluding $\beta' = \varepsilon$.

Since we shall deal with *binary codes* only, we are going to simplify notation and write Σ instead of Σ_1 and fix $\Sigma_2 = \{0, 1\}$.

Definition 2. *A* don't care deterministic finite automaton (dcDFA) *is a* 6*-tuple* $\mathcal{A} = \langle Q, \Sigma, q_{\text{I}}, f, F^{\oplus}, F^{\ominus} \rangle$*, in which* Q *is a finite set of states;* Σ *is a finite input alphabet;* $q_{\text{I}} \in Q$ *is the initial state;* $f : Q \times \Sigma \to Q$ *is a transition function;* $F^{\oplus} \subseteq Q$ *is the set of* accepting *states; and* $F^{\ominus} \subseteq Q$ *the set of* rejecting *states,* $F^{\oplus} \cap F^{\ominus} = \emptyset$*. The remaining states are called* neutral *or "don't care" states.*

A (standard) deterministic finite automaton (DFA) *is a* 5*-tuple* $\mathcal{A} = \langle Q, \Sigma, q_{\mathrm{I}}, f, F\rangle$*, with* $F \subseteq Q$ *denoting the set of accepting states and* $Q\backslash F$ *the set of rejecting states; the remaining components have the same meaning as above.*

The transition function f can be extended to $f^* : Q \times \Sigma^* \to Q$ in a natural way, taking by definition $f^*(q, \varepsilon) = q$ and $f^*(q, \alpha a) = f(f^*(q, \alpha), a)$, for each $q \in Q$, $\alpha \in \Sigma^*$, and $a \in \Sigma$. To simplify notation, $f^*(q, \alpha) = q'$ shall sometimes be displayed in a more compact form $q \xrightarrow{\alpha} q'$.

A *promise problem* (see e.g. [6]) is a pair of disjoint languages $\langle \mathcal{L}^{\oplus}, \mathcal{L}^{\ominus}\rangle$ over the same alphabet Σ. The promise problem is *solved* by a dcDFA $\mathcal{A}$, if $\mathcal{A}$ accepts each $w \in \mathcal{L}^{\oplus}$ (that is, $f^*(q_{\mathrm{I}}, w) \in F^{\oplus}$) and rejects each $w \in \mathcal{L}^{\ominus}$ ($f^*(q_{\mathrm{I}}, w) \in F^{\ominus}$). We do not have to worry about the outcome on inputs belonging neither to $\mathcal{L}^{\oplus}$ nor to $\mathcal{L}^{\ominus}$: on such inputs, $\mathcal{A}$ may accept, reject, or halt in a neutral state.

$\mathcal{A}$ is *optimal* for $\langle \mathcal{L}^{\oplus}, \mathcal{L}^{\ominus}\rangle$, if it solves $\langle \mathcal{L}^{\oplus}, \mathcal{L}^{\ominus}\rangle$ and there is no dcDFA $\mathcal{A}'$ that solves $\langle \mathcal{L}^{\oplus}, \mathcal{L}^{\ominus}\rangle$ with fewer states than does $\mathcal{A}$.

If $\mathcal{L}^{\oplus} \cup \mathcal{L}^{\ominus} = \Sigma^*$, then $\mathcal{A}$ has no neutral reachable states and can be viewed as a standard DFA; we have the standard language recognition and say that $\mathcal{L}^{\oplus}$ is *recognized* by $\mathcal{A}$. The language $\mathcal{L}^{\oplus}$ is then usually denoted by $\mathcal{L}(\mathcal{A})$ and its complement $\mathcal{L}^{\ominus}$ by $\mathcal{L}(\mathcal{A})^{\mathrm{C}}$. In this case, the concept of optimal dcDFA coincide with the concept of minimal DFA for $\mathcal{L}^{\oplus}$.

Note that if a promise problem $\langle \mathcal{L}^{\oplus}, \mathcal{L}^{\ominus}\rangle$ can be solved by a dcDFA $\mathcal{A}$ with n states, it can also be solved by a standard dcDFA $\mathcal{A}'$ with n states, none of which is neutral: any neutral state could be set as accepting or rejecting, without affecting $\langle \mathcal{L}^{\oplus}, \mathcal{L}^{\ominus}\rangle$. This leaves us some degree of freedom, leading to different machines. Namely, if $\mathcal{A}$ uses k neutral reachable states, we obtain 2^k different automata solving the same promise problem, all of them of size at most n. These automata do not agree in acceptance/rejection on inputs not belonging to $\mathcal{L}^{\oplus} \cup \mathcal{L}^{\ominus}$. Thus, the given dcDFA can also be viewed as a more concise template representing these 2^k DFAs.

This is related to the following *separation problem*: given DFAs $\mathcal{A}^{\oplus}$ and $\mathcal{A}^{\ominus}$ for two disjoint languages $\mathcal{L}^{\oplus}$ and $\mathcal{L}^{\ominus}$, find a DFA $\mathcal{A}'$ with minimal number of states, such that $\mathcal{L}^{\oplus} \subseteq \mathcal{L}(\mathcal{A}')$ and $\mathcal{L}^{\ominus} \subseteq \mathcal{L}(\mathcal{A}')^{\mathrm{C}}$. In general, this problem is NP-complete [13, Thm. 9]. This was shown by a simple application of NP-completeness for a slightly different computational model (in which some transitions $f(q, a)$ may be undefined), presented in [14,15].

The next theorem will play the same role for don't care automata as the *fooling set technique* [2] for standard automata:

Theorem 1. *Let* $\mathcal{L}^{\oplus}$*,* $\mathcal{L}^{\ominus}$ *be two disjoint languages over the same alphabet* Σ*. Suppose there exist* m*-tuple* $X = \langle x_e\rangle_{e=1}^{m}$ *and* $\binom{m}{2}$*-tuple* $Y = \langle y_{e,\tilde{e}}\rangle_{e,\tilde{e}=1,\, e<\tilde{e}}^{m}$ *consisting of strings in* Σ^* *such that, for each* $e, \tilde{e} \in \{1, \ldots, m\}$*,* $e < \tilde{e}$*,*

(I) *both* $x_e{\cdot}y_{e,\tilde{e}}$ *and* $x_{\tilde{e}}{\cdot}y_{e,\tilde{e}}$ *are in* $\mathcal{L}^{\oplus} \cup \mathcal{L}^{\ominus}$*,*
(II) $x_e{\cdot}y_{e,\tilde{e}} \in \mathcal{L}^{\oplus}$ *if and only if* $x_{\tilde{e}}{\cdot}y_{e,\tilde{e}} \in \mathcal{L}^{\ominus}$*.*

Then any dcDFA solving the promise problem $\langle \mathcal{L}^{\oplus}, \mathcal{L}^{\ominus}\rangle$ *uses at least* m *states.*

Proof. Let $\mathcal{A} = \langle Q, \Sigma, q_{\text{I}}, f, F^{\oplus}, F^{\ominus}\rangle$, satisfying $F^{\oplus} \cap F^{\ominus} = \emptyset$, be an arbitrary dcDFA for solving $\langle \mathcal{L}^{\oplus}, \mathcal{L}^{\ominus}\rangle$. Suppose that $\mathcal{A}$ uses less than m states.

Now, for the given m-tuple $X = \langle x_1, x_2, \ldots, x_m\rangle$, let $q_1, q_2, \ldots, q_m$ denote the respective states reached by $\mathcal{A}$ on these inputs, that is, $q_{\text{I}} \xrightarrow{x_e} q_e$, for each $e \in \{1, \ldots, m\}$. But then, using a pigeonhole argument, some state must be repeated, i.e., we have $q_e = q_{\tilde{e}}$, for some $1 \le e < \tilde{e} \le m$. This implies that the computations on inputs $x_e{\cdot}y_{e,\tilde{e}}$ and $x_{\tilde{e}}{\cdot}y_{e,\tilde{e}}$ must end in the same state, denoted here by r. That is, we have the following computations: $q_{\text{I}} \xrightarrow{x_e} q_e \xrightarrow{y_{e,\tilde{e}}} r$ and $q_{\text{I}} \xrightarrow{x_{\tilde{e}}} q_{\tilde{e}} = q_e \xrightarrow{y_{e,\tilde{e}}} r$. There are now two possibilities:

First, let $x_e{\cdot}y_{e,\tilde{e}} \in \mathcal{L}^{\oplus}$. Then, using (II), we also have that $x_{\tilde{e}}{\cdot}y_{e,\tilde{e}} \in \mathcal{L}^{\ominus}$. This implies that the computation on $x_e{\cdot}y_{e,\tilde{e}}$ ends in $r \in F^{\oplus}$ and, at the same time, the computation on $x_{\tilde{e}}{\cdot}y_{e,\tilde{e}}$ in $r \in F^{\ominus}$. But this is a contradiction: $F^{\oplus} \cap F^{\ominus} = \emptyset$.

Second, let $x_e{\cdot}y_{e,\tilde{e}} \notin \mathcal{L}^{\oplus}$. Then, using (II), we have $x_{\tilde{e}}{\cdot}y_{e,\tilde{e}} \notin \mathcal{L}^{\ominus}$. Now, by (I), we see that $x_e{\cdot}y_{e,\tilde{e}} \in \mathcal{L}^{\ominus}$ and $x_{\tilde{e}}{\cdot}y_{e,\tilde{e}} \in \mathcal{L}^{\oplus}$. This leads to the same kind of contradiction as above, swapping the roles of $x_e{\cdot}y_{e,\tilde{e}}$ and $x_{\tilde{e}}{\cdot}y_{e,\tilde{e}}$. □

Note that, apart from providing the lower bound on the number of states, the *statement* of the above theorem does not deal with states in any dcDFA solving $\langle \mathcal{L}^{\oplus}, \mathcal{L}^{\ominus}\rangle$ but, rather, with *strings* in Σ^*. However, if there does exist a dcDFA $\mathcal{A}$ with m states for $\langle \mathcal{L}^{\oplus}, \mathcal{L}^{\ominus}\rangle$, that is, if the lower bound provided by Theorem 1 matches the upper bound, then one can establish a one-to-one correspondence between states in $\mathcal{A}$ and strings in $X = \langle x_1, x_2, \ldots, x_m\rangle$, with $Y = \langle y_{e,\tilde{e}}\rangle_{e,\tilde{e}=1,\, e<\tilde{e}}^{m}$ giving the pairwise distinguishability of states in $\mathcal{A}$. The standard fooling set technique (for DFAs) uses $y_{e,\tilde{e}_1} = y_{e,\tilde{e}_2} = \ldots$, with the condition (I) satisfied automatically.

3 Upper and Lower Bounds

We are now going to show that $(\|\Sigma\| - 1){\cdot}n$ states are sufficient but n states necessary for a dcDFA that decides whether the given binary input is in $h(\mathcal{L})$ or in $h(\mathcal{L}^{\text{C}})$, that is, for a dcDFA solving $\langle h(\mathcal{L}), h(\mathcal{L}^{\text{C}})\rangle$, the binary promise-problem version of $\mathcal{L}$. Here $h : \Sigma^* \to \{0,1\}^*$ is a maximal prefix code and $\mathcal{A}$ is an optimal DFA with n states, recognizing a language $\mathcal{L}$ over a non-unary alphabet Σ.

For the given code h, let us begin by fixing some additional notation for the *images of letters* and *proper prefixes*:

$$\begin{aligned} H &= \{h(a) : a \in \Sigma\}, \\ P &= \{\pi : \pi{\cdot}\beta \in H, \text{ for some } \beta \in \{0,1\}^+\}. \end{aligned} \tag{1}$$

Recall that h is a maximal prefix code. Thus, P includes the empty string ε, but no string from H. Next, if $\pi \in P$, then $\pi{\cdot}0, \pi{\cdot}1 \in P \cup H$ (see also Fig. 1). The next two theorems provide the upper and lower bounds.

Theorem 2. *Let $h : \Sigma^* \to \{0,1\}^*$ be a maximal binary prefix code and let $\mathcal{L}$ be a language over the non-unary alphabet Σ. Then, if $\mathcal{L}$ can be recognized by a DFA $\mathcal{A} = \langle Q, \Sigma, q_{\text{I}}, f, F\rangle$ with n states, the binary promise problem $\langle h(\mathcal{L}), h(\mathcal{L}^{\text{C}})\rangle$ can be solved by a dcDFA $\mathcal{A}'$ with at most $n' \le (\|\Sigma\| - 1){\cdot}n$ states.*

Proof (a sketch). The idea of the construction is to remember $q \in Q$, the current state of $\mathcal{A}$ at the moment when $\mathcal{A}$ has read $a_1 \cdots a_\ell \in \Sigma^*$, and $\pi \in P$, the prefix of a code for the next input symbol $a_{\ell+1}$, not completed yet. This leads to $Q' = Q \times P$, with $q'_{\text{I}} = \langle q_{\text{I}}, \varepsilon \rangle$, $F^{\oplus} = F \times \{\varepsilon\}$, and $F^{\ominus} = (Q \backslash F) \times \{\varepsilon\}$. Transitions in $\mathcal{A}'$ are defined as follows, for each $q \in Q$, $\pi \in P$, and $b \in \{0, 1\}$:

(I) $f'(\langle q, \pi \rangle, b) = \langle q, \pi b \rangle$, provided that $\pi \cdot b \in P$,
(II) $f'(\langle q, \pi \rangle, b) = \langle f(q, a), \varepsilon \rangle$, provided that, for some $a \in \Sigma$, $\pi \cdot b = h(a) \in H$.

□

The above construction can be updated so that it works for prefix codes that are not maximal. Then $\pi \in P$ does not imply that $\pi \cdot b$ is in $P \cup H$. In such cases, we can define $f'(\langle q, \pi \rangle, b) = q'_{\text{E}}$, where q'_{E} is an additional trap state, in which we scan the rest of the input—the input can no longer be extended to a string in $h(\Sigma_1^*) = h(H^*)$. However, for such codes, $\|P\|$ is not bounded by $\|\Sigma\| - 1$.

Theorem 3. *Let $h : \Sigma^* \to \{0, 1\}^*$ be a binary prefix code (not necessarily maximal) and let $\mathcal{L}$ be a language over the alphabet Σ (not necessarily non-unary). Then, if the binary promise problem $\langle h(\mathcal{L}), h(\mathcal{L}^{\text{C}}) \rangle$ can be solved by a dcDFA $\mathcal{A}' = \langle Q', \{0, 1\}, q'_{\text{I}}, f', F^{\oplus}, F^{\ominus} \rangle$ with n' states, the language $\mathcal{L}$ can be recognized by a standard DFA $\mathcal{A} = \langle Q, \Sigma, q_{\text{I}}, f, F \rangle$ with at most $n \leq n'$ states.*

Proof. If, for some $q, q' \in Q'$ and $a \in \Sigma$, the original $\mathcal{A}'$ has a path beginning in q, ending in q', and reading from the input the string $h(a) \in H$ (introduced by (1)), we shall add the transition $q \xrightarrow{a} q'$ to $\mathcal{A}$. Recall that $\mathcal{A}'$ is deterministic and $h(a) = b_1 \cdots b_m$ is unique for each $a \in \Sigma$. But then $q' = f'^*(q, h(a))$ is also unique for each $q \in Q'$ and each $a \in \Sigma$, and hence $\mathcal{A}$ will be deterministic.

In addition, we can restrict the set of finite control states in $\mathcal{A}$ to states that can be reached from q'_{I} by reading some $\beta \in h(\Sigma^*) = H^*$, that is, to

$$R = \{f'^*(q'_{\text{I}}, \beta) : \beta \in H^*\}. \tag{2}$$

Thus, $Q = R$, $q_{\text{I}} = q'_{\text{I}}$, and $F = R \cap F^{\oplus}$. Clearly, $R \subseteq F^{\oplus} \cup F^{\ominus}$, since no $\mathcal{A}'$ solving $\langle h(\mathcal{L}), h(\mathcal{L}^{\text{C}}) \rangle$ halts in a neutral state on any $h(\alpha) \in h(\Sigma^*)$. Finally, let

$f(q, a) = f'^*(q, h(a))$, for each $q \in R$ and each $a \in \Sigma$.

It is easy to see, for each $q, q' \in R$ and each $a \in \Sigma$, that $\mathcal{A}$ has a transition from q to q' reading the symbol $a \in \Sigma$ if and only if $\mathcal{A}'$ has a path connecting the same states and reading the string $h(a) \in H$. By a straightforward induction on the length of $\alpha \in \Sigma^*$, we have $q \xrightarrow{\alpha} q'$ in $\mathcal{A}$ if and only if $q \xrightarrow{h(\alpha)} q'$ in $\mathcal{A}'$.

Thus, if $\alpha \in \mathcal{L}$, then $h(\alpha) \in h(\mathcal{L})$ must be accepted by dcDFA $\mathcal{A}'$, which gives $q'_{\text{I}} \xrightarrow{h(\alpha)} q'$ for some $q' \in F^{\oplus}$. Moreover, since $h(\alpha) \in H^*$, q' must also belong to R. But then, for $\mathcal{A}$, we have $q_{\text{I}} = q'_{\text{I}} \xrightarrow{\alpha} q'$ with $q' \in R \cap F^{\oplus} = F$, and hence α is accepted by $\mathcal{A}$. Similarly, if $\alpha \notin \mathcal{L}$, then $\mathcal{A}'$ has a path $q'_{\text{I}} \xrightarrow{h(\alpha)} q'$ ending in $q' \in R \cap F^{\ominus}$. For $\mathcal{A}$, this gives $q_{\text{I}} = q'_{\text{I}} \xrightarrow{\alpha} q'$ ending in $q' \in R \backslash F^{\oplus} = R \backslash (R \cap F^{\oplus}) = Q \backslash F$, and hence α is rejected by $\mathcal{A}$.

Summing up, $\mathcal{A}$ is a standard DFA recognizing $\mathcal{L}$, with $n \leq n'$ states. □

By combining Theorems 2 and 3, we get:

Theorem 4. *Let $h:\Sigma^*\to\{0,1\}^*$ be a maximal binary prefix code and let $\mathcal{L}$ be a language over the non-unary alphabet Σ. Then, if the optimal DFA $\mathcal{A}$ recognizing $\mathcal{L}$ uses n states, any optimal dcDFA $\mathcal{A}'$ solving the binary promise problem $\langle h(\mathcal{L}), h(\mathcal{L}^{\rm C})\rangle$ uses at least n states and at most $(\|\Sigma\|-1)\cdot n$ states.*

Proof. The upper bound is a direct consequence of Theorem 2: this theorem does not necessarily produce a dcDFA that is optimal, however, it does guarantee $(\|\Sigma\|-1)\cdot n$ states for $\mathcal{A}'$. The lower bound follows from Theorem 3: suppose that $\langle h(\mathcal{L}), h(\mathcal{L}^{\rm C})\rangle$ can be solved by $\mathcal{A}'$ with $n'<n$ states. But then, by this theorem, we could obtain a standard DFA recognizing $\mathcal{L}$ with fewer states than n, which is a contradiction, since $\mathcal{A}$ is optimal. □

4 The Hierarchy

We are now ready to establish the complete state hierarchy and provide a witness language for each N between n and $(\|\Sigma\|-1)\cdot n$. First, for the given

$$\Sigma=\{a_0,a_1,\ldots,a_d\},\ \text{where } d=\|\Sigma\|-1\geq 2,$$

define a maximal binary prefix code $h_\Sigma:\Sigma^*\to\{0,1\}^*$ as follows:

$$\begin{array}{l} h_\Sigma(a_j)=0^j1,\ \text{for } j\in\{0,\ldots,d-1\},\\ h_\Sigma(a_d)=0^d. \end{array} \tag{3}$$

Second, for the given Σ and any given

$$n\geq 2,\quad g\in\{0,\ldots,n\},\quad k\in\{0,\ldots,d-2\}, \tag{4}$$

define a DFA $\mathcal{A}_{\Sigma,n,g,k}=\langle Q,\Sigma,q_{\rm I},f,F\rangle$ with the state set $Q=\{0,\ldots,n-1\}$, $q_{\rm I}=0$, $F=\{0\}$, and the following transitions:

$$\begin{array}{rll} f(i,a)=(i+1)\bmod n, & \text{if } i<g \text{ and } a\in\Sigma,\\ f(g,a)=(g+1)\bmod n, & \text{if } g\leq n-2 \text{ and } a\in\{a_0,\ldots,a_k\}\cup\{a_d\},\\ \quad=(g+2)\bmod n, & \text{if } g\leq n-2 \text{ and } a\in\{a_{k+1},\ldots,a_{d-1}\},\\ f(i,a)=(i+1)\bmod n, & \text{if } i>g \text{ and } a\in\Sigma\backslash\{a_d\},\\ \quad=i, & \text{if } i>g \text{ and } a=a_d. \end{array} \tag{5}$$

There are two special cases. The first one is $g=n-1$, with no states $i\in Q$ satisfying $i>g$. Moreover, this case differs in one value, namely, in $f(g,a_d)$:

$$\begin{array}{l} f(g,a)=(g+1)\bmod n=0,\ \text{if } g=n-1 \text{ and } a\in\{a_0,\ldots,a_k\},\\ \quad=(g+2)\bmod n=1,\ \text{if } g=n-1 \text{ and } a\in\{a_{k+1},\ldots,a_d\}. \end{array} \tag{6}$$

The second special case is $g=n$, with no states $i\in Q$ satisfying $i>g$ or $i=g$. Thus, the condition $i<g$ is satisfied automatically for each $i\in Q$, which reduces (5)–(6) above to $f(i,a)=(i+1)\bmod n$ for each $i\in Q$ and each $a\in\Sigma$.

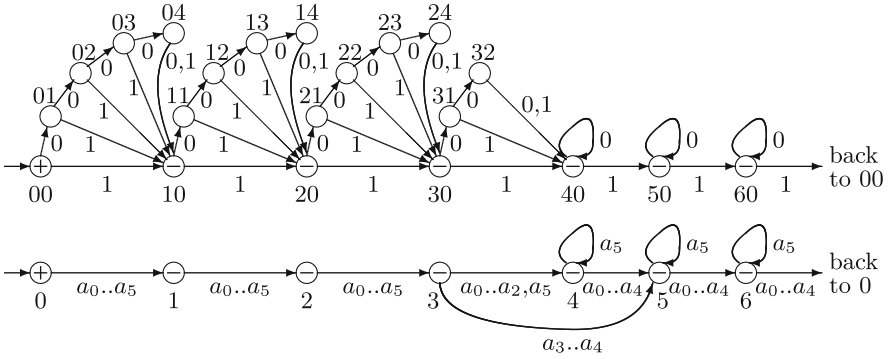

Fig. 2. Examples of a DFA $\mathcal{A}_{\Sigma,n,g,k}$ (bottom) and the corresponding dcDFA $\mathcal{A}'_{\Sigma,n,g,k}$ (top), if $\Sigma = \{a_0, \ldots, a_5\}, d = 5, n = 7, g = 3$, and $k = 2$. Accepting states are tagged by "+", rejecting states by "−", and neutral don't care states by no sign. To simplify notation for $\mathcal{A}'_{\Sigma,n,g,k}$, the ordered pairs "$\langle i, j\rangle$" are displayed here in the form "ij".

Examples of h_Σ, $\mathcal{A}_{\Sigma,n,g,k}$, and subsequent $\mathcal{A}'_{\Sigma,n,g,k}$ are displayed in Fig. 1 (left), Fig. 2 (bottom), and Fig. 2 (top), respectively, for $d = 5$, $n = 7$, $g = 3$, and $k = 2$. The special case of $g = n - 1$ is illustrated by Fig. 3.

Finally, for the given Σ, n, g, k satisfying (4), consider a dcDFA $\mathcal{A}'_{\Sigma,n,g,k} = \langle Q', \{0,1\}, q'_{\mathrm{I}}, f', F^{\oplus}, F^{\ominus}\rangle$, *not* constructed for $\mathcal{A}_{\Sigma,n,g,k}$ by the use of Theorem 2, but defined as follows. First, let $Q' = Q_0 \cup \ldots \cup Q_{n-1}$, where

$$\begin{aligned} Q_i &= \{\langle i, 0\rangle, \ldots, \langle i, d-1\rangle\}, \text{ for } i \in \{0, \ldots, g-1\}, \\ Q_g &= \{\langle g, 0\rangle, \ldots, \langle g, k\rangle\}, \\ Q_i &= \{\langle i, 0\rangle\}, \qquad\qquad \text{for } i \in \{g+1, \ldots, n-1\}. \end{aligned} \tag{7}$$

There are no sections Q_i with $i > g$, if $g = n - 1$, and no section Q_g, if $g = n$, in accordance with the two special cases for $\mathcal{A}_{\Sigma,n,g,k}$.

Now, let $q'_{\mathrm{I}} = \langle 0, 0\rangle, F^{\oplus} = \{\langle 0, 0\rangle\}$, and $F^{\ominus} = \{\langle 1, 0\rangle, \langle 2, 0\rangle, \ldots, \langle n-1, 0\rangle\}$. Transitions are defined as follows:

$$\begin{aligned} f'(\langle i, j\rangle, 1) &= \langle (i+1) \bmod n, 0\rangle, && \text{for each } \langle i, j\rangle \in Q', \\ f'(\langle i, j\rangle, 0) &= \langle i, j+1\rangle, && \text{if } i < g \text{ and } j < d-1, \\ &= \langle (i+1) \bmod n, 0\rangle, && \text{if } i < g \text{ and } j = d-1, \\ f'(\langle g, j\rangle, 0) &= \langle g, j+1\rangle, && \text{if } j < k, \\ &= \langle (g+1) \bmod n, 0\rangle && = \langle g+1, 0\rangle, \text{ if } g \le n-2 \text{ and } j = k, \\ &= \langle (g+1) \bmod n, j+1\rangle && = \langle 0, k+1\rangle, \text{ if } g = n-1 \text{ and } j = k, \\ f'(\langle i, 0\rangle, 0) &= \langle i, 0\rangle, && \text{if } i > g. \end{aligned} \tag{8}$$

Note that also here the case of $g = n - 1$ is different. (See also Fig. 3).

Lemma 1. *Let h_Σ, $\mathcal{A}_{\Sigma,n,g,k}$, and $\mathcal{A}'_{\Sigma,n,g,k}$ be the binary code, DFA, and dcDFA defined above. Then $\mathcal{A}'_{\Sigma,n,g,k}$ solves the binary promise problem $\langle h_\Sigma(\mathcal{L}(\mathcal{A}_{\Sigma,n,g,k})), h_\Sigma(\mathcal{L}(\mathcal{A}_{\Sigma,n,g,k})^{\mathrm{C}})\rangle$.*

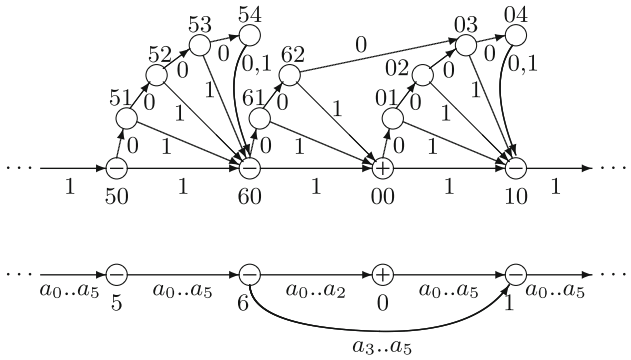

Fig. 3. Examples of $\mathcal{A}_{\Sigma,n,g,k}$ and $\mathcal{A}'_{\Sigma,n,g,k}$ for the special case of $g = n-1$, namely, for $\Sigma = \{a_0, \ldots, a_5\}, d = 5, n = 7,\ g = n-1 = 6$, and $k = 2$ (graph rotated, so that the state $g = n-1$ is not displayed at the right end).

Proof. First, it is not too hard to see that if $\mathcal{A}_{\Sigma,n,g,k}$ has a transition $i \xrightarrow{a} i'$, then $\mathcal{A}'_{\Sigma,n,g,k}$ has the corresponding computation path $\langle i,0\rangle \xrightarrow{h_\Sigma(a)} \langle i',0\rangle$. This can be shown by consulting (3) and by comparing all transitions presented by (5), (6), and (8), for each $i \in Q$ and each $a \in \Sigma$:

- $f(i,a) = (i+1) \bmod n$, if $i < g$ and $a \in \Sigma$ (which covers the case of $g = n$):
  - $\langle i,0\rangle \xrightarrow{0^j} \langle i,j\rangle \xrightarrow{1} \langle (i+1) \bmod n, 0\rangle$, for $h_\Sigma(a) = 0^j 1$, $j \in \{0,\ldots,d-1\}$,
  - $\langle i,0\rangle \xrightarrow{0^{d-1}} \langle i,d-1\rangle \xrightarrow{0} \langle (i+1) \bmod n, 0\rangle$, for $h_\Sigma(a) = 0^d$.
- $f(g,a) = (g+1) \bmod n$, if $g \le n-2$ and $a \in \{a_0,\ldots,a_k\} \cup \{a_d\}$:
  - $\langle g,0\rangle \xrightarrow{0^j} \langle g,j\rangle \xrightarrow{1} \langle (g+1) \bmod n, 0\rangle$, for $h_\Sigma(a) = 0^j 1$, $j \in \{0,\ldots,k\}$,
  - $\langle g,0\rangle \xrightarrow{0^k} \langle g,k\rangle \xrightarrow{0} \langle g+1,0\rangle \xrightarrow{0^{d-k-1}} \langle g+1,0\rangle = \langle (g+1) \bmod n, 0\rangle$, for $h_\Sigma(a) = 0^d$.
- $f(g,a) = (g+2) \bmod n$, if $g \le n-2$ and $a \in \{a_{k+1},\ldots,a_{d-1}\}$:
  - $\langle g,0\rangle \xrightarrow{0^k} \langle g,k\rangle \xrightarrow{0} \langle g+1,0\rangle \xrightarrow{0^{j-k-1}} \langle g+1,0\rangle \xrightarrow{1} \langle (g+2) \bmod n, 0\rangle$, for $h_\Sigma(a) = 0^j 1$, $j \in \{k+1,\ldots,d-1\}$.
- $f(i,a) = (i+1) \bmod n$, if $i > g$ and $a \in \Sigma \backslash \{a_d\}$:
  - $\langle i,0\rangle \xrightarrow{0^j} \langle i,0\rangle \xrightarrow{1} \langle (i+1) \bmod n, 0\rangle$, for $h_\Sigma(a) = 0^j 1$, $j \in \{0,\ldots,d-1\}$.
- $f(i,a) = i$, if $i > g$ and $a = a_d$:
  - $\langle i,0\rangle \xrightarrow{0^d} \langle i,0\rangle$, for $h_\Sigma(a) = 0^d$.

The same can be seen for different transitions in the case of $g = n-1$:

- $f(g,a) = (g+1) \bmod n = 0$, if $g = n-1$ and $a \in \{a_0,\ldots,a_k\}$:
  - $\langle g,0\rangle \xrightarrow{0^j} \langle g,j\rangle \xrightarrow{1} \langle (g+1) \bmod n, 0\rangle$, for $h_\Sigma(a) = 0^j 1$, $j \in \{0,\ldots,k\}$.
- $f(g,a) = (g+2) \bmod n = 1$, if $g = n-1$ and $a \in \{a_{k+1},\ldots,a_d\}$:
  - $\langle g,0\rangle \xrightarrow{0^k} \langle g,k\rangle \xrightarrow{0} \langle 0,k+1\rangle \xrightarrow{0^{j-k-1}} \langle 0,j\rangle \xrightarrow{1} \langle 1,0\rangle = \langle (g+2) \bmod n, 0\rangle$, for $h_\Sigma(a) = 0^j 1$, $j \in \{k+1,\ldots,d-1\}$,
  - $\langle g,0\rangle \xrightarrow{0^k} \langle g,k\rangle \xrightarrow{0} \langle 0,k+1\rangle \xrightarrow{0^{d-k-2}} \langle 0,d-1\rangle \xrightarrow{0} \langle 1,0\rangle = \langle (g+2) \bmod n, 0\rangle$, for $h_\Sigma(a) = 0^d$.

Now, by induction on the length of $\alpha = a_{i_1} \cdots a_{i_\ell} \in \Sigma^*$, we easily obtain that the computation path $i \xrightarrow{\alpha} i'$ in $\mathcal{A}_{\Sigma,n,g,k}$ implies the existence of the corresponding path $\langle i, 0\rangle \xrightarrow{h_\Sigma(\alpha)} \langle i', 0\rangle$ for $\mathcal{A}'_{\Sigma,n,g,k}$, for each $i, i' \in Q$ and each $\alpha \in \Sigma$.

Thus, if $\mathcal{A}_{\Sigma,n,g,k}$ has a path $q_{\text{I}} = 0 \xrightarrow{\alpha} 0 \in F$, then $\mathcal{A}'_{\Sigma,n,g,k}$ has the corresponding path $q'_{\text{I}} = \langle 0, 0\rangle \xrightarrow{h_\Sigma(\alpha)} \langle 0, 0\rangle \in F^{\oplus}$, and hence $h_\Sigma(\alpha)$ is accepted by $\mathcal{A}'_{\Sigma,n,g,k}$, if $\alpha \in \mathcal{L}(\mathcal{A}_{\Sigma,n,g,k})$. On the other hand, if this path in $\mathcal{A}_{\Sigma,n,g,k}$ halts in some $i' \neq 0$, that is, in some $i' \in Q \backslash F$, the corresponding path in $\mathcal{A}'_{\Sigma,n,g,k}$ will halt in $\langle i', 0\rangle \in F^{\ominus}$, and hence $h_\Sigma(\alpha)$ is rejected by $\mathcal{A}'_{\Sigma,n,g,k}$, if $\alpha \in \mathcal{L}(\mathcal{A}_{\Sigma,n,g,k})^{\text{C}}$.

Therefore, $\mathcal{A}'_{\Sigma,n,g,k}$ is a valid dcDFA for solving the binary promise problem $\langle h_\Sigma(\mathcal{L}(\mathcal{A}_{\Sigma,n,g,k})),\ h_\Sigma(\mathcal{L}(\mathcal{A}_{\Sigma,n,g,k})^{\text{C}})\rangle$. □

$\mathcal{A}'_{\Sigma,n,g,k}$ uses fewer states than dcDFA obtained by the use of Theorem 2, but it may accept/reject some binary inputs that are not images of any $\alpha \in \Sigma^*$. That is, it does not necessarily halt in a neutral state on such inputs.

Lemma 2. *Let $\mathcal{A}_{\Sigma,n,g,k}$ be the DFA defined above. Then $\mathcal{A}_{\Sigma,n,g,k}$ is optimal and uses exactly n states.*

Proof. Using (5) and (6), we see that $f(i, a_0) = (i + 1) \bmod n$, for each $i \in Q = \{0, \ldots, n-1\}$, not excluding the special cases of $g = n-1$ or $g = n$. Since $q_{\text{I}} = 0$ and $F = \{0\}$, $a_0^e \in \mathcal{L}(\mathcal{A}_{\Sigma,n,g,k})$ if and only if e is an integer multiple of n.

This implies that $\mathcal{A}_{\Sigma,n,g,k}$ cannot be replaced by an equivalent DFA using fewer states: such DFA would accept a_0^n by a computation path $r_0 \xrightarrow{a_0} r_1 \xrightarrow{a_0} r_2 \xrightarrow{a_0} \cdots \xrightarrow{a_0} r_n$ along which some state would be repeated, which gives a valid accepting computation path for some a_0^e with $0 < e < n$, a contradiction. □

Lemma 3. *Let h_Σ, $\mathcal{A}_{\Sigma,n,g,k}$, and $\mathcal{A}'_{\Sigma,n,g,k}$ be the binary code, DFA, and dcDFA defined above. Then $\mathcal{A}'_{\Sigma,n,g,k}$ is optimal for solving the binary promise problem $\langle h_\Sigma(\mathcal{L}(\mathcal{A}_{\Sigma,n,g,k})), h_\Sigma(\mathcal{L}(\mathcal{A}_{\Sigma,n,g,k})^{\text{C}})\rangle$ and uses exactly $m = (\|\Sigma\| - 1)\cdot g + (k + 1) + (n - g - 1)$ states, if $g \leq n-1$, but exactly $m = (\|\Sigma\| - 1)\cdot n$ states, if $g = n$.*

Proof. By Lemma 1, $\mathcal{A}'_{\Sigma,n,g,k}$ solves the given promise problem and, by (7), it uses $m = d\cdot g + (k+1) + (n-g-1)$ states, if $g \leq n-1$. For $g = n$, all sections Q_i are of equal size d in (7), which gives $m = d\cdot n$. Since $d = \|\Sigma\| - 1$, the upper bound for $m = \|Q'\|$ follows.

We only have to show that this bound cannot be reduced. Let $\langle i, j\rangle, \langle \tilde{\imath}, \tilde{\jmath}\rangle$ represent two arbitrary—but different—states in Q', that is, either $i < \tilde{\imath}$, or $i = \tilde{\imath}$ but $j < \tilde{\jmath}$. For $i = \tilde{\imath}$ we have two subcases, depending on whether $i < g$ or $i = g$. (There are no pairs of different states with $i = \tilde{\imath} > g$; this condition implies $j = \tilde{\jmath} = 0$, by (7). We do not consider $i > \tilde{\imath}$, or $i = \tilde{\imath}$ with $j > \tilde{\jmath}$ either—the roles of $\langle i, j\rangle, \langle \tilde{\imath}, \tilde{\jmath}\rangle$ can be swapped). Let us now define the following binary strings:

$$\begin{aligned}
x_{\langle i,j\rangle} &= 1^i{\cdot}0^j, && \text{for each } \langle i, j\rangle \in Q', \\
y_{\langle i,j\rangle\langle \tilde{\imath},\tilde{\jmath}\rangle} &= 1{\cdot}1^{n-i-1}, && \text{if } 0 \leq i < \tilde{\imath} \leq n-1, \\
y_{\langle i,j\rangle\langle i,\tilde{\jmath}\rangle} &= 0^{d-\tilde{\jmath}}1{\cdot}1^{n-i-1}, && \text{if } 0 \leq i < g \text{ and } 0 \leq j < \tilde{\jmath} \leq d-1, \\
y_{\langle g,j\rangle\langle g,\tilde{\jmath}\rangle} &= 0^{k+1-\tilde{\jmath}}1{\cdot}1^{n-g-1}, && \text{if } g \leq n-1 \text{ and } 0 \leq j < \tilde{\jmath} \leq k.
\end{aligned} \tag{9}$$

It can be seen from (8) that each state $\langle i,j\rangle \in Q'$ is reached by reading $x_{\langle i,j\rangle}$ from the input: $q'_{\rm I} = \langle 0,0\rangle \xrightarrow{1^i} \langle i \bmod n, 0\rangle = \langle i,0\rangle \xrightarrow{0^j} \langle i,j\rangle$.

If $g = n-1$, we get $y_{\langle g,j\rangle\langle g,\tilde{\jmath}\rangle} = 0^{k+1-\tilde{\jmath}}1$, if $g = n$, we do not define $y_{\langle g,j\rangle\langle g,\tilde{\jmath}\rangle}$.

We are now going to show that *(i)* both $x_{\langle i,j\rangle}{\cdot}y_{\langle i,j\rangle\langle\tilde{\imath},\tilde{\jmath}\rangle}$ and $x_{\langle\tilde{\imath},\tilde{\jmath}\rangle}{\cdot}y_{\langle i,j\rangle\langle\tilde{\imath},\tilde{\jmath}\rangle}$ are valid binary images of some strings in Σ^*, i.e., both of them are in $h_\Sigma(\Sigma^*)$, and that *(ii)* the computation of $\mathcal{A}'_{\Sigma,n,g,k}$ on $x_{\langle i,j\rangle}{\cdot}y_{\langle i,j\rangle\langle\tilde{\imath},\tilde{\jmath}\rangle}$ starts in $q'_{\rm I} = \langle 0,0\rangle$ and ends in $\langle 0,0\rangle \in F^{\oplus}$, while the computation on $x_{\langle\tilde{\imath},\tilde{\jmath}\rangle}{\cdot}y_{\langle i,j\rangle\langle\tilde{\imath},\tilde{\jmath}\rangle}$ starts in $q'_{\rm I} = \langle 0,0\rangle$ and ends in some $\langle i',0\rangle \in F^{\ominus}$, with $i' \neq 0$. Taking into account (i), this gives that $x_{\langle i,j\rangle}{\cdot}y_{\langle i,j\rangle\langle\tilde{\imath},\tilde{\jmath}\rangle} \in h_\Sigma(\mathcal{L}(\mathcal{A}_{\Sigma,n,g,k}))$ and $x_{\langle\tilde{\imath},\tilde{\jmath}\rangle}{\cdot}y_{\langle i,j\rangle\langle\tilde{\imath},\tilde{\jmath}\rangle} \in h_\Sigma(\mathcal{L}(\mathcal{A}_{\Sigma,n,g,k})^{\rm C})$. These statements can be shown by analyzing all cases, using (9), (3), and (8) (see also Figs. 2 and 3):

- If $0 \le i < \tilde{\imath} \le n-1$, and hence $0 < \tilde{\imath} - i \le n-1$, with $j,\tilde{\jmath} \in \{0,\ldots,d-1\}$:
  - $x_{\langle i,j\rangle}{\cdot}y_{\langle i,j\rangle\langle\tilde{\imath},\tilde{\jmath}\rangle} = 1^i{\cdot}0^j{\cdot}1{\cdot}1^{n-i-1} = h_\Sigma(a_0^i{\cdot}a_j{\cdot}a_0^{n-i-1})$,
    $\langle 0,0\rangle \xrightarrow{1^i0^j} \langle i,j\rangle \xrightarrow{1} \langle (i+1) \bmod n, 0\rangle \xrightarrow{1^{n-i-1}} \langle 0,0\rangle$,
  - $x_{\langle\tilde{\imath},\tilde{\jmath}\rangle}{\cdot}y_{\langle i,j\rangle\langle\tilde{\imath},\tilde{\jmath}\rangle} = 1^{\tilde{\imath}}{\cdot}0^{\tilde{\jmath}}{\cdot}1{\cdot}1^{n-i-1} = h_\Sigma(a_0^{\tilde{\imath}}{\cdot}a_{\tilde{\jmath}}{\cdot}a_0^{n-i-1})$,
    $\langle 0,0\rangle \xrightarrow{1^{\tilde{\imath}}0^{\tilde{\jmath}}} \langle \tilde{\imath},\tilde{\jmath}\rangle \xrightarrow{1} \langle (\tilde{\imath}+1) \bmod n, 0\rangle \xrightarrow{1^{n-i-1}} \langle \tilde{\imath}-i, 0\rangle \neq \langle 0,0\rangle$.
- If $0 \le i < g$ and $0 \le j < \tilde{\jmath} \le d-1$, and hence $1 \le j + d - \tilde{\jmath} \le d-1$:
  - $x_{\langle i,j\rangle}{\cdot}y_{\langle i,j\rangle\langle i,\tilde{\jmath}\rangle} = 1^i{\cdot}0^j{\cdot}0^{d-\tilde{\jmath}}1{\cdot}1^{n-i-1} = h_\Sigma(a_0^i{\cdot}a_{j+d-\tilde{\jmath}}{\cdot}a_0^{n-i-1})$,
    $\langle 0,0\rangle \xrightarrow{1^i0^j} \langle i,j\rangle \xrightarrow{0^{d-\tilde{\jmath}}} \langle i, j+d-\tilde{\jmath}\rangle \xrightarrow{1} \langle (i+1) \bmod n, 0\rangle \xrightarrow{1^{n-i-1}} \langle 0,0\rangle$,
  - $x_{\langle i,\tilde{\jmath}\rangle}{\cdot}y_{\langle i,j\rangle\langle i,\tilde{\jmath}\rangle} = 1^i{\cdot}0^{\tilde{\jmath}}{\cdot}0^{d-\tilde{\jmath}}1{\cdot}1^{n-i-1} = h_\Sigma(a_0^i{\cdot}a_d{\cdot}a_0^{n-i})$,
    $\langle 0,0\rangle \xrightarrow{1^i0^{\tilde{\jmath}}} \langle i,\tilde{\jmath}\rangle \xrightarrow{0^{d-1-\tilde{\jmath}}} \langle i, d-1\rangle \xrightarrow{0} \langle (i+1) \bmod n, 0\rangle \xrightarrow{1^{n-i}} \langle 1,0\rangle$.
- If $g \le n-1$ and $0 \le j < \tilde{\jmath} \le k$, and hence $1 \le j + k + 1 - \tilde{\jmath} \le k$:
  - $x_{\langle g,j\rangle}{\cdot}y_{\langle g,j\rangle\langle g,\tilde{\jmath}\rangle} = 1^g{\cdot}0^j{\cdot}0^{k+1-\tilde{\jmath}}1{\cdot}1^{n-g-1} = h_\Sigma(a_0^g{\cdot}a_{j+k+1-\tilde{\jmath}}{\cdot}a_0^{n-g-1})$,
    $\langle 0,0\rangle \xrightarrow{1^g0^j} \langle g,j\rangle \xrightarrow{0^{k+1-\tilde{\jmath}}} \langle g, j+k+1-\tilde{\jmath}\rangle \xrightarrow{1} \langle (g+1) \bmod n, 0\rangle \xrightarrow{1^{n-g-1}} \langle 0,0\rangle$,
  - $x_{\langle g,\tilde{\jmath}\rangle}{\cdot}y_{\langle g,j\rangle\langle g,\tilde{\jmath}\rangle} = 1^g{\cdot}0^{\tilde{\jmath}}{\cdot}0^{k+1-\tilde{\jmath}}1{\cdot}1^{n-g-1} = h_\Sigma(a_0^g{\cdot}a_{k+1}{\cdot}a_0^{n-g-1})$,
    $\langle 0,0\rangle \xrightarrow{1^g0^{\tilde{\jmath}}} \langle g,\tilde{\jmath}\rangle \xrightarrow{0^{k-\tilde{\jmath}}} \langle g,k\rangle$, with two different ways to continue:
    if $g \le n-2$, then $\langle g,k\rangle \xrightarrow{0} \langle g+1,0\rangle \xrightarrow{1^{n-g}} \langle (n+1) \bmod n, 0\rangle = \langle 1,0\rangle$,
    if $g = n-1$, then $\langle g,k\rangle \xrightarrow{0} \langle 0,k+1\rangle \xrightarrow{1^{n-g}} \langle (n-g) \bmod n, 0\rangle = \langle 1,0\rangle$.

Summing up, we have constructed m-tuple $X = \langle x_{\langle i,j\rangle}\rangle_{\langle i,j\rangle\in Q'}$ and $\binom{m}{2}$-tuple $Y = \langle y_{\langle i,j\rangle\langle\tilde{\imath},\tilde{\jmath}\rangle}\rangle_{\langle i,j\rangle,\langle\tilde{\imath},\tilde{\jmath}\rangle\in Q',\,\langle i,j\rangle\neq\langle\tilde{\imath},\tilde{\jmath}\rangle}$, where $m = \|Q'\|$, consisting of binary strings such that, for each pair $\langle i,j\rangle \neq \langle\tilde{\imath},\tilde{\jmath}\rangle$,

- both $x_{\langle i,j\rangle}{\cdot}y_{\langle i,j\rangle\langle\tilde{\imath},\tilde{\jmath}\rangle}$ and $x_{\langle\tilde{\imath},\tilde{\jmath}\rangle}{\cdot}y_{\langle i,j\rangle\langle\tilde{\imath},\tilde{\jmath}\rangle}$ are in
  $h_\Sigma(\Sigma^*) = h_\Sigma(\mathcal{L}(\mathcal{A}_{\Sigma,n,g,k})) \cup h_\Sigma(\mathcal{L}(\mathcal{A}_{\Sigma,n,g,k})^{\rm C})$,
- $x_{\langle i,j\rangle}{\cdot}y_{\langle i,j\rangle\langle\tilde{\imath},\tilde{\jmath}\rangle} \in h_\Sigma(\mathcal{L}(\mathcal{A}_{\Sigma,n,g,k}))$ and $x_{\langle\tilde{\imath},\tilde{\jmath}\rangle}{\cdot}y_{\langle i,j\rangle\langle\tilde{\imath},\tilde{\jmath}\rangle} \in h_\Sigma(\mathcal{L}(\mathcal{A}_{\Sigma,n,g,k})^{\rm C})$.

But then, by Theorem 1, any dcDFA solving the binary promise problem $\langle h_\Sigma(\mathcal{L}(\mathcal{A}_{\Sigma,n,g,k})), h_\Sigma(\mathcal{L}(\mathcal{A}_{\Sigma,n,g,k})^{\rm C})\rangle$ must use at least $m = \|Q'\|$ states, which gives that $\mathcal{A}'_{\Sigma,n,g,k}$ is optimal. □

Theorem 5. *For each non-unary input alphabet Σ, there exists a maximal binary prefix code $h\colon\Sigma^*\to\{0,1\}^*$ such that, for each $n \ge 2$ and each value $N \in \{n,\ldots,(\|\Sigma\|-1){\cdot}n\}$, there exists a language $\mathcal{L} \subseteq \Sigma^*$ such that the optimal DFA recognizing $\mathcal{L}$ uses exactly n states and any optimal dcDFA for solving $\langle h(\mathcal{L}), h(\mathcal{L}^{\rm C})\rangle$, the binary promise-problem version of $\mathcal{L}$, uses exactly N states.*

Proof. Let us handle the pathological case of $\Sigma = \{a_0, a_1\}$ first. There are only two maximal prefix codes in this case, both of them are bijections from $\{a_0, a_1\}$ to $\{0, 1\}$, and none of them can change the state complexity of any language. This corresponds to the fact that here $N \in \{n, \ldots, (\|\Sigma\| - 1)\cdot n\} = \{n\}$.

Now, for the given Σ, with $\|\Sigma\| \geq 3$, let us fix $h = h_\Sigma$, as introduced by (3). Next, the witness language $\mathcal{L}$ depends on Σ and on the given values n and N:

First, if $N \leq (\|\Sigma\| - 1)\cdot n - 1 = d\cdot n - 1$, we can take $\mathcal{L} = \mathcal{L}(\mathcal{A}_{\Sigma,n,g,k})$, using the following parameters:

$$g = \lfloor (N - n)/(d - 1) \rfloor, \;\; k = (N - n) \bmod (d - 1).$$

Clearly, $g \leq \lfloor (d\cdot n - 1 - n)/(d - 1) \rfloor = \lfloor n - \frac{1}{d-1} \rfloor = n - 1$ and $k \leq d - 2$. By Lemma 2, $\mathcal{A}_{\Sigma,n,g,k}$ is the optimal DFA for recognizing $\mathcal{L}$, using exactly n states. Similarly, by Lemma 3, $\mathcal{A}'_{\Sigma,n,g,k}$ is optimal dcDFA for solving $\langle h_\Sigma(\mathcal{L}), h_\Sigma(\mathcal{L}^{\mathrm{C}}) \rangle$ and uses exactly $m = d\cdot g + (k + 1) + (n - g - 1)$ states. This gives $m = d\cdot g + (k + 1) + (n - g - 1) = (d - 1)\cdot g + k + n = (d-1)\cdot \lfloor (N - n)/(d - 1) \rfloor + (N - n) \bmod (d - 1) + n = (N - n) + n = N$ states, using the fact that $a\cdot \lfloor b/a \rfloor + b \bmod a = b$.

Second, if $N = (\|\Sigma\| - 1)\cdot n$, take $\mathcal{L} = \mathcal{L}(\mathcal{A}_{\Sigma,n,g,k})$ with $g = n$ and $k = 0$. (Here $\mathcal{A}_{\Sigma,n,g,k}$ does not actually depend on k.) Again, by Lemma 2, $\mathcal{A}_{\Sigma,n,g,k}$ is optimal and uses exactly n states and, by Lemma 3, $\mathcal{A}'_{\Sigma,n,g,k}$ is optimal for solving $\langle h_\Sigma(\mathcal{L}), h_\Sigma(\mathcal{L}^{\mathrm{C}}) \rangle$, this time with exactly $m = (\|\Sigma\| - 1)\cdot n = N$ states. □

5 Concluding Remarks

By a more careful analysis of the construction in Theorem 3, we see that it does not increase the number of accepting or rejecting states. As a direct consequence, if the optimal DFA $\mathcal{A}$ recognizing $\mathcal{L}$ uses $n^\oplus$ accepting and $n^\ominus$ rejecting states (neither of these values can be reduced, since the optimal $\mathcal{A}$ is unique), then any optimal dcDFA $\mathcal{A}'$ solving the binary promise problem $\langle h(\mathcal{L}), h(\mathcal{L}^{\mathrm{C}}) \rangle$ must use at least $n^\oplus$ accepting and at least $n^\ominus$ rejecting states. But all states in $\mathcal{A}'$ that cannot be reached from q'_{I} by reading some $\beta \in h(\Sigma^*)$ can be made neutral (see also (2) in the proof of Theorem 3). This will only change acceptance/rejection to "don't care" answers on *some* inputs not belonging to $h(\Sigma^*)$.

This allows to establish some kind of pseudo-isomorphism between $\mathcal{A}$ and $\mathcal{A}'$. (Proof omitted here due to space constraints, to appear in a journal version.) Namely, there exists a bijective function $t : Q \to F^\oplus \cup F^\ominus$ that maps q_{I} to q'_{I}, F to $F^\oplus$, and $Q \setminus F$ to $F^\ominus$, preserving the machine's transitions, i.e., $t(f(q, a)) = f'^*(t(q), h(a))$, for each $q \in Q$ and $a \in \Sigma$. However, such pseudo-isomorphism does not exclude, for some transition $q \xrightarrow{a} q'$ in $\mathcal{A}$, passing through some state $t(q'') \in F^\oplus \cup F^\ominus$ along the corresponding path $t(q) \xrightarrow{h(a)} t(q')$ in $\mathcal{A}'$—even more than once in the meantime.[3]

[3] This phenomenon can be seen in Fig. 2, where we have "3" $\xrightarrow{a_4}$ "5" for $\mathcal{A}_{\Sigma,n,g,k}$. Since $h(a_4) =$ "00001", this corresponds to $t($"3"$) =$ "30" $\xrightarrow{00001}$ "50" $= t($"5"$)$ for $\mathcal{A}'_{\Sigma,n,g,k}$, passing twice through $t($"4"$) =$ "40".

There are more open questions in the related area than the known answers. As an example, we do not know the cost of binary coded intersection; the same holds for other basic operations with regular languages. It can be expected that answers may depend also on the code h in use, and we expect some anomalies for prefix codes that are not maximal.

References

1. Berstel, J., Perrin, D., Reutenauer, C.: Codes and Automata. Cambridge University Press, Cambridge (2010)
2. Birget, J.-C.: Intersection and union of regular languages and state complexity. Inform. Process. Lett. **43**, 185–190 (1992)
3. Bruyère, V.: Maximal codes with bounded deciphering delay. Theoret. Comput. Sci. **84**, 53–76 (1991)
4. Čevorová, K.: Kleene star on unary regular languages. In: Jurgensen, H., Reis, R. (eds.) DCFS 2013. LNCS, vol. 8031, pp. 277–288. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39310-5_26
5. Geffert, V.: Magic numbers in the state hierarchy of finite automata. Inform. Comput. **205**, 1652–1670 (2007)
6. Goldreich, O.: On promise problems: a survey. In: Goldreich, O., Rosenberg, A.L., Selman, A.L. (eds.) Theoretical Computer Science. LNCS, vol. 3895, pp. 254–290. Springer, Heidelberg (2006). https://doi.org/10.1007/11685654_12
7. Holzer, M., Rauch, C.: The range of state complexities of languages resulting from the cascade product—the unary case (extended abstract). In: Maneth, S. (ed.) CIAA 2021. LNCS, vol. 12803, pp. 90–101. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-79121-6_8
8. Hopcroft, J., Motwani, R., Ullman, J.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley, Boston (2001)
9. Hricko, M., Jirásková, G., Szabari, A.: Union and intersection of regular languages and descriptional complexity. In: Proceedings of Descriptional Complexity of Formal Systems, pp. 170–181. IFIP & University, Milano (2005)
10. Iwama, K., Kambayashi, Y., Takaki, K.: Tight bounds on the number of states of DFA's that are equivalent to n-state NFA's. Theoret. Comput. Sci. **237**, 485–494 (2000)
11. Jirásková, G.: Magic numbers and ternary alphabet. Internat. J. Found. Comput. Sci. **22**, 331–344 (2011)
12. Jirásková, G., Masopust, T.: On a structural property in the state complexity of projected regular languages. Theoret. Comput. Sci. **449**, 93–105 (2012)
13. Moreira, N., Pighizzini, G., Reis, R.: Optimal state reductions of automata with partially specified behaviors. Theoret. Comput. Sci. **658**, 235–245 (2017)
14. Paull, M.C., Unger, S.H.: Minimizing the number of states in incompletely specified sequential switching functions. IRE Trans. Electron. Comput. **3**, 356–367 (1959)
15. Pfleeger, C.P.: State reduction in incompletely specified finite-state machines. IEEE Trans. Comput. **C–22**, 1099–1102 (1973)
16. Rabin, M., Scott, D.: Finite automata and their decision problems. IBM J. Res. Develop. **3**, 114–125 (1959)