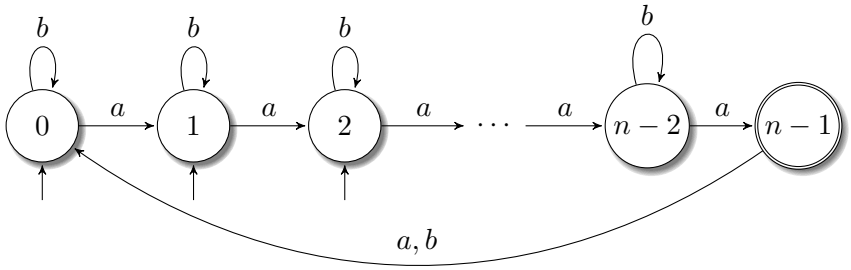
On the Plurality of Initial States

by

Gonçalo Teixeira

A report submitted to the University of Porto for the degree of
Bachelor in Computer Science

Advised By:
Rogério Reis and Nelma Moreira



Faculty of Sciences
University of Porto
January 2026

Contents

List of Figures

List of Tables

List of Algorithms

Resumo

Automatos finitos de multipla entrada (MFAs) são uma generalização dos autómatos finitos deterministicos (DFAs) que permitem a existência de um número arbitrário de estados iniciais. Este relatório extende a pesquisa existente sobre MFAs estabelecendo resultados sobre a sua complexidade de estado operacional e sobre a complexidade computacional dos seus problemas de decisão associados. Em particular, analisámos o custo de várias operações padrão da teoria de linguages quando aplicadas a MFAs e comparamos estes resultados com os resultados obtidos para DFAs e NFAs. Para além disso, também analisámos a complexidade de decidir se a linguagem aceite pelo MFA é vazia ou universal, assim como se a linguagem aceite por um MFA está inclusa na linguagem aceite por outro. Os nossos resultados contribuem para um melhor entendimento do papel do não-determinismo na dificuldade computacional de determinados problemas.

Palavras-Chave: Linguagens Regulares, Teoria de Autómatos, Não-Determinismo Limitado, Complexidade de Estado Operacional.

Abstract

Multi-entry finite automata (MFAs) are a generalization of deterministic finite automata (DFAs) that allow for an arbitrary number of initial states. This report extends existing research on MFAs by establishing results on their operational state complexity and computational complexity of their associated decision problems. In particular, we analyze the cost of the standard language-theoretic operations when performed on MFAs and compare these results with the well known complexities for DFAs and NFAs. Additionally, we also analyze the complexity of deciding the emptiness, universality and inclusion problems for MFAs. Our findings contribute to a deeper understanding of the role that nondeterminism plays in the computational difficulty of certain problems.

Keywords: Regular Languages, Automata Theory, Limited Nondeterminism, Operational State Complexity.

Words are events, they do things, they change things.

Ursula K. Le Guin

Esta é a madrugada que eu esperava
O dia inicial inteiro e limpo
Onde emergimos da noite e do silêncio
E livres habitamos a substância do tempo

Sophia de Mello Breyner Andresen

Chapter 1

Introduction

Regular languages and finite automata have long been central objects of study in theoretical computer science. Their conceptual simplicity, combined with their expressive power and ubiquity , has made them fundamental tools in both theory and practice. At a certain point in time, many believed that, echoing Philip von Jolly's comment about theoretical physics, "almost everything [in this field] is already discovered, and all that remains is to fill a few holes". Fortunatly, just like Max Plank, many computer scientists have ignored this words and, in recent years, interest in automata theory has experienced a revival.

Driven by both the theoretical desire to enhance our understandanding of these fundamental objects of computation and by applications in areas such as formal verification, cognitive science, and bioinformatics, significant attention has been devoted to the study of the descriptional complexity of several systems that describe regular and subregular languages. This study of the *economy of descriptions* [MF71] is of great importance since size matters and leads to more efficient and scalable implementations.

One computational model that has received comparatively little attention is that of *multi-entry finite automata* (MFAs) [GS76, Kap00, HSY01]. MFAs occupy an intermediate design space between the full determinism of deterministic finite automata (DFAs) and the full nondeterminism of nondeterministic finite automata (NFAs). They allow a restricted form of nondeterminism, namely nondeterminism only in the choice of initial states, that yields representations that are possibly exponentially more concise than those of DFAs. Although generally less succinct than NFAs, this restricted form of nondeterminism makes MFAs particularly more well suited to efficient implementation and parallelization.

This report provides a systematic study of the multi-entry finite automata. In Chapter 2, we introduce the necessary background from formal language and automata theory. Chapter 3 formally defines MFAs and develops their fundamental properties. Chapter 4, we investigate the operational state complexity of MFAs, providing several upper bounds and proving their tightness. Finally, Chapter 5 examines the computational complexity of several key decision problems for MFAs, showing that even a limited form of nondeterminism can lead to an increase in difficulty for these problems making a large portion of them computationally intractable.

All automata constructions and algorithms presented in this thesis have been implemented in Python [Pyt25] using the FAdo library [MR]. The complete implementation is publicly available at `https://codeberg.org/tttardigrado/mefa-fado` .

Chapter 2

Preliminaries

In this chapter, we present some basic mathematical notions and definitions from formal language and automata theory that will be used throughout the rest of this report. For more details, we refer the reader to the standard literature [HU79, Koz97, Sha08, Yu97].

2.1 Basic Concepts and Notation

Given two integers $i, j \in \mathbb{Z}$, we use $[i, j] := \{x \in \mathbb{Z} \mid i \leq x \leq j\}$ to denote the closed integer interval from i to j. Moreover, if the lower bound of the interval is zero, we use the simplified notation $[j] := [0, j]$.

To evaluate the asymptotic behavior of a function $f(n) : \mathbb{N} \to \mathbb{N}$ given another function $g(n) : \mathbb{N} \to \mathbb{N}$, we use the Bachmann-Landau notation: [AB09]

$$f(n) = \mathcal{O}(g(n)) \quad \text{if} \quad (\exists c, n_0 \in \mathbb{N})(\forall n > n_0)\ f(n) \leq c \cdot g(n)$$

We also recall the algebraic notion of a semigroup:

Definition 2.1 (Semigroup). *A semigroup is a pair $\langle S, \cdot \rangle$, where S is a set and $\cdot : S \times S \to S$ is an associative binary operation on S:*

$$(\forall a, b, c \in S) \qquad a \cdot (b \cdot c) = (a \cdot b) \cdot c \tag{S1}$$

Two particularly important semigroups are the *permutation* semigroup $\mathcal{S}_X$ of all permutations of a set X paired with functional composition, and the *transformation* semigroup $\mathcal{T}_X$ of all transformations $f : X \to X$ over X paired with functional composition.

2.2 Formal Languages and Words

In the theory of formal languages, the term *alphabet* is used to refer to a finite, non-empty set whose elements are called *symbols* or *letters*. Usually, an alphabet is denoted by Σ. A *word* over Σ is a finite sequence of letters from Σ. This includes the *empty word*, which is denoted by ε. The *concatenation* of two words is the word obtained through the juxtaposition of the symbols of each word. Continuing with the linguistic metaphor, a *language* is a set of words. The set of all words (resp. nonempty words) is denoted by $\Sigma^\star$ (resp. Σ^+) and corresponds to the *free monoid* (resp. *free semigroup*) generated by Σ, considering the concatenation operation.

The *length* of a word $w \in \Sigma^\star$, denoted by $|w|$, corresponds to the number of symbols in w. More formally, the length function $|\cdot| : \Sigma^\star \to \mathbb{N}$ is inductively defined as:

$$\begin{aligned} |\varepsilon| &= 0 \\ |w\sigma| &= |w| + 1 \qquad (\forall w \in \Sigma^\star)(\forall \sigma \in \Sigma) \end{aligned}$$

A similar notation, $|w|_\sigma$, is used to denote the occurrences of the symbol $\sigma \in \Sigma$ in the word $w \in \Sigma^\star$.

Given two words $w, u \in \Sigma^\star$, we say that u is a *prefix* of w if $w = uv$ for some $v \in \Sigma^\star$. Similarly, we say that u is a *suffix* of w if $w = vu$ for some $v \in \Sigma^\star$. In addition, we say that u is a *factor* of w if $w = vuz$ for some $v, z \in \Sigma^\star$. Furthermore, we say that u is a *proper prefix* of w (resp. *proper suffix*, *proper factor*) if u is a prefix of w (resp. suffix, factor) and $u \neq \varepsilon$.

We write w^n for $w \in \Sigma^\star$ and $n \in \mathbb{N}$ to denote the concatenation of n copies of w. Formally, repeated concatenation is inductively defined as:

$$\begin{aligned} w^0 &= \varepsilon \\ w^{n+1} &= w^n w \qquad (\forall w \in \Sigma^\star)(\forall n \in \mathbb{N}) \end{aligned}$$

We write w^R to denote the *reversal* of a word $w \in \Sigma^\star$, i.e., the word written backwards. Formally, it is inductively defined by:

$$\begin{aligned} \varepsilon^R &= \varepsilon \\ (w\sigma)^R &= \sigma w^R \qquad (\forall w \in \Sigma^\star)(\forall \sigma \in \Sigma) \end{aligned}$$

Since languages are sets of words, all the usual set theoretic functions (such as union, intersection and complementation) apply to them. Since their elements are words, we can define some specific language-theoretic operations.

The *concatenation* $L_1 L_2$ of two languages $L_1, L_2 \subseteq \Sigma^\star$ is defined by

$$L_1 L_2 := \{wu \in \Sigma^\star \mid w \in L_1 \wedge u \in L_2\}.$$

Since this operation is associative we use L^n to denote the concatenation of n copies of $L \subseteq \Sigma^\star$:

$$\begin{aligned} L^0 &= \{\varepsilon\} \\ L^{n+1} &= L^n L \qquad (\forall L \subseteq \Sigma^\star)(\forall n \in \mathbb{N}) \end{aligned}$$

The *Kleene star* (resp. *Kleene plus*) $L^\star$ (resp. L^+) of a language $L \subseteq \Sigma^\star$ is the union of all finite powers (resp. nonzero finite powers) of L:

$$L^\star := \bigcup_{n \geq 0} L^n \qquad L^+ := \bigcup_{n \geq 1} L^n$$

The *reversal* L^R of a language $L \subseteq \Sigma^\star$ is the language formed by the reversal of the words that constitute it:

$$L^R := \{w^R \in \Sigma^\star \mid w \in L\}.$$

The *suffix*, *prefix*, and *factor* of a language $L \subseteq \Sigma^\star$, denoted by $\mathrm{Suff}(L)$, $\mathrm{Pref}(L)$, and $\mathrm{Fact}(L)$, respectively, are the languages formed by the suffixes, prefixes and factors of the words in L:

$$\begin{aligned} \mathrm{Suff}(L) &:= \{w \in \Sigma^\star \mid (\exists u \in \Sigma^\star)\ uw \in L\} \\ \mathrm{Pref}(L) &:= \{w \in \Sigma^\star \mid (\exists v \in \Sigma^\star)\ wv \in L\} \\ \mathrm{Fact}(L) &:= \{w \in \Sigma^\star \mid (\exists u, v \in \Sigma^\star)\ uwv \in L\} \end{aligned}$$

A particularly important class of languages, corresponding to the simplest level of the Chomsky Hierarchy [CS63, Cho56], is the class of *Regular languages*: [Yu97]

Definition 2.2 (Regular Languages). *The class RL of Regular Languages is the smallest class that contains $\emptyset$, $\{\varepsilon\}$, $\{\sigma\}$ for $\sigma \in \Sigma$, and is closed under union, concatenation, and Kleene star.*

2.3 Finite Automata

Finite automata form the backbone of theoretical computer science. They are some of simplest and most ubiquitous models of computation with applications in diverse areas such as pattern matching, lexical analysis, biology, linguistics, and cognitive science. Kleene [Kle56] proved that the class of languages that are recognized by finite automata correspond exactly to the Regular Languages, thus making them the model *par excellence* to describe them.

In this section we will define two equivalent classes of finite automata: the deterministic and the nondeterministic finite automata.

Definition 2.3 (NFA). *A* nondeterministic finite automaton *is a* 5*-tuple* $N = \langle Q, \Sigma, \delta, I, F\rangle$*, where* Q *is a finite set of states,* Σ *is the input alphabet,* $\delta : Q \times \Sigma \to 2^Q$ *is the nondeterministic transition function,* $I \subseteq Q$ *is the set of initial states,* $F \subseteq Q$ *is the set of final states.*

The size of an NFA is given by the cardinality of its set of states, i.e. $|N| = |Q|$.

By abuse of notation, the symbol δ is also used to denote the extension of the transition function to act on sets of states and words. This extension can be defined in the natural way:

$$\delta(S, \sigma) = \bigcup_{s \in S} \delta(s, \sigma) \qquad \delta(S, \varepsilon) = S \qquad \delta(S, w\sigma) = \delta(\delta(S, w), \sigma).$$

Additionally, we also define a *path* on an automaton to be a finite sequence of consecutive transitions

$$q_0 \xrightarrow{\sigma_0} q_1 \xrightarrow{\sigma_1} q_2 \xrightarrow{\sigma_2} \cdots \xrightarrow{\sigma_{n-1}} q_n,$$

where $q_{i+1} \in \delta(q_i, \sigma_i)$ for $i \in [n-1]$. A path is said to be *initial* if it starts with an initial state, i.e. $q_0 \in I$, and *final* if it ends in a final state, i.e. $q_n \in F$.

The language recognized by an NFA N is the set of words that label paths that are both initial and final, that is $\mathcal{L}(N) := \{w \in \Sigma^\star \mid \delta(I, w) \cap F \neq \emptyset\}$. Two NFAs are equivalent if they recognize the same language.

Given a state $q \in Q$, we can also define the *right language* of q, denoted by $\mathcal{L}_q^{\rightarrow}(N)$, as the set of words that label final paths that start in q, i.e. $\mathcal{L}_q^{\rightarrow}(N) := \{w \in \Sigma^\star \mid \delta(q, w) \cap F \neq \emptyset\}$. Analogously, we define the *left language* of q, denoted by $\mathcal{L}_q^{\leftarrow}(N)$, as the set of words that label initial paths that end in q, i.e. $\mathcal{L}_q^{\leftarrow}(N) := \{w \in \Sigma^\star \mid q \in \delta(I, w)\}$.

A state $q \in Q$ is *reachable* if there is an initial path that ends in q, i.e. $\mathcal{L}_q^{\leftarrow}(N) \neq \emptyset$. Similarly, a state $q \in Q$ is *co-reachable* if there is a final path that starts in q, i.e. $\mathcal{L}_q^{\rightarrow}(N) \neq \emptyset$. An NFA is *reachable* (resp. *co-reachable*) if all of its states are reachable (resp. co-reachable). If an NFA is both reachable and co-reachable we say that it is *trim*.

Given an NFA $N = \langle Q, \Sigma, \delta, I, F\rangle$, its *reverse automaton* is defined to be the NFA $N^R = \langle Q, \Sigma, \delta^R, F, I\rangle$ obtained by reversing all arrows and swapping the initial and final states. The language recognized by the reverse NFA is the reversal of the language of the original NFA, i.e. $\mathcal{L}\big(N^R\big) = \mathcal{L}(N)^R$.

An NFA is said to be *minimal* if it has the smallest number of states out of all NFAs to which it is equivalent.

Moreover, an NFA is said to be *deterministic* if it has a single initial state, that is $|I| = 1$, and $|\delta(q, \sigma)| \leq 1$ for all states $q \in Q$ and symbols $\sigma \in \Sigma$. If the transition function is defined for all states and symbols, the automaton is said to be *complete*. This allows us to define the class of *deterministic finite automata* in the following way:

Definition 2.4 (DFA). *A* deterministic finite automaton *is a* 5*-tuple* $A = \langle Q, \Sigma, \delta, q_0, F\rangle$*, where* Q *is a finite set of states,* Σ *is the input alphabet,* $\delta : Q \times \Sigma \to Q$ *is the transition function* $q_0 \in Q$ *is the initial state,* $F \subseteq Q$ *is the set of final states.*

In a similar fashion to what we did with NFAs, we extend the transition function to words. The definitions of size of a DFA, language recognized by a DFA, right and left languages of a state, reachable, co-reachable, trim, and minimal DFAs mirrors the definitions of the nondeterministic versions of those concepts.

By disregarding the initial and final states of a DFA and focusing solely on its transition structure, we obtain what is known as a semiautomaton [Pin97]. Semiautomata play a central role in algebraic approaches to automata theory, as their transition structure induces a finite semigroup whose algebraic properties reflect important aspects of the languages recognized by the original DFA.

Definition 2.5 (Transition Semigroup). *Let $A = \langle Q, \Sigma, \delta \rangle$ be a semiautomaton. For each symbol $\sigma \in \Sigma$, let the transformation $\delta_\sigma : Q \to Q$ be defined as*

$$\delta_\sigma(q) = \delta(q, \sigma).$$

The transition semigroup of A, denoted $\mathcal{T}(A)$, is the finite semigroup generated by the set $\{\delta_\sigma \mid \sigma \in \Sigma\}$ with the semigroup operation given by function composition.

Note that $\mathcal{T}(A)$ is always a subsemigroup of the transformation semigroup on the set of states. If the transition semigroup $\mathcal{T}(A)$ and the transformation semigroup $\mathcal{T}_Q$ are isomorphic, we say that A is *functionally complete*.

2.3.1 Determinization

Although NFAs offer a higher degree of succinctness when describing regular languages, nondeterminism introduces difficulties in terms of implementation and performance. In a paper that later earned them the Turing Award, Rabin and Scott [RS59] proved that every NFA can be converted into an equivalent DFA using the following construction:

Theorem 2.6 (Powerset Construction). *Let $N = \langle Q, \Sigma, \delta, I, F \rangle$ be an NFA. The powerset DFA $\mathcal{D}(N) = \langle 2^Q, \Sigma, \delta', I, F' \rangle$, where $F' = \{S \in 2^Q \mid S \cap F \neq \emptyset\}$ and*

$$\delta'(S, \sigma) = \bigcup_{s \in S} \delta(s, \sigma),$$

is equivalent to N, i.e., $\mathcal{L}(N) = \mathcal{L}(\mathcal{D}(N))$.

It follows from the definition of the Powerset Construction that the process of determinization may incur, in the worst case, in an exponential blow-up in the number of states. Later, Meyer and Fischer [MF71], Moore [Moo71], and Lupanov [Lup63] independently proved that this upper bound is tight by constructing families of n-state NFAs whose equivalent DFAs require 2^n states. This is regarded as the first major result in descriptional complexity.

2.3.2 Minimization

The problem of computing reduced – and ideally minimal – automata for a given language is of central importance in both theory and practice. For DFAs, this problem is well understood. Its origins trace back to the seminal work of Moore [Moo56] in the 1950s, where he presented a minimization algorithm that runs in $\mathcal{O}(n^2)$ time. This result was later improved on by Hopcroft [Hop71], whose algorithm achieves a runtime of $\mathcal{O}(n \log n)$. Moreover, as a consequence of the Myhill-Nerode Theorem [Myh57, Ner58], the minimal DFA recognizing a regular language is unique up to isomorphism.

In contrast, the situation for NFAs is considerably more complex. Unlike the deterministic case, minimal NFAs for a given language are not necessarily unique. More importantly, computing a minimal NFA is PSPACE-COMPLETE [MS72].

Although Gruber and Holzer [GH06] have proved that even determining the size of a minimal NFA

is computationally difficult, Glaister and Shallit [GS96], and Birget [Bir92, Bir93] have presented a technique that allows us to prove lower bounds on the size of NFAs. This technique is derived from communication complexity [AUY83], and is commonly referred to as the *fooling set* method:

Definition 2.7 (Fooling Set)**.** *A set of pairs of strings $\mathcal{F} = \{\langle x_i, y_i\rangle \mid i \in [1, n]\}$ is called a* fooling set *for a language $L \subseteq \Sigma^\star$ if for each $i, j \in [1, n]$:*

$$x_i y_i \in L \tag{F1}$$
$$i \neq j \implies x_i y_j \notin L \vee x_j y_i \notin L \tag{F2}$$

Lemma 2.8. *Let $\mathcal{F}$ be a fooling set for a regular language $L \subseteq \Sigma^\star$. Then every NFA for L has at least $|\mathcal{F}|$ states.*

Unfortunately, Glaister and Shallit [GS96] have pointed out that this bound is not always tight, and indeed, that the gap can be arbitrarily large. Despite this theoretical limitation, Hospodár et al. [HJM18] have studied this method and empirically proved that fooling sets can often be used to provide tight lower bounds for the operations we are interested in.

2.3.3 State Complexity

The most natural way to measure the complexity of regular languages is through the number of states required by a given computational model (DFAs, NFAs, etc.) to recognize them. This leads to the definition of the *state complexity* $\mathrm{sc}(L)$ of a regular language L as the number of states in the minimum DFA recognizing L. Similarly, the *nondeterministic state complexity* of L, denoted $\mathrm{nsc}(L)$, is defined as the number of states in a minimal NFA that recognizes it. It is worth noting that, since our definitions allow DFAs to be incomplete and NFAs to have multiple initial states, the (nondeterministic) state complexities may differ by 1 from the values obtained when requiring complete DFAs and a unique initial state.

Naturally, one can ask about the worst-case complexity resulting from applying a regularity-preserving operation to a set of regular languages, as a function of the complexities of the operands. This is known as *operational state complexity*:

Definition 2.9 (Operational State Complexity)**.** *Let f be a k-ary operation on regular languages. The state complexity of f is given by the function*

$$(n_1, \cdots, n_k) \mapsto \max\{\ \mathrm{sc}(f(L_1, \cdots, L_k)) \mid (\forall i \in [1, k])\ \mathrm{sc}(L_i) \leq n_i\ \}.$$

Nondeterministic operational state complexity is defined analogously.

The first results on operational state complexity were obtained by the Soviet mathematicians Mirkin [Mir66] and Maslov [Mas70] in the 1960s and 1970s. However, the topic only became a prolific area of research in theoretical computer science after 1994, when a paper by Yu et al. [YZS94] reignited the interest in the field.

Operational state complexity results are proved in two steps:

1. First, we prove the upper bound. This can be done by providing an algorithm that converts the input automata into an automaton that recognizes the resulting language, whose size is a function of the sizes of the input automata;
2. Then, we prove that the bound is tight. This can be done by providing an infinite family of *witness* languages $(L_n \mid n \geq k)$, for some small integer $k \in \mathbb{N}$, that reach the upper bound.

For a comprehensive overview of results in operational state complexity, we refer the reader to the survey by Gao et al. [GMRY16].

Chapter 3

Multi-entry Finite Automata

As the saying goes, "with great power there must also come great responsibility," [LD62] and nondeterminism is no exception. As discussed in Chapter 2, nondeterminism can yield exponentially more succinct descriptions of regular languages. However, this is accompanied by significant complexity drawbacks, since fundamental operations such as minimization become computationally intractable.

In this section, we introduce and study an intermediate computational model that lies strictly between the full determinism of DFAs and the full nondeterminism of NFAs. This model, called a *multi-entry finite automaton* (MFA), restricts nondeterminism to the choice of the initial state, while maintaining the deterministic behavior of the transition function. Formally:

Definition 3.1 (MFA)**.** *A multi-entry finite automaton is a 5-tuple $M = \langle Q, \Sigma, \delta, I, F\rangle$ where Q is a finite set of states, Σ is an alphabet, $\delta : Q \times \Sigma \to Q$ is a deterministic transition function, $I \subseteq Q$ is a set of initial states, and $F \subseteq Q$ is a set of final states.*

From the definition, it follows immediately that MFAs recognize exactly the class of regular languages. Indeed, every MFA is a special case of a NFA, and, conversely, every DFA can be viewed as an MFA with a single initial state. This observation yields the following characterization of the MFA-recognizable languages:

Theorem 3.2. *A language $L \subseteq \Sigma^\star$ is recognized by an MFA iff L is regular.*

Intuitively, an MFA can be viewed as a collection of DFAs that share the same transition structure and set of final states, but differ in their initial states. This makes MFAs exceptionally well suited for parallel and distributed computation, as noted in, for example, [GK74, BBMR24, BBCRM26, Pol07], and shown in Fig. 3.1. The following theorem proves that this intuition is correct.

Theorem 3.3 (Decomposition)**.** *Let $M = \langle Q, \Sigma, \delta, I, F\rangle$ be an MFA. For each $i \in I$, let $M_i = \langle Q, \Sigma, \delta, i, F\rangle$ denote the DFA obtained from M by taking i as the unique initial state. Then,*

$$\mathcal{L}(M) = \bigcup_{i \in I} \mathcal{L}(M_i).$$

Proof. Let $w \in \Sigma^\star$. By definition, $w \in \mathcal{L}(M)$ iff there exists an initial state $i \in I$ such that $\delta(i, w) \in F$. This holds iff there exists $i \in I$ such that $w \in \mathcal{L}(M_i)$. Therefore, $w \in \mathcal{L}(M)$ iff $w \in \bigcup_{i \in I} \mathcal{L}(M_i)$. □

Moreover, we note that MFAs generalize a class of automata previously studied by Gill and Kou [GK74, GS76, VG79]. In that model, all states were required to be initial (i.e., $I = Q$), which restricts the recognized languages to the class of *suffix-closed* regular languages. By allowing an arbitrary subset of states to serve as initial states, we get a more general model of computation that still retains the interesting properties of the earlier model, such as parallelization and limited nondeterminism.

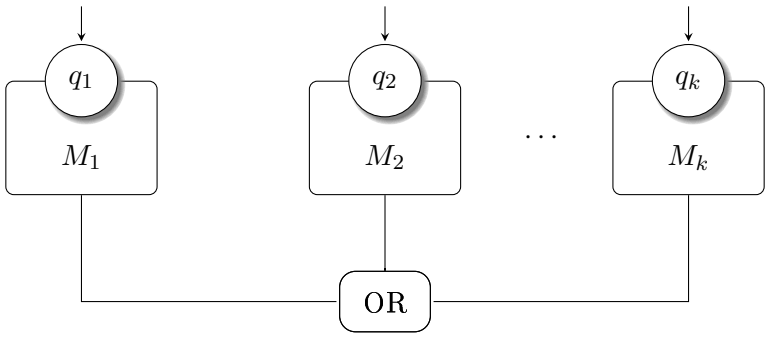


Figure 3.1: Paralel implementation of an MFA using a OR-gate

3.1 Determinization

As noted in Chapter 2, NFAs can be exponentially smaller than their minimal equivalent DFAs. It is natural to ask whether the same holds for MFAs. Kappes [Kap00] answered this question affirmatively by presenting a modified version of the powerset construction of Rabin and Scott [RS59]. Rather than computing the entire powerset of the state space, this construction generates only those subsets whose cardinality does not exceed the number of initial states, yielding an upper bound of $\sum_{i=0}^{|I|} \binom{n}{i}$ states. Note that in the worst case, where all states are initial, this bound reduces to 2^n, matching the upper bound for NFAs. This bound had already been shown by Gill and Kou [GK74] and proved to be tight independently by Veloso and Gill [VG79] and by Galil and Simon [GS76].

Moreover, Kappes [Kap00] proved that this general bound is tight for MFAs with any number of initial states, over an alphabet of a growing size. Subsequently, Holzer et al. [HSY01] strengthened this result by proving tightness over a binary alphabet using the family of MFAs depicted in Fig. 3.2.

Theorem 3.4. *Let $M_k = \langle Q, \Sigma, \delta, I, F\rangle$ be an n-state, k-entry MFA. The DFA $\mathcal{D}(M) = \langle Q', \Sigma, \delta', I, F'\rangle$, where*

$$Q' = \{S \in 2^Q \mid |S| \leq k\}$$
$$F' = \{S \in Q' \mid S \cap F \neq \emptyset\}$$
$$\delta'(S, \sigma) = \bigcup_{s \in S} \delta(s, \sigma),$$

is equivalent to M, i.e. $\mathcal{L}(M) = \mathcal{L}(\mathcal{D}(M))$.

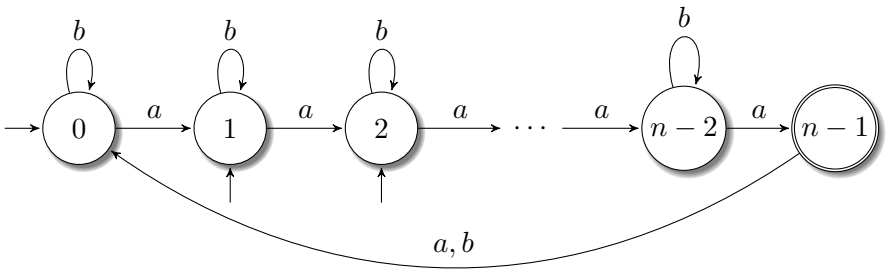


Figure 3.2: n-state, k-initial MFA whose determination achieves a maximal blow-up

Another natural question to ask is whether MFAs offer any gain in economy of description over NFAs. Perhaps surprisingly, Kappes [Kap00] showed that NFAs can also be exponentially more succinct than any equivalent MFA. To prove this, Kappes provided an algorithm that converts any n-state NFA with finite branching k, into an equivalent kn-state MFA with k initial states. Here branching refers to the minimal amount of nondeterminism that an NFA N needs in order to recognize any word in

$\mathcal{L}(N)$. Since converting an NFA with unbounded branching to one with finite branching can incur in an exponential blowup in the number of states [GKW90], the overall conversion from NFA to MFA is, in the worst case, exponential.

3.2 Minimization

We now consider the problem of finding state-minimal MFAs. In their seminal work, Gill and Kou [GK74] established that minimal MFAs are not unique in general, by considering MFAs where all states are initial. Later, Holzer et al. [HSY01] strengthened this result by demonstrating non-uniqueness for MFAs with any number of initial states $|I| \geq 2$ even over a unary alphabet.

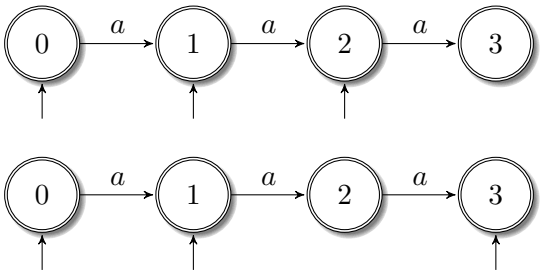


Figure 3.3: Two minimal 4-state MFAs recognizing the same language

In the same paper, Holzer et al. [HSY01] also proved that MFA minimization is PSPACE-COMPLETE under logspace many-one reductions. Subsequently, Malcher [Mal04] refined their result by showing that when the number of initial states is fixed, the problem becomes NP-COMPLETE. This mirrors the behavior of the DFA-INTERSECTION-EMPTINESS problem, used in the reduction of [HSY01], where fixing the number of DFAs similarly reduces complexity from PSPACE-COMPLETE to NP-COMPLETE.

This shows that even the most limited amounts of nondeterminism lead to the non-uniqueness of a minimal automaton and to the computational intractability of the minimization proble. Thus, these results provide compelling evidence that both uniqueness and tractability of minimization problems are strongly related to determinism.

Chapter 4

MFA Operational State Complexity

In this chapter we explore the operational state complexity of multi-entry finite automata. Following the definitions of Section 2.3.3, the *multi-entry state complexity* of a regular language $L \subseteq \Sigma^\star$, denoted by $\mathrm{msc}(L)$, is defined to be the number of states of a minimal MFA recognizing L. The *multi-entry operational state complexity* of an operation is defined analogously to Definition 2.9.

To prove that a given MFA is minimal, we make use of the following corollary that we derive from Lemma 2.8 via the fact that any MFA is also an NFA:

Corollary 4.1. *Let $\mathcal{F}$ be a fooling set for a regular language $L \subseteq \Sigma^\star$. Then every MFA for L has at least $|\mathcal{F}|$ states.*

As an example of how Corollary 4.1 can be used, let us prove the following technical lemma that we will make use of in the proofs of Theorem 4.5 and Theorem 4.8:

Lemma 4.2. *Let $L_{na} = \{w \in \{a,b\}^\star \mid |w|_a \in [n]\}$. Then, $\mathrm{msc}(L_{na}) = n+1$.*

Proof. Clearly, L_{na} is recognized by the $(n+1)$-state automaton of Fig. 4.1.

Consider the following set of $n+1$ pairs:

$$\mathcal{A} = \{\langle a^i, a^{n-i}\rangle \mid i \in [n]\}.$$

We will prove that $\mathcal{A}$ is a fooling set for L_{na}:

(F1) For any i, $|a^i a^{n-i}|_a = n$. Therefore, the word belongs to L_{na}.
(F2) If $j < i$, then $|a^i a^{n-j}|_a = n + i - j > n$ and the word does not belong to L_{na}.

This shows that $\mathcal{A}$ is a fooling set for L_{na}, which, by Corollary 4.1, proves the result. □

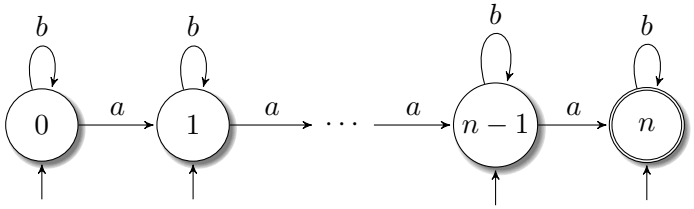


Figure 4.1: $(n+1)$-state MFA recognizing L_{na}

4.1 Boolean Operations

We begin our investigation into the realm of multi-entry operational state complexity by considering the three most common Boolean operations on automata: *intersection*, *union*, and complementation.

4.1.1 Intersection

Let us start by considering the *intersection* operation. It is well known [YZS94, HK03] that both the deterministic and nondeterministic state complexities of intersection achieve the same value of mn. This suggests that the multi-entry state complexity should also be the same, and Theorem 4.5 proves this claim.

Lemma 4.3. *Let $L_1, L_2 \subseteq \Sigma^\star$ be two languages with $\mathrm{msc}(L_1) = n$ and $\mathrm{msc}(L_2) = m$. Then, nm states are sufficient for an MFA to recognize $L_1 \cap L_2$.*

Proof. Let $M_1 = \langle Q_1, \Sigma, \delta_1, I_1, F_1\rangle$ and $M_2 = \langle Q_2, \Sigma, \delta_2, I_2, F_2\rangle$ be two MFAs that recognize L_1 and L_2, respectively. We define a product automaton $M = \langle Q_1 \times Q_2, \Sigma, \delta, I_1 \times I_2, F_1 \times F_2\rangle$, where

$$\delta(\langle p, q\rangle, \sigma) = \langle \delta_1(p, \sigma), \delta_2(q, \sigma)\rangle,$$

if both $\delta_1(p, \sigma)$ and $\delta_2(q, \sigma)$ are well-defined. Since both δ_1 and δ_2 are deterministic, δ is also deterministic.

It is easy to see that $\mathcal{L}(M) = L_1 \cap L_2$, since $\delta(\langle i_1, i_2\rangle, w) \in F_1 \times F_2$ iff $\delta_1(i_1, w) \in F_1$ and $\delta_2(i_2, w) \in F_2$, for any word $w \in \Sigma^\star$ and initial states $i_1 \in I_1$ and $i_2 \in I_2$. This implies that the construction is correct, thus establishing an upper bound for the multi-entry state complexity of intersection. □

Lemma 4.4. *Let $L_1, L_2 \subseteq \Sigma^\star$ be two languages with $\mathrm{msc}(L_1) = n$ and $\mathrm{msc}(L_2) = m$. Then, nm states are necessary, in the worst case, for an MFA to recognize $L_1 \cap L_2$.*

Proof. To prove that the bound of Lemma 4.3 is tight, consider the languages $L_{na} = \{w \in \Sigma^\star \mid |w|_a \in [n]\}$ and $L_{mb} = \{w \in \Sigma^\star \mid |w|_b \in [m]\}$ of Lemma 4.2. We want to prove that the set of pairs

$$\mathcal{A} = \{\langle a^i b^j, a^{n-i} b^{m-j}\rangle \mid i \in [n], j \in [m]\}$$

is a fooling set for $L_{na} \cap L_{mb}$:

(F1) For any $\langle i, j\rangle$, the word $a^i b^j a^{n-i} b^{m-j}$ contains n occurrences of a and m occurrences of b. Thus, the word belongs to $L_{na} \cap L_{mb}$.

(F2) Let $\langle i, j\rangle \neq \langle k, \ell\rangle$:

- If $k < i$, then $|a^i b^j a^{n-k} b^{m-\ell}|_a = n + (i - k) > n$, so the word does not belong to $L_{na} \cap L_{mb}$.
- If $i = k$ and $\ell < j$, then $|a^i b^j a^{n-k} b^{m-\ell}|_b = m + (j - \ell) > m$, and the word is not in $L_{na} \cap L_{mb}$.

Therefore, $\mathcal{A}$ is a fooling set of size $(m + 1)(n + 1)$ for $L_{na} \cap L_{mb}$, and, by Corollary 4.1, any MFA recognizing it requires at least $(m + 1)(n + 1)$ states. This completes the proof. □

From Lemmas 4.3 to 4.4, we get the following theorem:

Theorem 4.5 (Intersection)**.** *Let $L_1, L_2 \subseteq \Sigma^\star$ be two languages with $\mathrm{msc}(L_1) = n$ and $\mathrm{msc}(L_2) = m$. Then, $\mathrm{msc}(L_1 \cap L_2) \leq mn$, and this bound is tight for $|\Sigma| \geq 2$.*

4.1.2 Union

The case for the *union* operation is slightly different. The deterministic state complexity achieves the same bound as intersection [YZS94], but nondeterminism allows us to reduce this value to $n + m + 1$ in the case that we enforce a single initial state, and $n + m$ in the general case [HSY01, HJM18]. This raises the following question: is initial state nondeterminism enough to lower this bound? Theorem 4.8 answers this question positively.

Lemma 4.6. *Let $L_1, L_2 \subseteq \Sigma^\star$ be two languages with $\mathrm{msc}(L_1) = n$ and $\mathrm{msc}(L_2) = m$. Then, $n+m$ states are sufficient for an MFA to recognize $L_1 \cup L_2$.*

Proof. Let $M_1 = \langle Q_1, \Sigma, \delta_1, I_1, F_1 \rangle$ and $M_2 = \langle Q_2, \Sigma, \delta_2, I_2, F_2 \rangle$ be MFAs that recognize L_1 and L_2, respectively. Without loss of generality, we assume Q_1 and Q_2 are disjoint. We define a union automaton $M = \langle Q_1 \cup Q_2, \Sigma, \delta, I_1 \cup I_2, F_1 \cup F_2 \rangle$, where

$$\delta(q,\sigma) = \begin{cases} \delta_1(q,\sigma) & \text{if } q \in Q_1 \text{ and } \delta_1(q,\sigma) \text{ is well-defined} \\ \delta_2(q,\sigma) & \text{if } q \in Q_2 \text{ and } \delta_2(q,\sigma) \text{ is well-defined} \\ \bot & \text{otherwise} \end{cases}$$

To prove the upper bound, we want to show that $\mathcal{L}(M) = L_1 \cup L_2$.

($\Longrightarrow$) Let $w \in \mathcal{L}(M)$. By definition, $\delta(i, w) \in F_1 + F_2$ for some $i \in I_1 + I_2$. There are two possibilities:

1. If $i \in I_1$, then $\delta(i, w) \in F_1$, so $w \in L_1$,
2. If $i \in I_2$, then $\delta(i, w) \in F_2$, so $w \in L_2$.

Hence, $w \in L_1 \cup L_2$.

($\Longleftarrow$) Let $w \in L_1 \cup L_2$. If $w \in L_1$, then there exists an initial state $i \in I_1$ such that $\delta_1(i, w) \in F_1$. Similarly, if $w \in L_2$, then there exists an initial state $i \in I_2$ such that $\delta_2(i, w) \in F_2$. In either case, we have that $\delta(i, w) \in F_1 + F_2$ for some $i \in I_1 + I_2$, and thus $w \in \mathcal{L}(M)$. □

Lemma 4.7. *Let $L_1, L_2 \subseteq \Sigma^\star$ be two languages with $\mathrm{msc}(L_1) = n$ and $\mathrm{msc}(L_2) = m$. Then, $n+m$ states are necessary, in the worst case, for an MFA to recognize $L_1 \cup L_2$.*

Proof. Consider again the witness families L_{na} and L_{ma} used in the proof of Lemma 4.4. Let

$$\mathcal{A} = \{\langle a^i, a^{n-i} \rangle \mid i \in [1, n]\} \cup \{\langle a^n, a^n \rangle\}$$
$$\mathcal{B} = \{\langle b^i, b^{m-i} \rangle \mid i \in [1, m]\} \cup \{\langle b^m, b^m \rangle\}$$

be two fooling sets of size n and m for L_{na} and L_{mb}, respectively. Since $\mathcal{A}$ and $\mathcal{B}$ are disjoint, their union $\mathcal{A} \cup \mathcal{B}$ is a fooling set of size $n+m$ for $L_{na} \cup L_{mb}$. By Corollary 4.1, we conclude that the minimal MFA recognizing $L_{na} \cup L_{mb}$ has, at least, $n + m$ states, thus proving that the bound is tight. □

Lemmas 4.6 to 4.7 establish the following result:

Theorem 4.8 (Union)**.** *Let $L_1, L_2 \subseteq \Sigma^\star$ be two languages with $\mathrm{msc}(L_1) = n$ and $\mathrm{msc}(L_2) = m$. Then, $\mathrm{msc}(L_1 \cup L_2) \leq m + n$, and this bound is tight for $|\Sigma| \geq 2$.*

4.1.3 Complementation

The last Boolean operation we consider is *complementation*. This operation is an entirely different beast: contrary to the other two, it is easy on DFAs, requiring no increase in the number of states, but can be exponentially expensive for NFAs, requiring 2^n states in the worst case [Jir05]. This raises the following question: is this exponential blow-up an inherent consequence of any form of nondeterminism, or is nondeterminism in the *transition function* itself necessary to achieve this bound?

We begin by establishing the same upper bound as in the general case for NFAs.

Lemma 4.9. *Let $L \subseteq \Sigma^\star$ be a language with $\mathrm{msc}(L) = n$. Then, 2^n states are sufficient for an MFA to recognize $\overline{L}$.*

Proof. Let M be an n-state MFA recognizing L. By applying the determinization construction of Theorem 3.4, we obtain an equivalent DFA $\mathcal{D}(M)$ with at most 2^n states. Complementation of a DFA is achieved by swapping accepting and non-accepting states. The resulting DFA $\overline{\mathcal{D}(M)}$ therefore recognizes $\overline{L}$ and has at most 2^n states. □

We now show that this bound is tight. To this end, consider the automata $\mathcal{M}_n = \langle Q, \Sigma, \delta, Q, F\rangle$, of Fig. 4.2, where $Q = [n-1]$, $\Sigma = \{a, b, c, d\}$, $F = \{n-1\}$, and

$$\begin{aligned} \delta(q, a) &= q + 1 \bmod n && (\forall q \in [n-1]),\\ \delta(0, b) = 1, \quad \delta(1, b) = 0, \quad \delta(q, b) &= q && (\forall q \in [2, n-1]),\\ \delta(n-1, c) = 0, \quad \delta(q, c) &= q && (\forall q \in [0, n-2]),\\ \delta(0, d) &= 0. \end{aligned}$$

Let $\mathcal{M}'_n$ denote the restriction of $\mathcal{M}_n$ to the alphabet $\Sigma' = \{a, b, c\}$. From an algebraic perspective, the input symbols act as follows: a induces the cyclic shift $(0, 1, \ldots, n-1)$, b induces the transposition $(0, 1)$, and c induces the singular transformation $\binom{n-1}{0}$. Consequently, $\mathcal{M}'_n$ is a variation of Brzozowski's universal witness automaton [Brz12] in which all states are initial. In particular, the associated semiautomaton coincides with that of the universal witnesses, and its transition semigroup $\mathcal{T}(\mathcal{M}'_n)$ is therefore *functionally complete* as a consequence of a result of Piccard [Pic35] and Dénes [Dén66].

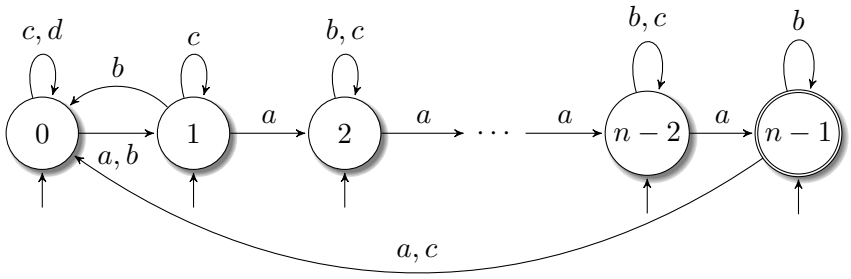


Figure 4.2: Variation on Brzozowski's Universal Witness automaton $\mathcal{M}_n$

To establish tightness, we use the following lemma due to Hospodár et al. [HJM18].

Lemma 4.10. *Let N be an n-state NFA in which every subset of the state set is reachable as well as co-reachable. Then, every NFA for the complement of the language $\mathcal{L}(A)$ has at least 2^n states.*

In order to apply this lemma, we start by proving that all 2^n subsets of states of $\mathcal{M}_n$ are reachable, and then that they are co-reachable.

Lemma 4.11. *If a subset $S \subseteq Q$ is reachable, then all other subsets of the same cardinality are also reachable.*

Proof. Fix a reachable subset $S \subseteq Q$. Since S is reachable, there is a word $w \in \Sigma^\star$ such that $\delta(Q, w) = S$. Let $T \subseteq Q$ be another subset with the same cardinality. There is a permutation $\pi \in \mathcal{S}_Q$ that maps S to T. Since $\mathcal{M}_n$ is functionally complete, there is a word $v \in \Sigma^\star$ whose action on the set of states corresponds to π. Thus, $\delta(Q, wv) = T$, and T is reachable. □

Lemma 4.12. *If a subset $S \subseteq Q$ with $|S| \geq 2$ is reachable, then all subsets of cardinality $|S| - 1$ are also reachable.*

Proof. Fix a reachable subset $S \subseteq Q$ with at least two elements $p, q \in S$. Since S is reachable, there is a word $w \in \Sigma^\star$ such that $\delta(Q, w) = S$. There is a permutation $\pi \in \mathcal{S}_Q$ that maps p to 0, q to $n-1$. From functional completeness, we can establish the existence of a word $v \in \Sigma^\star$ whose action on the set of states corresponds to π. Since $c \in \Sigma$ maps both 0 and 1 to the same state, the set $T = \delta(Q, wvc)$ is reachable and has cardinality $|S| - 1$.

Since a set of cardinality $|S| - 1$ is reachable, then, by Lemma 4.11, all other subsets of the same cardinality are reachable. □

Now we are ready to prove the following:

Lemma 4.13. *All 2^n subsets of the state set of $\mathcal{M}_n$ are reachable.*

Proof. Since all states are initial, the full set Q is reachable. By repeated application of Lemma 4.12, all non-empty subsets are reachable. Finally, because $\mathcal{M}_n$ is incomplete, the empty set is also reachable. □

We now establish co-reachability.

Lemma 4.14. *All 2^n subsets of the state set of $\mathcal{M}_n$ are co-reachable.*

Proof. Salomaa et al. [SWY04] showed that any automaton recognizing a non-empty, non-universal language over an alphabet with at least three symbols has 2^n co-reachable subsets of states, provided its associated semiautomaton is functionally complete. Since $\mathcal{M}'_n$ satisfies these conditions and co-reachability is independent of the choice of initial states and the addition of a transition by $d \in \Sigma$, all 2^n subsets of the state set of $\mathcal{M}_n$ are co-reachable. □

An important consequence of this is the fact that $\mathcal{M}_n$ is minimal.

Lemma 4.15. *The MFA $\mathcal{M}_n$ is minimal*

Proof. Hospodár et al. [HJM18] showed that a language L admits a fooling set of size n iff there exists an NFA recognizing L in which every singleton set of states is both reachable and co-reachable. Since every subset of states of $\mathcal{M}_n$ is both reachable and co-reachable, it follows in particular that all singleton subsets satisfy this property. Hence, the language recognized by $\mathcal{M}_n$ admits a fooling set of size n. By Corollary 4.1, this implies that $\mathcal{M}_n$ is minimal. □

Moreover, since all 2^n subsets are both reachable and co-reachable, it follows from Lemma 4.10 that the bound is tight.

Theorem 4.16 (Complementation)**.** *Let $L \subseteq \Sigma^\star$ be a language with* $\mathrm{msc}(L) = n$*. Then,* $\mathrm{msc}(\overline{L}) \leq 2^n$*, and this bound is tight for $|\Sigma| \geq 4$.*

4.2 Closure Operations

The state complexities of the *suffix*, *prefix*, and *factor* closure operations have been investigated in detail by Brzozowski et al. [BJZ14] and Kao et al. [KRS09]. They established that, in the worst case, the prefix, suffix, and factor closure operations require $2^n - 1$, n, and 2^{n-1} states, respectively, when applied to an n-state DFA. In this section we show that initial state nondeterminism alone is sufficient to reduce all these bounds to n.

We start by establishing the upper bounds for all three operations.

Lemma 4.17. *Let $L \subseteq \Sigma^\star$ be a language with* $\mathrm{msc}(L) = n$*. Then, n states are sufficient for an MFA to recognize* $\mathrm{Suff}(L)$*.*

Proof. Let $M = \langle Q, \Sigma, \delta, I, F\rangle$ be an n-state MFA that recognizes L. We define $M_S = \langle Q, \Sigma, \delta, I', F\rangle$ to be the automaton constructed from M by making every reachable state initial. Let us start by proving that $\mathcal{L}(M_S) = \mathrm{Suff}(L)$.

($\Longrightarrow$) Let $w \in \mathcal{L}(M_S)$. Then, there exists an initial state $i' \in I'$ such that $\delta(i', w) \in F$. Since every initial state is reachable in M, there is a word $u \in \Sigma^\star$ such that $\delta(i, u) = i'$, thus $\delta(i, uw) \in F$. This means that $uw \in L$, and so $w \in \mathrm{Suff}(L)$.

($\Longleftarrow$) Conversely, let $w \in \mathrm{Suff}(L)$. By definition, there exists a word $u \in \Sigma^\star$ such that $uw \in L$, i.e., $\delta(i, uw) \in F$ for some initial state $i \in I$. Since all reachable state in M are initial in M_S, $\delta(i, u) \in I'$, thus proving that $w \in \mathcal{L}(M)$. □

Lemma 4.18. *Let $L \subseteq \Sigma^\star$ be a language with $\mathrm{msc}(L) = n$. Then, n states are sufficient for an MFA to recognize $\mathrm{Pref}(L)$.*

Proof. Let $M = \langle Q, \Sigma, \delta, I, F\rangle$ be an n-state MFA that recognizes L. We define $M_P = \langle Q, \Sigma, \delta, I, F'\rangle$ to be the automaton constructed from M by making every co-reachable state final. Let us start by proving that $\mathcal{L}(M_P) = \mathrm{Pref}(L)$.

($\Longrightarrow$) Let $w \in \mathcal{L}(M_P)$. By definition, there is an initial state $i \in I$ such that $\delta(i, w) \in F'$. Since every final state of M_S is co-reachable in M, there is a word $u \in \Sigma^\star$ such that $\delta(i, wu) \in F$. This means that $wu \in L$ and, therefore, that $w \in \mathrm{Pref}(L)$.

($\Longleftarrow$) Conversely, let $w \in \mathrm{Pref}(L)$. This means that there exists a word $u \in \Sigma^\star$ such that $wu \in L$, i.e., $\delta(i, wu) \in F$ for some initial state $i \in I$. Since all co-reachable states in M are final in M_S, $\delta(i, w) \in F'$, thus proving that $w \in \mathcal{L}(M)$. □

Lemma 4.19. *Let $L \subseteq \Sigma^\star$ be a language with $\mathrm{msc}(L) = n$. Then, n states are sufficient for an MFA to recognize $\mathrm{Fact}(L)$.*

Proof. Let $M = \langle Q, \Sigma, \delta, I, F\rangle$ be an n-state MFA that recognizes L. We define $M_F = \langle Q, \Sigma, \delta, I', F'\rangle$ to be the automaton constructed from M by making every reachable state initial and every co-reachable state final. Let us start by proving that $\mathcal{L}(M_F) = \mathrm{Fact}(L)$.

($\Longrightarrow$) Let $w \in \mathcal{L}(M_F)$. By definition, there is an initial state $i' \in I'$ such that $\delta(i, w) \in F'$. Since every initial state of M_F is reachable in M and every final state of M_F is co-reachable in M, there are words $u, v \in \Sigma^\star$ such that $\delta(i, uwv) \in F$ for some initial state $i \in I$. This means that $uwv \in L$ and, therefore, that $w \in \mathrm{Fact}(L)$.

($\Longleftarrow$) Conversely, let $w \in \mathrm{Fact}(L)$. This means that there are words $u, v \in \Sigma^\star$ such that $uwv \in L$, i.e., $\delta(i, uwv) \in F$ for some initial state $i \in I$. Since all reachable states in M are initial in M_F, $\delta(i, u) \in I'$. Similarly, since all co-reachable states in M are final in M_F, $\delta(i, uw) \in F'$, thus proving that $w \in \mathcal{L}(M)$. □

The following lemma proves the tightness of the three upper bounds:

Lemma 4.20. *Let $L \subseteq \Sigma^\star$ be a language with $\mathrm{msc}(L) = n$. Then, n states are necessary, in the worst case, for an MFA to recognize $\mathrm{Suff}(L)$, $\mathrm{Pref}(L)$, and $\mathrm{Fact}(L)$.*

Proof. Let $w \in \Sigma^\star$ be a word of length $n - 1$, and let M_w be the linear n-state MFA recognizing the singleton language $\{w\}$. By the constructions of Lemma 4.17, Lemma 4.18, and Lemma 4.19, we obtain n-state MFAs M_S, M_P, and M_F recognizing $\mathrm{Suff}(\{w\})$, $\mathrm{Pref}(\{w\})$, and $\mathrm{Fact}(\{w\})$, respectively.

We now prove that these automata are minimal. We start by noting that the four languages considered are finite and that the largest word that belongs to each of them is w. Suppose, for the sake of contradiction, that there are MFAs recognizing these with less than n states. By the pigeonhole principle, any accepting computation of w must visit some states more than once. This implies the existence of a loop which we can iterate over and, thus, accept words of an arbitrary length. This contradicts finiteness and, thus, such an automaton cannot exist. Hence, the automata M_w, M_S, M_P, and M_F are minimal, and the bound of n states is tight. □

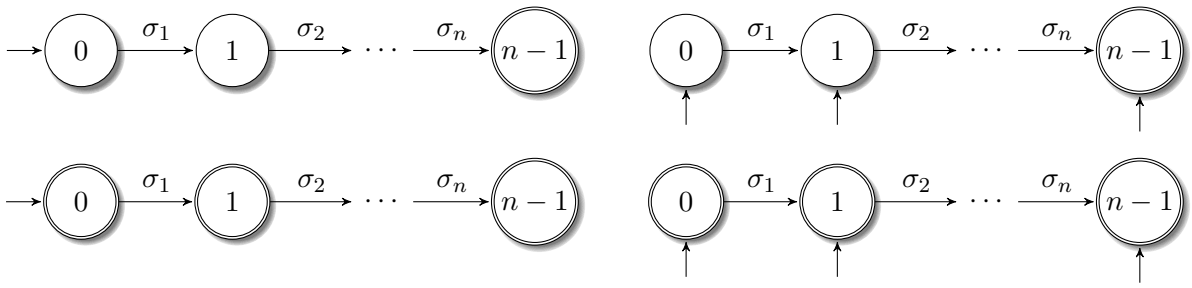


Figure 4.3: n-state automata M_w, M_S, M_P, M_F, from left to right, top to bottom.

From Lemmas 4.17 to 4.20, we can directly conclude that:

Theorem 4.21. *Let* $L \subseteq \Sigma^\star$ *be a language with* $\mathrm{msc}(L) = n$. *Then,* $\mathrm{msc}(\mathrm{Suff}(L)) \leq n$, $\mathrm{msc}(\mathrm{Pref}(L)) \leq n$ *and* $\mathrm{msc}(\mathrm{Fact}(L)) \leq n$, *and these bounds are tight for* $|\Sigma| \geq 2$.

Chapter 5

Decision Problems on MFA

The computational complexity of decision problems for finite automata has been extensively studied [MS72, HK11]. For DFAs, many classical decision problems have efficient algorithms, whereas for NFAs the presence of unrestricted nondeterminism often leads to many of those problems becoming computationally intractable in practice. In this chapter we investigate the complexity of several decision problems for MFAs.

5.1 Emptiness

We begin by analyzing the complexity of deciding whether the language recognized by an MFA is empty. Contrary to all other decision problems here considered, we show that initial state nondeterminism does not affect the complexity of this problem.

Problem 5.1 (MFA-Emptiness)**.**
Input: *An MFA* $M = \langle Q, \Sigma, \delta, I, F \rangle$.
Question: *Does* $\mathcal{L}(M) = \emptyset$*?*

Theorem 5.2. MFA-Emptiness *is* NL-complete.

Proof. Since, by the Immerman-Szelepcsényi theorem [Imm88, Sze88], NL = coNL it suffices to show that the complementary problem, i.e., deciding whether $\mathcal{L}(M) \neq \emptyset$, is in NL.

Let $M = \langle Q, \Sigma, \delta, I, F \rangle$ be an n-state MFA. Notice that $\mathcal{L}(M) \neq \emptyset$ iff there exists a word labeling a path that is both initial and final. A nondeterministic logspace machine can verify this property as follows: it first guesses an initial state, then repeatedly guesses outgoing transitions, and accepts as soon as a final state is reached. To ensure termination, the machine rejects if more than n transitions are taken. This upper bound on the length of the accepting path is sound due to the fact that any accepting path of a longer length must contain a loop, and, thus, can be shortened by removing said loop.

The algorithm stores only the current state and a counter, requiring only $\mathcal{O}(\log n)$ space. Hence, the problem is in NL.

Hardness follows from the fact that the specialization of this problem to DFAs, DFA-Emptiness, is well known to be NL-complete [Jon75]. Therefore, MFA-Emptiness is NL-hard.

From membership and hardness we can conclude that MFA-Emptiness is NL-complete. □

5.2 Universality

We now turn to the universality problem, which asks whether a given automaton accepts all possible words over its alphabet. Contrary to the previous problem, initial state nondeterminism is already sufficient to make this problem computationally intractable.

Problem 5.3 (MFA-Universality).
Input: *An MFA* $M = \langle Q, \Sigma, \delta, I, F \rangle$.
Question: *Does* $\mathcal{L}(M) = \Sigma^\star$*?*

Theorem 5.4. MFA-Universality *is* PSpace-complete.

Proof. We first establish membership in PSpace. The corresponding problem for NFAs, NFA-Universality, is known to be PSpace-complete [MS72]. Since MFAs form a subclass of NFAs, the same upper bound applies. For completeness, an explicit algorithm for MFAs is presented in Appendix A.

To prove hardness, we give a reduction from the DFA-Intersection-Emptiness problem (DIE), which is known to be PSpace-complete. Let $\langle A_1, \ldots, A_k \rangle$ be an instance of DIE. By De Morgan's laws, we have

Then, we prove hardness via a reduction from the DFA-Intersection-Emptiness problem (DIE), which is known to be PSpace-complete [Koz77]. Let $\langle A_1, \cdots, A_k \rangle$ be an instance of the DIE problem. By De Morgan's laws, we have

$$\bigcap_{i=1}^{k} \mathcal{L}(M_i) = \emptyset \quad \text{iff} \quad \bigcup_{i=1}^{k} \overline{\mathcal{L}(M_i)} = \Sigma^\star.$$

By Theorem 3.3, the union of the complemented DFAs can be represented as an MFA accepting the same language. Since complementation and union of DFAs can be performed in polynomial time on the size of the input, MFA-Universality is PSpace-hard.

From membership and hardness we can conclude that MFA-Universality is PSpace-complete. □

5.3 Inclusion

Finally, we consider the language inclusion problem for MFAs, which generalizes universality and is central in many verification tasks. Similarly to the universality problem, initial state nondeterminism makes this problem much harder.

Problem 5.5 (MFA-Inclusion).
Input: *Two MFA* $M_1 = \langle Q_1, \Sigma, \delta_1, I_1, F_1 \rangle$ *and* $M_2 = \langle Q_2, \Sigma, \delta_2, I_2, F_2 \rangle$.
Question: *Does* $\mathcal{L}(M_1) \subseteq \mathcal{L}(M_2)$*?*

Theorem 5.6. MFA-Inclusion *is* PSpace-complete

Proof. Membership in PSpace follows from the fact that the corresponding problem for NFAs, NFA-Inclusion, is PSpace-complete [MS72]. As before, since MFAs are a special case of NFAs, the same upper bound applies. An explicit algorithm is given in Appendix A.

For hardness, we provide a polynomial-time reduction from the MFA-Universality problem, which we have already established as PSpace-complete in Theorem 5.4. Given an MFA M over Σ, we construct an instance $\langle U, M \rangle$ of MFA-Inclusion, where U is a fixed MFA such that $\mathcal{L}(U) = \Sigma^\star$. Clearly,

$$\mathcal{L}(U) \subseteq \mathcal{L}(M) \quad \text{iff} \quad \mathcal{L}(M) = \Sigma^\star.$$

The construction of U can be performed in constant time. Hence, MFA-Inclusion is PSpace-hard.

We conclude that MFA-Inclusion is PSpace-complete. □

Chapter 6

Conclusion

In this report, we investigated *multi-entry finite automata* (MFAs), a relatively underexplored model of computation characterized by a very limited form of nondeterminism. MFAs are particularly appealing due to their balance between expressiveness and ease of implementation, making them a promising candidate for efficient implementation and parallelization.

The first part of this work surveyed and systematized the existing results on MFAs, establishing the necessary foundations for our subsequent analysis. Building on this background knowledge, we studied the operational state complexity of the standard language-theoretic operations when applied to MFAs. Let $L_1, L_2 \subseteq \Sigma^\star$ be two regular languages recognized by the MFAs $M_1 = \langle Q_1, \Sigma, \delta_1, I_1, F_1 \rangle$ and $M_2 = \langle Q_2, \Sigma, \delta_2, I_2, F_2 \rangle$, respectively. Let $|Q_1| = n$, $|Q_2| = m$, $|F_1| = f$, $|F_1 \setminus I_1| = \ell$ and $|I_2| = k$. Our results are summarized in Table 6.1.

	DFA	**MFA**	**NFA**
$L_1 \cap L_2$	nm	nm	nm
$L_1 \cup L_2$	nm	$n+m$	$n+m$
$\overline{L}$	n	2^n	2^n
$\mathrm{Suff}(L)$	$2^n - 1$	n	n
$\mathrm{Pref}(L)$	n	n	n
$\mathrm{Fact}(L)$	2^{n-1}	n	n
$L_1 L_2$	$n2^m - f2^{m-1}$	$\leq n2^m - f2^{m-k}$	$n+m$
L^R	2^n	$\leq 2^n$	n
$L^\star$	$2^{n-1} - 2^{n-l-1}$	$\leq 2^n + 1$	$n+1$
L^+	$2^{n-1} - 2^{n-l-1} - 1$	$\leq 2^n$	n

Table 6.1: Summary of operational state complexity results

In particular, we would like to point out that the limited kind of nondeterminism available to MFAs suffices to improve the operational state complexity bounds for the union, and the suffix and factor closure operations, while preserving the bounds for intersection and prefix closure. Unfortunately, even this small amount of nondeterminism seems to be sufficient to make complementation as difficult as for the general case for NFAs. For the operations of concatenation, reversal, Kleene star and Kleene plus we were only able to prove an upper bound. Although we couldn't prove their tightness we conjecture that these bounds are tight and, at least, significantly worse than their NFA counterparts.

In the final part of the report, we examined the computational complexity of fundamental decision problems for MFAs. As summarized in Table 6.2, while emptiness remains NL-complete, problems such as universality, inclusion, and minimization become PSpace-complete. This sharp increase in complexity suggests that even the most limited amount of nondeterminism can dramatically jeopardize the computational tractability of these problems.

As future work, we would like to improve our operational state complexity results. Namely, we would like to prove tight lower bounds for the concatenation, reversal, Kleene star and Kleene plus operations and to prove the existence (or non-existence) of binary or ternary witnesses for complementation.

	DFA	**MFA**	**NFA**
Emptiness	NL-complete	NL-complete	NL-complete
Universality	NL-complete	PSpace-complete	PSpace-complete
Inclusion	P	PSpace-complete	PSpace-complete
Minimization	P	PSpace-complete	PSpace-complete

Table 6.2: Summary of the complexity of decision problems for MFAs

Furthermore, extending our results to the unary case may reveal significantly different behavior, as is often observed in other automata models [GMRY16]. Finally, developing uniform random generation methods for MFAs, similar to the ones that exist for other classes of automata such as ICDFAs [AMR07], FIFAs [FMR18], and WDFAs [BCK+24], would provide valuable tools for experimental and average-case analysis.

Overall, this work contributes to a deeper understanding of the structural and complexity-theoretic properties of MFAs, positioning them as an intermediate model between the fully deterministic DFAs and the fully nondeterministic NFAs.

Bibliography

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 1 edition, 2009.

[AMR07] Marco Almeida, Nelma Moreira, and Rogério Reis. Enumeration and generation with a string automata representation. *Theoretical Computer Science*, 387(2):93–102, 2007. doi:10.1016/j.tcs.2007.07.029.

[AUY83] Alfred Aho, Jeffrey Ullman, and Mihalis Yannakakis. On notions of information transfer in vlsi circuits. In *Symposium on Theory of Computing*, page 133–139, 1983. doi:10.1145/800061.808742.

[BBCRM26] Angelo Borsotti, Luca Breveglieri, Stefano Crespi Reghizzi, and Angelo Morzenti. Multi-entry dfa with reduced initial states to speedup parallel recognition. In *Implementation and Application of Automata*, pages 55–72, 2026. doi:10.1007/978-3-032-02602-6_5.

[BBMR24] Angelo Borsotti, Luca Breveglieri, Angelo Morzenti, and Stefano Crespi Reghizzi. Minimizing speculation overhead in a parallel recognizer for regular texts. In *Symposium on Principles and Practice of Parallel Programming*, 2024. doi:10.1145/3710848.3710866.

[BCK+24] Ruben Becker, Davide Cenzato, Sung-Hwan Kim, Bojana Kodric, Riccardo Maso, and Nicola Prezza. Random Wheeler Automata. In *Symposium on Combinatorial Pattern Matching*, pages 5:1–5:15, 2024. doi:10.4230/LIPIcs.CPM.2024.5.

[Bir92] Jean-Camille Birget. Intersection and union of regular languages and state complexity. *Information Processing Letters*, 43(4):185–190, 1992. doi:10.1016/0020-0190(92)90198-5.

[Bir93] Jean-Camille Birget. Partial orders on words, minimal elements of regular languages, and state complexity. *Theoretical Computer Science*, 119(2):267–291, 1993. doi:10.1016/0304-3975(93)90160-U.

[BJZ14] Janusz Brzozowski, Galina Jirásková, and Chenglong Zou. Quotient complexity of closed languages. *Theory of Computing Systems*, 54(2):277–292, 2014. doi:10.1007/s00224-013-9515-7.

[Brz12] Janusz Brzozowski. In search of most complex regular languages. In *Implementation and Application of Automata*, pages 5–24, 2012. doi:10.1007/978-3-642-31606-7_2.

[Cho56] Noam Chomsky. Three models for the description of language. *IRE Transactions on Information Theory*, 2(3):113–124, 1956. doi:10.1109/TIT.1956.1056813.

[CS63] Noam Chomsky and Marcel-Paul Schützenberger. The algebraic theory of context-free languages. *Studies in Logic and the Foundations of Mathematics*, 35:118–161, 1963. doi:10.1016/S0049-237X(08)72023-8.

[Dén66] János Dénes. On transformations, transformation-semigroups and graphs. In *Theory of Graphs: Proceedings of the Colloquium on Graph Theory held at Tihany*, pages 65–75, 1966.

[FMR18] Miguel Ferreira, Nelma Moreira, and Rogério Reis. Forward injective finite automata: Exact and random generation of nonisomorphic nfas. In *Descriptional Complexity of Formal Systems*, page 88–100, 2018. doi:10.1007/978-3-319-94631-3_8.

[GH06] Hermann Gruber and Markus Holzer. Finding lower bounds for nondeterministic state complexity is hard. In *Developments in Language Theory*, pages 363–374, 2006. doi:10.1007/11779148_33.

[GK74] Arthur Gill and Lawrence Kou. Multiple-entry finite automata. *Journal of Computer and System Sciences*, 9(1):1–19, 1974. doi:10.1016/S0022-0000(74)80034-6.

[GKW90] Jonathan Goldstine, C.M.R. Kintala, and Detlef Wotschke. On measuring nondeterminism in regular languages. *Information and Computation*, 86(2):179–194, 1990. doi:10.1016/0890-5401(90)90053-K.

[GMRY16] Yuan Gao, Nelma Moreira, Rogério Reis, and Sheng Yu. A survey on operational state complexity. *Journal of Automata, Languages and Combinatorics*, 21(4):251–310, 2016. doi:10.25596/jalc-2016-251.

[GS76] Zvi Galil and Janos Simon. A note on multiple-entry finite automata. *Journal of Computer and System Sciences*, 12(3):350–351, 1976. doi:10.1016/S0022-0000(76)80006-2.

[GS96] Ian Glaister and Jeffrey Shallit. A lower bound technique for the size of nondeterministic finite automata. *Information Processing Letters*, 59(2):75–77, 1996. doi:10.1016/0020-0190(96)00095-6.

[HJM18] Michal Hospodár, Galina Jirásková, and Peter Mlynárčik. A survey on fooling sets as effective tools for lower bounds on nondeterministic complexity. In *Adventures Between Lower Bounds and Higher Altitudes: Essays Dedicated to Juraj Hromkovič on the Occasion of His 60th Birthday*, pages 17–32. 2018. doi:10.1007/978-3-319-98355-4_2.

[HK03] Markus Holzer and Martin Kutrib. State complexity of basic operations on nondeterministic finite automata. In *Implementation and Application of Automata*, pages 148–157, 2003. doi:10.1007/3-540-44977-9_14.

[HK11] Markus Holzer and Martin Kutrib. Descriptional and computational complexity of finite automata—a survey. *Information and Computation*, 209(3):456–470, 2011. doi:10.1016/j.ic.2010.11.013.

[Hop71] John Hopcroft. An n log n algorithm for minimizing states in a finite automaton. In *Theory of Machines and Computations*, pages 189–196. 1971. doi:10.1016/B978-0-12-417750-5.50022-1.

[HSY01] Markus Holzer, Kai Salomaa, and Sheng Yu. On the state complexity of k-entry deterministic finite automata. *Journal of Automata, Languages and Combinatorics*, 6(4):453–466, 2001. doi:10.25596/jalc-2001-453.

[HU79] John Hopcroft and Jeffrey Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1 edition, 1979.

[Imm88] Neil Immerman. Nondeterministic space is closed under complementation. *SIAM Journal on Computing*, 17(5):935–938, 1988. doi:10.1137/0217058.

[Jir05] Galina Jirásková. State complexity of some operations on binary regular languages. *Theoretical Computer Science*, 330(2):287–298, 2005. doi:10.1016/j.tcs.2004.04.011.

[Jon75] Neil D. Jones. Space-bounded reducibility among combinatorial problems. *Journal of Computer and System Sciences*, 11(1):68–85, 1975. doi:10.1016/S0022-0000(75)80050-X.

[Kap00] Martin Kappes. Descriptional complexity of deterministic finite automata with multiple initial states. *Journal of Automata, Languages and Combinatorics*, 5(3):269–278, 2000. doi:10.25596/jalc-2000-269.

[Kle56] Stephen Cole Kleene. Representation of events in nerve nets and finite automata. In *Automata Studies*, pages 3–42. 1956. doi:10.1515/9781400882618-002.

[Koz77] Dexter Kozen. Lower bounds for natural proof systems. In *Symposium on Foundations of Computer Science*, pages 254–266, 1977. doi:10.1109/SFCS.1977.16.

[Koz97] Dexter Kozen. *Automata and Computability*. Springer, 1 edition, 1997.

[KRS09] Jui-Yi Kao, Narad Rampersad, and Jeffrey Shallit. On nfas where all states are final, initial, or both. *Theoretical Computer Science*, 410(47):5010–5021, 2009. doi:10.1016/j.tcs.2009.07.049.

[LD62] Stan Lee and Steve Ditko. Amazing fantasy #15, August 1962.

[Lup63] Oleg Lupanov. A comparison of two types of finite sources. *Problemy Kibernetiki*, 9:321–326, 1963.

[Mal04] Andreas Malcher. Minimizing finite automata is computationally hard. *Theoretical Computer Science*, 327(3):375–390, 2004. doi:10.1016/j.tcs.2004.03.070.

[Mas70] Abraham Maslov. Estimates of the number of states of finite automata. *Soviet Mathematics*, 11:1373–1375, 1970.

[MF71] Albert Meyer and Michael Fischer. Economy of description by automata, grammars, and formal systems. In *Symposium on Switching and Automata Theory*, pages 188–191, 1971. doi:10.1109/SWAT.1971.11.

[Mir66] Boris Mirkin. On dual automata. *Cybernetics*, 2:6–9, 1966. doi:10.1007/BF01072247.

[Moo56] Edward Moore. Gedanken-experiments on sequential machines. pages 129–153. 1956. doi:10.1515/9781400882618-006.

[Moo71] F.R. Moore. On the bounds for state-set size in the proofs of equivalence between deterministic, nondeterministic, and two-way finite automata. *IEEE Transactions on Computers*, C-20(10):1211–1214, 1971. doi:10.1109/T-C.1971.223108.

[MR] Nelma Moreira and Rogério Reis. Fado: Tools for formal languages manipulation. URL: https://fado.dcc.fc.up.pt/.

[MS72] Albert Meyer and Larry Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *Symposium on Switching and Automata Theory*, pages 125–129, 1972. doi:10.1109/SWAT.1972.29.

[Myh57] John Myhill. Finite automata and the representation of events. In *WADD Technical Report*, volume 57-624. 1957.

[Ner58] Anil Nerode. Linear automaton transformations. *Proceedings of the American Mathematical Society*, 9(4):541–544, 1958. doi:10.2307/2033204.

[Pic35] Sophie Piccard. Sur les fonctions définies dans les ensembles finis quelconques. *Fundamenta Mathematicae*, 24(1):298–301, 1935. doi:10.4064/fm-24-1-298-301.

[Pin97] Jean-Éric Pin. Syntactic semigroups. In *Handbook of Formal Languages*, pages 679–746. 1997. doi:10.1007/978-3-642-59136-5_10.

[Pol07] Libor Polák. Remarks on multiple entry deterministic finite automata. *Journal of Automata, Languages and Combinatorics*, 12(1):279–288, 2007.

[Pyt25] Python Software Foundation. Python programming language, 2025. URL: https://www.python.org.

[RS59] Michael Rabin and Dana Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3(2):114–125, 1959. doi:10.1147/rd.32.0114.

[Sav70] Walter Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2):177–192, 1970. doi:10.1016/S0022-0000(70)80006-X.

[Sha08] Jeffrey Shallit. *A Second Course in Formal Languages and Automata Theory*. Cambridge University Press, 1 edition, 2008.

[SWY04] Arto Salomaa, Derick Wood, and Sheng Yu. On the state complexity of reversals of regular languages. *Theoretical Computer Science*, 320(2):315–329, 2004. doi:10.1016/j.tcs.2004.02.032.

[Sze88] Róbert Szelepcsényi. The method of forced enumeration for on-line recognition of languages. *Acta Informatica*, 26:279–284, 1988. doi:10.1007/BF00299636.

[VG79] Paulo Veloso and Arthur Gill. Some remarks on multiple-entry finite automata. *Journal of Computer and System Sciences*, 18(3):304–306, 1979. doi:10.1016/0022-0000(79)90038-2.

[Yu97] Sheng Yu. Regular languages. In *Handbook of Formal Languages*, pages 41–110. 1997. doi:10.1007/978-3-642-59136-5_2.

[YZS94] Sheng Yu, Qingyu Zhuang, and Kai Salomaa. The state complexities of some basic operations on regular languages. *Theoretical Computer Science*, 125(2):315–328, 1994. doi:10.1016/0304-3975(92)00011-F.

Appendix A

Decision Algorithms

In this appendix, we present explicit algorithms for the decision problems studied in Chapter 5. The purpose is twofold: first to make the complexity arguments fully explicit, and, second, to provide concrete algorithms formulations that may serve as a basis for further research.

While the algorithms we present are asymptotically optimal in the worst case, prior work on NFAs indicates that significant improvements may be possible on average. This has been a prolific line of research, largely motivated by applications in model checking and formal verification.

Throughout, we give algorithms for the complements of the problems under consideration. Since nondeterministic and deterministic space complexity classes are closed under complement by the Immerman-Szelepcsényi Theorem [Imm88, Sze88], this suffices to establish membership in the stated classes.

A.1 Emptiness

We first consider the emptiness problem. Rather than deciding emptiness directly, we give a nondeterministic procedure for MFA-NonEmptiness, i.e., determining whether $\mathcal{L}(M) \neq \emptyset$.

The algorithm nondeterministically guesses a path starting from an initial state and checks whether an accepting state is reachable within $|Q|$ steps. This bound suffices because any accepting path of a greater length must revisit a state, and the resulting cycle can be removed without affecting acceptance.

Input: MFA $M = \langle Q, \Sigma, \delta, I, F \rangle$
Output: Does $\mathcal{L}(M) \neq \emptyset$?
guess $q \in I$;
repeat $|Q|$ **times**
  if $q \in F$ **then**
    return true;
  end
  guess $\sigma \in \Sigma$;
  $q \leftarrow \delta(q, \sigma)$;
end
return false;

Algorithm 1: Decision procedure for MFA-NonEmptyness

The algorithm stores only the current state and counter, requiring $\mathcal{O}(\log |Q|)$ space. Hence, MFA-Emptiness belongs to NL.

A.2 Universality

We next consider the universality problem. We present an algorithm for MFA-NonUniversality, i.e., deciding whether $\mathcal{L}(M) \neq \Sigma^\star$.

The algorithm performs an on-the-fly subset construction. If M is not universal, then there exists

a word whose corresponding reachable subset of states contains no accepting state. The algorithm nondeterministically explores reachable subsets and checks for such a rejecting subset. The search depth is bounded by $2^{|Q|}$.

Input: MFA $M = \langle Q, \Sigma, \delta, I, F \rangle$
Output: Does $\mathcal{L}(M) \neq \Sigma^\star$?
$S = I$;
repeat $2^{|Q|}$ **times**
  if $S \cap F = \emptyset$ **then**
    return true;
  end
  guess $\sigma \in \Sigma$;
  $S \leftarrow \delta(S, \sigma)$;
end
return false;

Algorithm 2: Decision procedure for MFA-NonUniversality

The subset S of active states requires $\mathcal{O}(|Q|)$ bits, as does the counter. Therefore, MFA-Universality is in NPSpace, and by Savitch's Theorem [Sav70], also in PSpace.

A.3 Inclusion

Finally, we consider the inclusion problem for two MFAs. We give a procedure for MFA-NonInclusion, i.e., deciding whether $\mathcal{L}(M_1) \not\subseteq \mathcal{L}(M_2)$.

Notice that $\mathcal{L}(M_1) \not\subseteq \mathcal{L}(M_2)$ iff $\mathcal{L}(M_1) \cap \overline{\mathcal{L}(M_2)} \neq \emptyset$. Accordingly, the algorithm nondeterministically guesses a word that is accepted by M_1 while being rejected along all possible runs of M_2. This is achieved by simultaneously tracking a single state of M_1 and the subset of possible states of M_2. The search depth is bounded by $|Q_1| \cdot 2^{|Q_2|}$.

Input: Two MFAs $M_1 = \langle Q_1, \Sigma, \delta_1, I_1, F_1 \rangle$ and $M_2 = \langle Q_2, \Sigma, \delta_2, I_2, F_2 \rangle$
Output: Does $\mathcal{L}(M_1) \not\subseteq \mathcal{L}(M_2)$?
guess $q \in I_1$;
$S = I_2$;
repeat $|Q_1| \cdot 2^{|Q_2|}$ **times**
  if $q \in F_1 \wedge S \cap F_2 = \emptyset$ **then**
    return true;
  end
  guess $\sigma \in \Sigma$;
  $q \leftarrow \delta_1(q, \sigma)$;
  $S \leftarrow \delta_2(S, \sigma)$;
end
return false;

Algorithm 3: Decision procedure for MFA-NonInclusion

The algorithm stores a state of M_1, a subset of states of M_2, and a counter, using $\mathcal{O}(\log |Q_1| + |Q_2|)$ space overall. Thus, MFA-Inclusion belongs to NPSpace, and hence, by Savitch's Theorem [Sav70], to PSpace.

Appendix B

Upper Bounds for Other Operations

In this Appendix, we establish upper bounds for the multi-entry state complexity of the concatenation, reversal and star operations. Unfortunately, we were not able to prove the tightness of the upper bounds, but conjecture that they are tight due to the fact that the known constructions for NFAs make extensive use of nondeterminism.

B.1 Concatenation

We start by considering the concatenation operation.

Theorem B.1 (Concatenation). *Let $L_1, L_2 \subseteq \Sigma^\star$ be regular languages such that L_1 is recognized by an n-state MFA with f final states and L_2 is recognized by an m-state MFA with k initial states. Then, $n2^m - f2^{m-k}$ states are sufficient for an MFA to recognize L_1L_2.*

Proof. Let $M_1 = \langle Q_1, \Sigma, \delta_1, I_1, F_1\rangle$ and $M_2 = \langle Q_2, \Sigma, \delta_2, I_2, F_2\rangle$ be two MFA such that $\mathcal{L}(M_1) = L_1$, $\mathcal{L}(M_2) = L_2$, $|Q_1| = n$, $|Q_2| = m$, $|F_1| = f$, and $|I_2| = k$.

We construct an MFA recognizing L_1L_2. Let $M = \langle Q, \Sigma, \delta, I, F\rangle$, where

$$
\begin{aligned}
Q &= Q_1 \times 2^{Q_2} \setminus F_1 \times 2^{Q_2 \setminus I_2} \\
I &= \{\langle i, \emptyset\rangle \in Q \mid i \in I_1 \setminus F_1\} \cup \{\langle i, I_2\rangle \in Q \mid i \in I_1 \cap F_1\} \\
F &= \{\langle q, S\rangle \in Q \mid S \cap F_2 \neq \emptyset\}
\end{aligned}
$$

$$
\delta(\langle q, S\rangle, \sigma) = \begin{cases} \langle \delta_1(q,\sigma), \delta_2(S,\sigma) \cup I_2\rangle & \text{if } \delta_1(q,\sigma) \in F_1 \text{ and is well-defined} \\ \langle \delta_1(q,\sigma), \delta_2(S,\sigma)\rangle & \text{if } \delta_1(q,\sigma) \notin F_1 \text{ and is well-defined} \\ \bot & \text{otherwise} \end{cases}
$$

To prove the upper bound, we want to show that $\mathcal{L}(M) = L_1L_2$.

($\Longrightarrow$) Let $w \in \mathcal{L}(M)$. By definition there exists an initial state $\langle i, S\rangle \in I$ such that $\delta(\langle i, S\rangle, w) \in F$. If $i \in F_1$, then $\varepsilon \in L_1$, $w \in L_2$ and, thus, $w \in L_1L_2$. Otherwise, if $i \notin F_1$, then there are words $u, v \in \Sigma^\star$ such that $w = uv$, $\delta_1(i, u) \in F_1$ and $\delta_2(I_2, v) \cap F_2 \neq \emptyset$. Thus, since $u \in L_1$ and $v \in L_2$, $w \in L_1L_2$.

($\Longleftarrow$) Let $w \in L_1L_2$. By definition there are words $u \in L_1$ and $v \in L_2$, such that $uv = w$. This means that $\delta_1(i, u) \in F_1$, for some initial state $i \in I_1$, and $\delta_2(I_2, v) \cap F_2 \neq \emptyset$. Thus, there is an initial state $\langle i, S\rangle \in I$ such that $\delta(\langle i, S\rangle, uv) \in F$, i.e., $w \in \mathcal{L}(M)$.

Since $|M| = n2^m - f2^{m-k}$, this completes the proof. □

B.2 Reversal

Next, we consider the reversal operation.

Theorem B.2 (Reversal). *Let $L \subseteq \Sigma^\star$ be a language with $\mathrm{msc}(L) = n$. Then, 2^n states are sufficient for an MFA to recognize L^R.*

Proof. Let $M = \langle Q, \Sigma, \delta, I, F\rangle$ be an n-state MFA such that $\mathcal{L}(M) = L$. It is trivial to see that $\mathcal{L}\big(\mathcal{D}\big(M^R\big)\big) = L^R$. Since, in the worst case, $\big|\mathcal{D}\big(M^R\big)\big| = 2^n$, this proves the upper bound. □

B.3 Kleene Star and Kleene Plus

Lastly, we consider the Kleene star and Kleene plus operation.

Theorem B.3 (Star and Plus). *Let $L \subseteq \Sigma^\star$ be a language with $\mathrm{msc}(L) = n$. Then, $2^n + 1$ (resp. 2^n) states are sufficient for an MFA to recognize $L^\star$ (resp. L^+).*

Proof. Let $M = \langle Q, \Sigma, \delta, I, F\rangle$ be an n-state MFA such that $\mathcal{L}(M) = L$. Let $N = \langle Q, \Sigma, \delta', I, F\rangle$, where

$$\delta'(q,\sigma) = \begin{cases} \{\delta(q,\sigma)\} & \text{if } q \notin F \text{ and } \delta(q,\sigma) \text{ is well-defined} \\ \{\delta(q,\sigma)\} \cup \delta(I,\sigma) & \text{if } q \in F \text{ and } \delta(q,\sigma) \text{ is well-defined} \\ \delta(I,\sigma) & \text{if } q \in F \text{ and } \delta(q,\sigma) \text{ is not well-defined} \\ \bot & \text{otherwise.} \end{cases}$$

It is straightforward to verify that $\mathcal{L}(N) = \mathcal{L}(M)^+$. This construction is a direct analogue of the standard NFA construction for the Kleene plus operation as presented by Holzer and Kutrib [HK03]. As noted by them, the same construction yields the Kleene star operation iff $\varepsilon \in \mathcal{L}(M)$.

The stated upper bound for the plus operation follows by determinizing N, yielding the automaton $\mathcal{D}(N)$ with at most 2^n states. For the star operation, the bound of $2^n + 1$ states is obtained by adding a new initial and final state to $\mathcal{D}(N)$ that has no incoming or outgoing transitions. □