Doctoral Thesis

Measuring Defects in Finite Automata

Katja Meckel

1st Reviewer (Supervisor): Prof. Dr. Martin Kutrib

2nd Reviewer: Prof. Dr. Markus Holzer

Fachbereich 07 - Mathematik und Informatik, Physik, Geographie

Institut für Informatik

Justus-Liebig-Universität Gießen

Disputation: 12.05.2016

Danksagung

An dieser Stelle möchte ich mich bei meinen Betreuern, den Herren Prof. Dr. Martin Kutrib und Prof. Dr. Markus Holzer, bedanken. Ihre Ratschläge und Anregungen waren eine unschätzbare Hilfe bei der Erarbeitung der vorliegenden Ergebnisse. Zusätzlich möchte ich auch noch einen Dank an Dr. Matthias Wendlandt und Dr. Sebastian Jakobi aussprechen. Die beiden haben mich in meinem Forschungsvorhaben ebenfalls sehr unterstützt und bei Problemen, die während der Forschung auftraten, mit ihren Ideen neue Perspektiven und Blickwinkel auf das Problem eröffnet.

Selbstverständlich möchte ich mich hiermit ebenfalls bei all jenen bedanken, die mit kritischen Bemerkungen zu einer Verbesserung dieser Arbeit beigetragen haben. Ich bedanke mich auch für die Geduld und das Verständnis, das meine Familie und Freunde während der letzten fünf Jahre mir gegenüber aufgebracht und damit ebenfalls einen Beitrag zum Gelingen dieser Arbeit geleistet haben.

Contents

List of Figures

List of Tables

1 Introduction

Back in the 1950s, Chomsky started to research for formalisms to decribe the English language and its grammar. In the papers [14] and [15], he characterised special types of grammars, where a grammar is given by rules that describe and build words and sentences of the described language. Nowadays, the introduced hierarchy for these grammars is well known as the Chomsky hierarchy. Each level of this hierachy has to obey certain restrictions for the structure of the grammar rules. The grammars with less and weaker restrictions have a greater power of description than the ones with stronger restrictions. In the theory of formal languages, the grammars describing a certain class of languages in this hierarchy are of special interest.

The grammars obeying the weakest restrictions are the type 0 grammars. They describe the class of recursively enumerable languages. The next level is given by the type 1 grammars. They describe the context-sensitive languages. This type of grammar needs to fulfill stronger restrictions than the type 0 grammars. The context-free languages are generated by the so called type 2 grammars. The lowest level of this hierarchy describes the regular languages. The grammars of this level are also called type 3 grammars. These grammars underly the strongest restrictions in the whole hierarchy defined by Chomsky.

The language classes defined in terms of grammars in the Chomsky hierarchy can also be described by different formal models. For example, there exist automata models for each of the classes.

Already in the 1930s, Turing [66] introduced a model of an abstract machine, long before there were computers. These machines are called Turing machines. The aim of Turing was to find a model that describes all that is computable, which comprises all computable numbers, functions, and so on. The Turing machines can be considered to be an abstract model of todays computers, so the Turing machines also describe the computational power of computers.

A Turing machine can also be regarded as a recognition model for formal languages. Chomsky proved that this automaton model describes precisely the class of recursively enumerable languages [15]. This means, the class of languages that are described by the type 0 grammars is the same one that is recognised by the Turing machines.

In general, a Turing machine is a device having a finite central processing unit that converts an input into an output using some internal memory. Its input is given on a tape. The memory is also considered to be a tape that the machine may use to store intermediate results needed for the computation of the output. Usually, this tape is called working tape. Genarally, these two tapes are considered to be unbounded by any size to the left and to the right. In a standard Turing machine, the transformation of an input into an output is done deterministically. This means each step of the conversion is uniquely determined by the currently read input symbol, the content of the working tape, and the current state of the machine. The output may be given on a separate tape, but, more conveniently, all tapes, the input tape, the working tape, and the output tape are considered to be only one tape. Due to the unbounded size of the tapes, this is possible.

Also modifications of Turing machines were considered over the past decades, like Turing machines with more than only one working tape, or with a working tape that is bounded on one side, or non-deterministic Turing machines. All of these modified Turing machines can be simulated by a deterministic Turing machine with only one unbounded tape. This means, the mentioned modified Turing machines also recognise the class of recursively enumerable languages (see for example [29]).

Also for the other classes of languages described in the Chomsky hierarchy, there exist recognising automata models. Just like for the grammars, the restrictions of the automata models grow for lower levels in the hierarchy. All of these automata models may be considered to be restricted Turing machines. The following three automata models are restricted Turing machines describing the other language classes of the Chomsky hierarchy.

For example, a linear bounded automaton (LBA) is a non-deterministic Turing machine, that may only access a limited number of cells of the tape. This number is a linear function in the size of the machines initial input. This automaton model was shown to recognise precisely the class of context-sensitive languages [43]. Another type of linear bounded automata are the deterministic ones (DLBA). They were introduced by Myhill [56], and were shown to recognise a proper subset of the context-sensitive languages [46].

When starting from a non-deterministic Turing machine with a one-way and read-only input tape, and one working tape that is bounded on one side, another possible restriction concerns the accessability of the content on the working tape. If the last inserted symbol needs to be processed first, that is the working tape is restricted to a last-in-first-out (LIFO) mode, the class of languages recognised by this automaton model is the class of context-free languages. This new type of automaton is called pushdown automaton (PDA) [62].

Also the finite automata (FA) that recognise the regular languages (see for example [29]) can be interpreted as restricted Turing machines. A finite automaton only consists of the finite processing unit of the Turing machine, and a read-only tape. The input of the automaton is contained on this tape in the beginning of a computation. While processing this word, the finite automaton may read the input only from left to right, and may not access any other cell. This especially means, every symbol of the input is processed only once.

Like for the Turing machines, modifications of the other automata models were introduced and investigated. Some of those new models recognise language classes that are different from the ones defined in the Chomsky hierarchy. For example the deterministic pushdown automata were shown to be strictly weaker than non-deterministic PDA [22, 23]. Also the language class accepted by DLBA mentioned before differs from the ones described in the Chomsky hierarchy.

From all of the classes of languages yet known, this thesis concentrates on the regular languages. This language class is perhaps the best researched of all yet known classes of languages. This is no surprise, since this type of language and its representations have several applications. The regular languages are used, for example, in the lexical analysis in programming language compilation [1, 2], circuit design [10], or text editing and pattern matching [42].

The class of regular languages is still one of the most intensely researched language classes in the theory of formal languages. This is particularly interesting since already in the late seventies, it was a wide spread belief that all important and everything of interest has already been known about regular languages. The only exceptance were some very hard problems, that were summarised by Brzozowski in 1979 at the International Symposium on Formal Language Theory [9].

Over the last few decades, this belief was proven to be wrong. First of all, at least three of the six very hard open problems have been solved: the restricted star height problem [24], the regularity of noncounting classes problem [18], and the problem of the optimality of prefix codes [63]. Additionally, new problems have been stated and solved for this language class. For example, the complexities for operations on regular languages respectively finite automata were introduced [26, 39, 59, 60, 68]. Some recent surveys about descriptional complexities are given by Gao, Moreira, Reis and Yu [19] and by Holzer and Kutrib [27, 28].

Since the beginnigs of the theory of formal languages and automata theory, a lot of different representations for the regular languages have been defined. Besides the type 3 grammars and finite automata, the regular languages were shown to be representable also by so called regular expressions. This model was presented by Kleene in 1956 [41]. A regular expression consists of symbols from a given alphabet, that are combined by union, concatenation, and Kleene star. Additionally, also $\emptyset$ and λ are regular expressions.

Over the years, also different models of automata were introduced, like two-way finite automata, alternating automata, and many more (see, for example, [29]). All of these models were shown to be equivalent to finite automata, and, therefore, they also describe the class of regular languages. Two somehow different types of finite automata were introduced by Mealy [50] and Moore [53]. In their definitions, the automata produce an output when processing some sequence of symbols. This is why they can be interpreted to be translaters or transducers.

Most of the equivalence results are proven by simulating one model by another one. This naturally leads to the question for a possibility to compare the different models. One such possiblity for automata recognising the regular languages is to compare the numbers of states. This number is called the size of such an automaton. The size of finite automata is also called their state complexity. For example, the size of a DFA that is equivalent to an NFA with n states can at most be 2^n. This was first shown by Myhill [55] and Nerode [57]. Their famous algorithm for this conversion is called powerset construction. There exist several examples that prove the tightness of this bound, which means there exist NFA with n states that cannot be transformed into a DFA having strictly less than 2^n states [48, 51, 54].

Other costs for conversions are for example the one between alternating finite automata and deterministic finite automata. For an AFA with n states there exist DFA with 2^{2^n} states. There exist examples such that any DFA needs not less than this many states [12]. The costs for the conversion of a 2NFA with n states into a DFA are shown in [40] to be $n(n^n - (n-1)^n)$.

The mentioned numbers for the costs of conversions are so called trade offs. In general, a trade off is a number, that enables the comparison of the size of one automaton model with another model simulating the first model. Trade offs between models describing the same family of languages can normally be expressed by a function depending on the size of the simulated model. Such a trade off is called recursive. If there does not exist such a function, the trade off is non-recursive. This type of trade off often occurs when simulating one model by another model, where the models do not recognise the same class of languages. This phenomenon is discussed in detail by Kutrib in [44]. One such non-recursive trade off occurs when simulating a pushdown automaton by a deterministic finite automaton. This was already shown by Meyer and Fischer [51].

The trade offs for the conversions of automata recognising the regular languages often differ for unary and at least binary alphabets. Chrobak showed in [16, 17] that the trade off between non-deterministic and deterministic finite automata for unary alphabets is given by $e^{\Theta(\sqrt{n \cdot \ln n})}$. Also finite languages often differ in the state complexity. For finite languages over arbitrary alphabets Salomaa and Yu showed tight bounds in [61]. Due to this, also in this thesis finite and unary regular languages are considered separately for some of the considered problems.

But not only for language classes belonging to the hierarchy defined by Chomsky trade offs were considered. Also for several classes of subregular languages trade offs for the conversion of non-deterministic to deterministic finite automata were considered for example by Bordihn, Holzer, and Kutrib [3]. In addition to trade offs for subregular languages, also the state complexity for operations on subregular languages and the finite automata recognising these types of languages were subject of research. Results on star-free languages were shown by Brzozowski and Liu in 2012 [8]. In this work, another definition for the size of deterministic finite automata was used. The used quotient complexity is not defined on the states of the automata but is related to the quotients of an automaton. Even though the approach differs, the quotient and the state complexities are equivalent. For more details on the definition of quotient complexity, the reader is referred to [5]. Other result on quotient complexities are found for example in [6] and [7].

Besides the state complexity for regular languages respectively their recognising finite automata, there exist several other measures. The state complexity often only compares the size of different automata recognising the same fixed regular language. A measure for the comparison of different regular languages is the distance. Introduced by Choffrut and Pighizzini [13], the already existing measures between words were extended to measures between regular languages. Examples for such word distances are the prefix, suffix, subword, Hamming, and the edit distance. All of these distances can be extended in the way Choffrut and Pighizzini did. They defined the distance between two regular languages to be the maximum of the suprema of the minimal distances of all words of one language to all words of the other language. This may be difficult to use in pratice, since, in general, a regular language consists of infinitely many words.

In practice, there exist a lot of applications using finite automata. Especially in language processing, finite automata with millions of states are used [52]. Like for circuits, also in finite automata defects may occur. In such big automata, it is more likely that a defect occurs, especially when changing between different representation models. Such defects may affect the states, or the transitions, or both. For a defective automaton with that many states it may be of interest to know, if the defect results in a large or small blow up in the size of an equivalent minimal deterministic automaton. If may be uselful, if the size does not change too much. Another question concerns the accepted languages. Do the languages of the defective respectivley the original automaton differ much? A practical example for such a comparison is given in the production of flat screens. A limited number of defective pixels is acceptable, in contrast to many such defects.

Another example in practice is the comparison of reproduced parts with a sample. In case that there exist too many differences, the reproduced parts will not be used or selled. For this, there needs to exist a sample. In our case for defective automata, this means the original automaton is reproducible in some way. One possibility is by its accepted language.

Another question arises naturally for defective automata: is the defect recognisable? And if it can be recognised, is possible to repair the automaton? Then the whole problem concerning blow ups for determinising and minimising such automata does not bother at all. Unfortunately, only one of the considered defects in this thesis is possible to be repaired. But for all of the defects, we will consider languages related to the languages to the defective and the original finite automaton. Depending on the sizes of the these languages, the language of the defective automaton may not differ too much from the one of the original deterministic finite automaton. But also the opposite may happen.

The second chapter of this thesis gives a short introduction to the fundamentals of formal languages and automata theory used in the subsequent chapters. This introduction focusses on the regular languages and their representation by finite automata. Also the defects investigated in most of the chapters are defined. Examples are given for the possible defects and their consequences. In the end of this chapter we explain by example the reason for the decision to only investigate such defective automata that are received by only one defect of one of the defined types.

In Chapter 3, for all defined defects, bounds for the size of minimal deterministic automata equivalent to the defective automata are given. For most of the defects, the bounds for at least binary alphabets differ from the one for unary alphabets. The results for the unary case are stated separately in this chapter.

For some defects, we also consider the possibility of the existence of defective finite automata of size n such that the equivalent minimal deterministic finite automaton has a fixed size less than or equal to the upper size bound. This question is related to the magic number problem. This problem asks for the existence of minimal deterministic automata equivalent to non-deterministic automata of size n, having a fixed size i between n and the upper bound of 2^n. If there exists a number i such that there exists no non-deterministic finite automaton having an equivalent deterministic finite automaton of size i, then this number is called magic [32].

The last section of the third chapter covers the question for the state complexity of deterministic finite automata equivalent to defective automata for finite languages. It is shown that most of the bounds for arbitrary regular languages and the finite languages differ massively.

Chapter 4 covers the questions for the recognisability of defects and the possibility of correction. Assuming that only one occurrence of one type of defect results in a given defective automaton, properties of these automata to recognise the type of defect are summarised. It is proven by example that for most of the defects it is impossible to fix the defective automaton to receive the original minimal deterministic automaton.

In this chapter, also three types of languages related to the defective and the original automaton are defined and researched. For the definition of these languages, the defective automaton, the type of defect occurring in this automaton, and the affected state is assumed to be provided. One of these languages collects all words that are accepted by the defective automaton without using the defect. Another language collects all those words that are rejected by the automaton that do not use the defect while being processed by the defective automaton. These two languages are subsets of the languages accepted respectively rejected by the original automaton. The last considered language collects all words that use the defect while being processed by the defective automaton. This includes all such accepted or rejected words. The union of all the three languages is Σ^*, where Σ denotes the underlying alphabet. For these three defined languages, upper bounds for the state complexity of minimal deterministic finite automata are considered.

In Chapter 5, a distance between two regular languages is introduced. In the first part of this chapter, this distance is based on the prefix distance between two words, and in the second part on the suffix distance. The introduced distances are parameterised. Since they sum up the word distances for words in one language to the other language and vice versa, only words up to a certain length are considered in these sums. This length is the parameter in the language distance.

Properties of this distance are investigated. It is proven that the defined distances can only be constant, polynomial, or exponential. A construction of an automaton assisting to compute the precise distance between two languages is given. Different cases are separated that may occur when calculating the distance between a word and a regular language. Based on these criteria and the constructed automaton, the precise distance for two languages can be computed. This computation is also analysed in detail which gives criteria for the decision if the distance is constant, polynomial, or exponential. All these results are proven for the parameterized prefix distance and are inherited for the parameterized suffix distance in the last part of this chapter. For this, the relation between the two distances is also proven.

The last chapter concludes this thesis. In one section, the languages related to the defective automata and the distances for defective automata are linked. It is shortly summarised that the combination of two possibilities to measure the influence of a defect may give better criteria for the decision if a defective automaton differs much or not much from the original automaton. Another section covers another definition for the languages related to the defective automaton. This definition is based only on the given defective automaton. It does not presume the knowledge of the affected state or transition. Upper bounds for minimal deterministic finite automata recognising these languages are given. The last part of Chapter 6 gives perspectives of possible future research.

2 Definitions

In this chapter, nearly all of the formalisms used in the following chapters will be defined. Most of them were already introduced by other researchers. One source for these definitions is [29], others for example are [67] or [47].

2.1 Words and Languages

A non-empty and finite set of symbols is called an *alphabet*. This is the base for the definition of languages. The symbols belonging to an alphabet are also called letters. A *word* over an alphabet Σ is a finite sequence of symbols from Σ. For example, the sequence *ababba* is a word over the alphabet $\Sigma = \{a, b\}$. The sequenc *abc* is not a word over Σ, since c does not belong to Σ, but it is a word over the alphabet $\{a, b, c, d\}$. This also shows, that it is not necessary for a word to contain all the symbols of the alphabet.

The *length* of a word w is the number of its not necessarily different symbols and is denoted by $|w|$. Let $w = a_1a_2 \ldots a_n$ be a word that consists of symbols $a_1, a_2, \ldots, a_n$ from a fixed alphabet. Then the length of w is $|w| = n$. The word of length zero, that does not contain any symbols, is called *empty word* and is denoted by λ.

The *reversal* of a word w is denoted by w^R. If $w = a_1a_2 \cdots a_k$ for some $k \geq 1$, its reversal is $w^R = a_k \cdots a_2a_1$.

Let $u = a_1a_2 \cdots a_n$ and $v = b_1b_2 \cdots b_m$ be two words, where all the symbols $a_1, a_2, \ldots, a_n$ and $b_1, b_2, \ldots, b_m$ belong to some alphabet. The *concatenation* of the words u and v is defined to be

$$u \cdot v = a_1a_2 \cdots a_nb_1b_2 \cdots b_m.$$

This means, the concatenation of two words is received by stringing them together. The concatenation symbol $\cdot$ between the two words is often omitted to simplify the expression. For every word w, its concatenation with the empty string is given by $w \cdot \lambda = \lambda \cdot w = w$.

For two words u, v over some alphabet Σ, v is called a *prefix* of u, if there exists a word w over Σ such that $u = vw$. If $v \neq u$, and $v \neq \lambda$, v is said to be non-trivial, otherwise it is a trivial prefix. In the same way, the *suffix* of a word is defined to be a word v such that the word u can be split up into $u = wv$, where w is another word over Σ. The suffix v is also called non-trivial, if $v \neq u$ and $v \neq \lambda$.

The set of all words over a fixed alphabet Σ is denoted by Σ^*. This set also includes the empty word λ. The set $\Sigma^* \setminus \{\lambda\}$ is denoted by Σ^+. It consists of all words of length at least one over Σ. The set containing all words over Σ up to a fixed length $n \geq 0$ is denoted by $\Sigma^{\leq n}$, and the set of words of length precisely n is $\Sigma^{=n}$. The *size* or *cardinality* of a set A is denoted by $|A|$. For finite sets, this number is the number of elements belonging to A. For infinite sets, this number is ∞. The *power set* of a given finite set A is denoted by 2^A, and consists of $\left|2^A\right| = 2^{|A|}$ many elements. Each of its elements is a subset of A. This also includes the empty set $\emptyset$, since this always is a subset. The set A itself also belongs to its power set.

A subset L of Σ^* is a collection of certain words. Such a set is called a *language*. The trivial languages are the empty set $\emptyset$ and the whole set Σ^*. For languages $L, L_1, L_2 \subseteq \Sigma^*$, the union

$$L_1 \cup L_2 = \{w \in \Sigma^* \mid w \in L_1 \text{ or } w \in L_2\},$$

intersection

$$L_1 \cap L_2 = \{w \in \Sigma^* \mid w \in L_1 \text{ and } w \in L_2\},$$

difference

$$L_1 \setminus L_2 = \{w \in \Sigma^* \mid w \in L_1 \text{ and } w \notin L_2\},$$

and symmetric difference

$$L_1 \triangle L_2 = (L_1 \setminus L_2) \cup (L_2 \setminus L_1)$$

are defined in the usual way. Also the Kleene star of a language

$$L^* = \{w \in \Sigma^* \mid w = w_1 \cdot w_2 \cdot \dots \cdot w_k \text{ for some } k \geq 0 \text{ and some } w_1, w_2, \dots, w_k \in L\}$$

is defined as usual, where $k = 0$ means $w = \lambda$, even though λ may not belong to L. Notice that Kleene star applied to the empty language $\emptyset$ yields the language $\{\lambda\}$.

The Kleene Star for a language can also be expressed by

$$L^* = \bigcup_{i \geq 0} L^i,$$

where the languages L^i are inductively defined by

$$L^0 = \{\lambda\}, \qquad L^{i+1} = L \cdot L^i.$$

2.2 Regular Languages and Regular Expressions

A lot of different models are existent that represent the regular languages. This class of languages, denoted by REG, is important and well studied. In the Chomsky hierarchy, described by Chomsky [14, 15], the regular languages represent the lowest level.

A common and quite natural way to describe this class of languages are the *regular expressions*. This model was introduced by Kleene [41]. A regular expression is a finite combination of concatenation, union, and Kleene star over the symbols of an alphabet and the empty language $\emptyset$.

Formally, a regular expression over an alphabet Σ is defined as follows:

- For each $a \in \Sigma$, λ, $\emptyset$, and a are regular expressions.
- If r and s are regular expressions, so are $(r \cdot s)$, $(r + s)$, and r^*.

The *language* described by regular expressions r and s is given by

- $L(\lambda) = \{\lambda\}$, $L(\emptyset) = \emptyset$, and $L(a) = \{a\}$ for all $a \in \Sigma$.
- $L(r \cdot s) = L(r) \cdot L(s)$, $L((r + s)) = L(r) \cup L(s)$, and $L(r^*) = (L(r))^*$.

To simplify the notation, it is common to give highest priority to the star operator *, followed by $\cdot$, and then $+$. Due to this priority and the associativity of concatenation and union, superfluous parentheses can be omitted. Usually, also the operator $\cdot$ is omitted. It is conventional to write r^i instead of $r \cdot r \cdot \ldots \cdot r$ (i times), and r^+ for $r \cdot r^*$.

Example 2.2.1. Let

$$r = \Bigg(\big((a+b)+c\big)^* \cdot \bigg(\Big(\big((a+c)\cdot(a)\big) \cdot \big(a+(b+c)\big)^* \Big) \cdot (a \cdot b) \bigg) \Bigg) \cdot (c)$$

be a regular expression. This expression can be simplified by omitting unnecessary braces, and the $\cdot$ for the concatenation. So, r can also be denoted by the expression $(a+b+c)^*(a+c)a(a+b+c)^*abc$. If $\Sigma = \{a, b, c\}$, the expression can be further simplified to $r = \Sigma^*(a+c)a\Sigma^*abc$. The language $L(r)$ described by the regular expression r consists of all words over $\{a, b, c\}$ that end on abc and contain at least one of the sequences aa, or ca before the last sequence of abc. This is precisely the language $L(r) = \Sigma^*\{aa, ca\}\Sigma^*\{abc\}$. $\square$

Regular expressions can also be used to prove some closure properties for REG. A language class $\mathcal{L}$ is said to be *closed* under some language operation, if all possible applications of this operation to a language belonging to $\mathcal{L}$ leads to a language that also belongs to $\mathcal{L}$. From the definition of regular expressions it is derived immediately, that the class REG is closed under the operations concatenation, union, and Kleene star. Also under the operation of reversal, REG is closed. The language $L(r)^R$ for some regular expression r can be described by its reversal r^R.

2.3 Finite Automata

Another possibility to describe the regular languages is given in form of finite automata. A finite automaton can be considered to be a device with a finite input tape, that either accepts or rejects an input. The device changes its state while reading symbols from left to right provided on the input tape, but it does not produce any output. The only information obtained from finite automata after processing some word, is the type of the state that is entered at the end of the computation. The type of a state can be accepting or non-accepting. This means, finite automata can be regarded to be some kind of recognition device for regular languages.

The following definition for non-deterministic finite automata was first introduced by Rabin and Scott [58]. They were the first to introduce non-determinism for finite automata.

A *non-deterministic finite automaton* (NFA) is a quintuple $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F\rangle$, where

- Q is a finite set of states,
- Σ is an alphabet,
- $\delta : Q \times \Sigma \to 2^Q$ is the state transition function (or simply transition function),
- $q_0 \in Q$ is the initial state, and
- $F \subseteq Q$ is the set of accepting states.

The rules defined by the state transition function determine the behaviour of the automaton. If the NFA is in some state $q \in Q$, and reads a symbol $a \in \Sigma$ on its input tape, the NFA can change its state into some state $p \in \delta(q, a)$. Since there may exist more than only one such state p, the automaton is called non-deterministic. It is also possible, that such a state p does not exist at all. In this case, the transition for q on a is said to be undefined.

In each step of the automaton, such an NFA consumes one symbol of the input and changes its state according to its transition function. If more than one state is existent, into which the automaton may change, one of them is chosen non-deterministically.

If $|\delta(q, a)| \leq 1$ for all $q \in Q$ and all $a \in \Sigma$, the automaton is called deterministic. In such a *deterministic finite automaton* (DFA), there exists no possibility for the choice of the next state for each combination of a state and a symbol in the input. In this thesis, all DFA are considered to be complete, which means there exist no undefined transitions. Every incomplete DFA can be converted into a complete DFA. This is done by letting each undefined transition lead into a rejecting *sink state*, which is a non-leaving state.

The state transition function δ of a finite automaton $\langle Q, \Sigma, \delta, q_0, F\rangle$ is defined only for single symbols. For all states $q \in Q$, it can be extended to a transition function for words $w \in \Sigma^*$ by $\delta(q, w) = \bigcup\limits_{p \in \delta(q,a)} \delta(p, w)$, if $w = av$ for some symbol $a \in \Sigma$ and a word $v \in \Sigma^*$, respectively by $\delta(q, w) = \{q\}$, if $w = \lambda$. To simplify the notation of the transition function for DFA, the set braces are omitted for the successing state.

While processing a word, a finite automaton enters several states. Such a sequence of states is called a *path* of the automaton. In this sequence, it is possible that one state appears several times. The *length* of a path is defined to be the number of states contained in the path. If a state occurs several times, each occurrence is counted separately. Let for example $s_1 s_2 s_1 s_3$ denote a path. Then its length is four, even though s_1 appears twice. A subsequence of the path that begins with a state s, ends with the same state s, with s not appearing elsewhere in this subsequence, is called a *cycle* or *ring* of the DFA. The *length* of a cycle is defined as the number of states within the cycle minus one, since state s builds the beginning and the end of the cycle. If a cycle is of length one, it is also called a *loop.* A loop is nothing else but a single transition that does not change the state.

For every word over the alphabet of the automaton there exists a path that begins with the initial state. Let $s_0 s_1 \dots s_k$ be a path for some word, where $s_0 = q_0$ and $k \geq 0$, and let s_i be one of these states. All the states, that are visited before entering s_i are called its *predecessors*, and all the states that are visited after this state are the *successors* of s_i. In the path, this means thart all states s_j, where $j < i$ are predecessors, and that all states s_l with $l > i$ are the successors.

Let $\mathcal{A}$ be a finite automaton, either deterministic or non-deterministic. The *accepted language* of $\mathcal{A}$, in terms $L(\mathcal{A})$, is the set of words that lead the automaton into acceptance, when starting the processing in the initial state. Formally, for a finite automaton $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F\rangle$, its accepted language is defined to be $L(\mathcal{A}) = \{w \in \Sigma^* \mid \delta(q_0, w) \cap F \neq \emptyset\}$.

Two finite automata $\mathcal{A}$ and $\mathcal{B}$ are said to be equivalent, in terms $\mathcal{A} \equiv \mathcal{B}$, if they accept the same language, which means $L(\mathcal{A}) = L(\mathcal{B})$.

Kleene showed that the language class that is recognised by finite automata is precisely the class REG [41].

Theorem 2.3.1 (Kleene [41]). *A language L is regular if and only if there exists a finite automaton $\mathcal{A}$ with $L(\mathcal{A}) = L$.*

Conversions of NFA into DFA Since NFA and DFA are both models that describe the class REG, it should be possible to convert one model into the other. Indeed, this is possible. It was shown by Rabin and Scott in 1959 [58]. The conversion is done by the so-called power set construction.

Let $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F\rangle$ be an arbitrary NFA. An equivalent DFA is given by $\mathcal{B} = \langle 2^Q, \Sigma, \delta', \{q_0\}, F'\rangle$, where $\delta'(P, a) = \bigcup_{p \in P} \delta(p, a)$ for all $P \subseteq Q$ and $a \in \Sigma$, and $F' = \{P \in 2^Q \mid P \cap F \neq \emptyset\}$.

The following example gives an idea of how the power set construction works. It also shows how finite automata are represented by state transition diagrams.

Example 2.3.1. Let $\mathcal{A} = \langle\{q_0, q_1, q_2, q_3\}, \{a, b\}, \delta, q_0, \{q_3\}\rangle$ be the NFA with state transition function

$$\begin{aligned}
&\delta(q_0, a) = \{q_1, q_2\}, \delta(q_0, b) = \{q_2\},\\
&\delta(q_1, a) = \delta(q_1, b) = \{q_2\},\\
&\delta(q_2, a) = \{q_3\},\\
&\delta(q_3, a) = \delta(q_3, b) = \{q_3\}
\end{aligned}$$

that is depicted in Figure 2.1.

An equivalent DFA is given by $\mathcal{B} = \langle Q, \{a, b\}, \delta', \{q_0\}, F\rangle$, where its state set is $Q = \{\emptyset, \{q_0\}, \{q_2\}, \{q_3\}, \{q_1, q_2\}, \{q_2, q_3\}\}$, its set of accepting states is given by

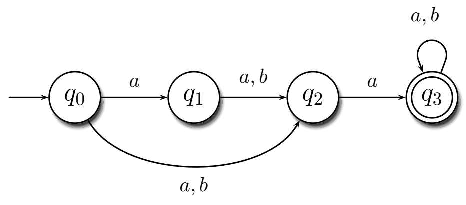


Figure 2.1: NFA that is converted into a DFA in Example 2.3.1.

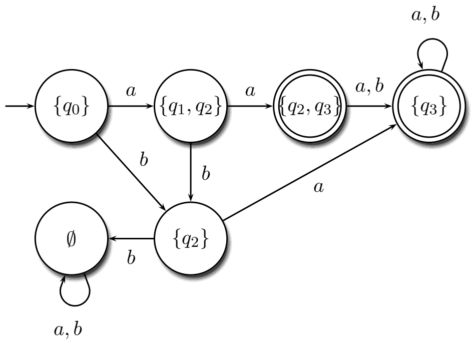


Figure 2.2: DFA that is constructed by the power set construction from the NFA depicted in Figure 2.1.

$F = \{\{q_3\}, \{q_2, q_3\}\}$, and its transition function is defined by

$$\begin{aligned}
&\delta'(\{q_0\}, a) = \{q_1, q_2\}, \delta'(\{q_0\}, b) = \{q_2\},\\
&\delta'(\{q_2\}, a) = \{q_3\}, \delta'(\{q_2\}, a) = \emptyset,\\
&\delta'(\{q_3\}, a) = \delta'(\{q_3\}, b) = \{q_3\},\\
&\delta'(\{q_1, q_2\}, a) = \{q_2, q_3\}, \delta'(\{q_1, q_2\}, b) = \{q_2\},\\
&\delta'(\{q_2, q_3\}, a) = \delta'(\{q_2, q_3\}, b) = \{q_3\},\\
&\delta'(\emptyset, a) = \delta'(\emptyset, b) = \emptyset.
\end{aligned}$$

This automaton is depicted in Figure 2.2 ⌜⌟

The formal description of the DFA constructed by the power set construction contains all possible states. The states that are not depicted in Figure 2.2 cannot be reached from the initial state $\{q_0\}$ by any word over $\{a, b\}$. It is a general convention to omit such states for finite automata.

Minimal Deterministic Finite Automata Taking a closer look at the constructed DFA from Example 2.3.1, one can recognise that by the states $\{q_2, q_3\}$ and $\{q_3\}$ the same words are accepted. This leads to the formal definition of equivalent states:

Given a DFA with state transition function δ. Two states p and q are said to be *equivalent*, if for any word w, $\delta(p,w)$ is accepting if and only if $\delta(q,w)$ is accepting. Otherwise, the states p and q can be distinguished and are *inequivalent*.

Equivalent states can be merged. This means, for a DFA $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F\rangle$, and two equivalent states $p, q \in Q$, a new DFA $\mathcal{A}' = \langle Q \setminus \{p\}, \Sigma, \delta', q_0', F \setminus \{p\}\rangle$ is constructed, where

$$\delta'(s,a) = \begin{cases} q & \text{if } \delta(p,a) = p, \\ \delta(s,a) & \text{else} \end{cases},$$

and

$$q_0' = \begin{cases} q & \text{if } q_0 = p, \\ q_0 & \text{else} \end{cases}.$$

Merging equivalent states always leads to a new DFA. If a DFA only contains states that are not equivalent, it is called *minimal*.

There exist several algorithms to minimise a DFA, for example one by Brzozowski [4], one by Hopcroft and Ullman [29], one by Huffman [30], or one by Moore [53].

One well-known algorithm to minimise a DFA is the so called *table-filling algorithm*, see for example [29]. This algorithm determines recursively all inequivalent states by marking pairs of states. For a given DFA $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F\rangle$, at first all state pairs of states are marked, where only one of them is accepting. Then, for all non-marked pairs of states p and q, and symbols $a \in \Sigma$, this pair is marked, if and only if the state pair $\delta(p,a)$ and $\delta(q,a)$ is already marked. This algorithm stops, if there either exist no more unmarked pairs of states, or if there exists no more possibility to mark the remaining unmarked pairs. The unmarked pairs are exactly the equivalent states.

Theorem 2.3.2. *If two states are not distinguished by the table-filling algorithm, then the states are equivalent.*

The following important property of deterministic finite automata results from this.

Theorem 2.3.3. *If $\mathcal{A}$ is a DFA, and $\mathcal{B}$ the DFA constructed from $\mathcal{A}$ by the table-filling algorithm, then $\mathcal{B}$ has as few states as any DFA equivalent to $\mathcal{A}$.*

From all these properties of deterministic finite automata, the following theorem can be derived, that gives a characterisation of minimal DFA.

Theorem 2.3.4. *A DFA is minimal if and only if all states are reachable from its initial state, and if all of its states are pairwise inequivalent.*

This especially means, that for any regular language, there exists a DFA that accepts this language, and has a minimal number of states. This also means, all minimal DFA are the same except for a renaming of the states. Therefore, this minimal number of states can be used to measure the size of DFA and indirectly the size of regular languages. This measure is called *state complexity.*

This measure can also be extended to non-deterministic finite automata. A minimal NFA accepting a certain language is defined to be an NFA, for which there exists no equivalent NFA that has fewer states.

Now it is possible to use this measure to express the compactness of the representation of regular languages by NFA instead of DFA. It was shown by Rabin and Scott [58] that an upper bound for the conversion of NFA into DFA is exponential. The tightness of this bound was shown later independently by Lupanov [48], Moore [54], and Meyer and Fischer [51].

Theorem 2.3.5. *There exist regular languages, that are accepted by NFA with n states, and by minimal DFA with 2^n states.*

2.4 Defects in Finite Automata

Finite automata can be manipulated. Such a manipulation can affect either the transitions or the states. In the following, the manipulations are called defects. A finite automaton that contains a defect will be called *defective.* The automaton, from which the defective automaton is received by some defect, is called *original* automaton. The investigations in this thesis concentrate on the following defects.

Definition 2.4.1. The following defects can affect a deterministic finite automaton. Their definitions will be reduced to only one occurrence of the defect.

Exchange Symbol: The symbol of a transition is exchanged by another one.

Flip Transition: The source and target of a transition are interchanged.

Delete Transition: A transition is deleted.

Insert Transition: A new transition is inserted.

Delete Accepting State: An accepting state is deleted, including all transitions having this state as source or target.

Delete Non-Accepting State: A non-accepting state is deleted, including all transitions having this state as source or target.

Remove Acceptance: The property of acceptance is removed for a state.

Add Acceptance: The property of acceptance is added for a state.

Example 2.4.1. Let $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F\rangle$ be the DFA depicted in Figure 2.3, where $Q = \{q_0, q_1, q_2\}$, $\Sigma = \{a, b\}$, $F = \{q_0, q_2\}$, and

$$\begin{aligned}\delta(q_0, a) = \delta(q_0, b) = q_1,\\ \delta(q_1, a) = \delta(q_1, b) = q_2,\\ \delta(q_2, a) = \delta(q_2, b) = q_2.\end{aligned}$$

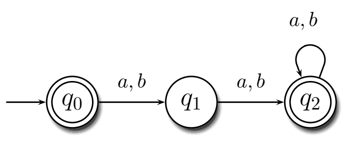


Figure 2.3: Minimal DFA accepting the language $\{a,b\}^2\{a,b\}^* \cup \{\lambda\}$.

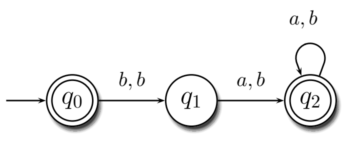


Figure 2.4: Defective finite automaton derived from the DFA depicted in Figure 2.3 for the defect exchange symbol affecting the transition on a for state q_0.

This automaton accepts the language $\{a,b\}^2\{a,b\}^* \cup \{\lambda\}$.

The defect exchange symbol affecting the transition from state q_0 to state q_1 by replacing a by the symbol b in DFA $\mathcal{A}$ results in an incomplete non-deterministic finite automaton. This defective automaton is depicted in Figure 2.4, and accepts the language $\{b\}\{a,b\}^+ \cup \{\lambda\}$.

If the defect flip transition affects the transition from state q_0 to q_1 on symbol b for DFA $\mathcal{A}$, the resulting defective DFA is the one given in Figure 2.5 which accepts the language $\{a\}\{ba\}^*\{a,b\}^+ \cup \{\lambda\}$.

The automaton depicted in Figure 2.6 accepts the language $\{a\}\{a,b\}^+ \cup \{\lambda\}$, but is the result of the defect delete the transition affecting the transition on b for state q_0 in DFA $\mathcal{A}$.

If the defect insert transition affects the DFA $\mathcal{A}$ by inserting a transition on b from state q_1 to state q_0, the accepted language does not change. The resulting defective automaton is depicted in Figure 2.7.

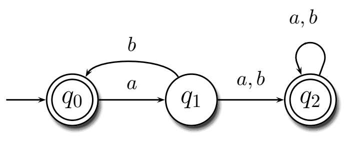


Figure 2.5: Defective finite automaton derived from the DFA depicted in Figure 2.3 for the defect flip transition affecting the transition on b for state q_0.

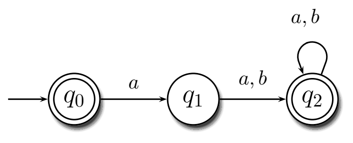


Figure 2.6: Defective finite automaton derived from the DFA depicted in Figure 2.3 for the defect delete transition affecting the transition on b for state q_0.

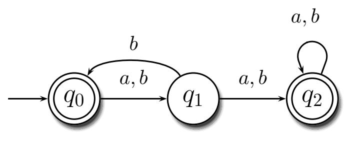


Figure 2.7: Defective finite automaton derived from the DFA depicted in Figure 2.3 for the defect insert transition that inserts the transition $\delta(q_1, b) = q_0$.

The deletion of the accepting state q_2, or the non-accepting state q_1 both result in defective finite automata, that are both incomplete DFA. These two automata are depicted in Figures 2.8 respectively 2.9. Both automata only accept the language $\{\lambda\}$.

If the acceptance property of state q_0 for DFA $\mathcal{A}$ is removed, the language accepted by the defective automaton only differs by the empty word from language $L(\mathcal{A})$. The defective automaton is depicted in Figure 2.10.

The last remaining defect of adding the acceptance for some state of $\mathcal{A}$ results in the automaton from Figure 2.11. This automaton accepts the language $\{a, b\}^*$. ⌜⌟

In general, all defects change the accepted language of the affected automaton. If the DFA is affected by more than only one defect, it is possible that the accepted language does not change at all, which means in detail that the defective DFA is equivalent to the original DFA. This is the case if, for example, the defects delete transition and insert transition affect the same transitions.

If is also possible that one defect applied several times results in a massive change

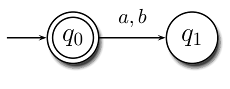


Figure 2.8: Defective finite automaton derived from the DFA depicted in Figure 2.3 by the defect delete accepting state affecting state q_2.

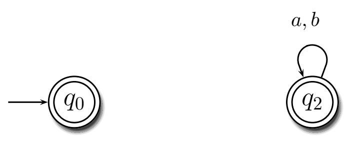


Figure 2.9: Defective finite automaton derived from the DFA depicted in Figure 2.3 by the defect delete non-accepting state affecting state q_1.

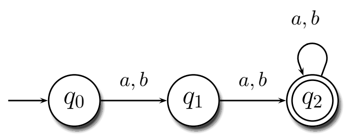


Figure 2.10: Defective finite automaton derived from the DFA depicted in Figure 2.3 for the defect remove acceptance affecting state q_0.

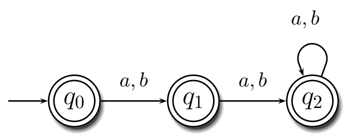


Figure 2.11: Defective finite automaton derived from the DFA depicted in Figure 2.3 for the defect add acceptance affecting state q_1.

of the accepted language. For example, if the defect of removing acceptance for some states affects a minimal DFA several times, the accepted language may change massively. This can be seen by removing the acceptance of the states q_0 and q_2 of the minimal DFA depicted in Figure 2.3. The resulting defective automaton does not accept any word, which means that infinitely many words are not accepted anymore. Similar things may happen when combining different defects. Combinations of defects may also lead to other types of defects. For example, the combination of inserting and deleting one transition each can result in the defect flip transition. If the transition of DFA $\mathcal{A}$ depicted in Figure 2.3 on b is deleted for state q_0, but a transition on b is added to q_1 that leads into state q_0, the resulting defective finite automaton is the one we received in Example 2.4.1 for the defect flip transition.

This is the reason why, in this thesis, only one occurrence of a single defect will be considered.

3 State Complexity of DFA for Defects

This chapter deals with the state complexity of finite automata resulting from defects. These automata are retrieved from minimal and complete DFA which are affected by a defect. Those defects always lead to at least incomplete DFA or even to non-deterministic finite automata. These automata will be converted into complete and minimal DFA again. The size of these automata will be given in dependency of the size of the original, non-defective DFA.

3.1 Defects Occurring on the Transitions

In this section, we concentrate on defects on the transitions of a DFA. The possible defects on transitions are exchange symbol, flip transition, delete transition, and insert transition.

The first defect to be examined is exchange symbol. The following theorem gives a tight upper bound for the state complexity of a complete and minimal DFA accepting the language of the defective automaton. Its size will be determined in terms of the size of the original DFA.

Theorem 3.1.1. *Let $\mathcal{A}$ be a minimal DFA with $n \geq 2$ states and let the NFA, which is received by the defect exchange symbol affecting DFA $\mathcal{A}$ be denoted by $\mathcal{A}'$. Then the equivalent minimal DFA of $\mathcal{A}'$ needs at most 2^n states. This bound is tight even for binary alphabets.*

Proof. In the worst case the considered defect implements ambiguity in the automaton and leads to undefined transitions. An equivalent minimal DFA may need all states generated by the power set construction. This means a minimal DFA has at most 2^n states.

We now show that this bound is reachable by a defect on a single transition in a DFA with binary alphabet. Let $\mathcal{A} = \langle Q, \{a,b\}, \delta, q_1, \{q_n\}\rangle$ be the DFA depicted in Figure 3.1 with $Q = \{q_1, q_2, \ldots, q_n\}$ and

$$\delta(q_j, a) = \begin{cases} q_{j+1} & \text{if } j \in \{1,2,\ldots,n-1\}, \\ q_2 & \text{if } j = n \end{cases},$$

$$\delta(q_j, b) = \begin{cases} q_1 & \text{if } j = 1 \text{ or } j = n, \\ q_{j+1} & \text{if } j \in \{2,3,\ldots,n-1\} \end{cases}.$$

This DFA is complete. It is also minimal since for $x, y \in \{1,2,\ldots,n-1\}, x < y$, we have that $\delta(q_x, a^{n-y}) \neq q_n$ but $\delta(q_y, a^{n-y}) = q_n$.

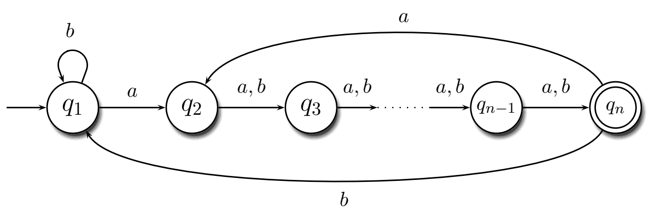


Figure 3.1: DFA that proves the optimality of the upper bound for the defect exchange symbol.

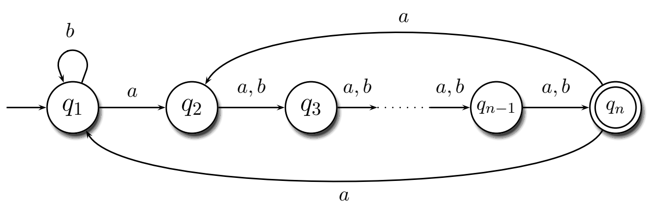


Figure 3.2: NFA of Moore whose equivalent minimal DFA needs precisely 2^n states for $n \geq 2$ [54].

Modifying the transition $\delta(q_n, b) = q_1$ by exchanging the letter b by a results in the NFA given in Figure 3.2. For this automaton it was already shown by Moore in [54] that the equivalent minimal DFA needs precisely 2^n states for $n \geq 2$. $\square$

This defect has no effect for unary alphabets since in such an alphabet there do not exist enough symbols to exchange one by another. This gives a tight bound of n states for unary alphabets.

The theorem below provides the upper bound for the state complexity of a DFA affeted by the defect flip transition.

Theorem 3.1.2. *Let $\mathcal{A}$ be a minimal DFA with $n \geq 2$ states and let the NFA $\mathcal{A}'$ be the automaton that is received from $\mathcal{A}$ by the defect flip transition. Then the equivalent minimal and complete DFA of NFA $\mathcal{A}'$ needs at most 2^n states. This bound is tight even for binary alphabets.*

Proof. In the worst case, the modification implements ambiguity in the automaton and leads to undefined transitions. An equivalent complete and minimal DFA may need all states generated by the power set construction. This means such a DFA needs at most 2^n states.

In the following we show that this bound can be reached by manipulating a single transition in a DFA with a binary alphabet. Let $\mathcal{A} = \langle Q, \{a, b\}, \delta, q_1, \{q_n\} \rangle$ be the

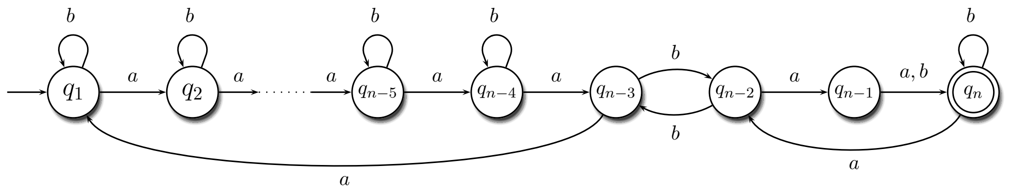


Figure 3.3: DFA that proves the tightness of the upper bound for the defect flip transition.

DFA from Figure 3.3 with state set $Q = \{q_1, q_2, \ldots, q_n\}$ and

$$\delta(q_j, a) = \begin{cases} q_{j+1} & \text{if } j \in \{1, 2, \ldots, n-4, n-2, n-1\} \\ q_1 & \text{if } j = n-3, \\ q_{n-2} & \text{if } j = n \end{cases},$$

$$\delta(q_j, b) = \begin{cases} q_j & \text{if } j \in \{1, 2, \ldots, n-4, n\}, \\ q_{j+1} & \text{if } j \in \{n-3, n-1\}, \\ q_{n-3} & \text{if } j = n-2 \end{cases}.$$

This DFA is complete. The following considerations prove the minimality of this automaton.

In case state q_i belongs to the set $\{q_1, q_2, \ldots, q_{n-3}\}$ it is reachable from state q_1 by the word a^{i-1}. If q_i is one of the states q_{n-2}, q_{n-1}, or q_n it can be reached from the initial state by the word $a^{n-4}ba^{i-(n-2)}$.

For indices $x, y \in \{1, 2, \ldots, n-3\}, x < y$, we have that $\delta(q_x, a^{n-3-x}ba^2) = q_n$ but $\delta(q_y, a^{n-3-x}ba^2) \neq q_n$. It also holds that $\delta(q_{n-2}, ab) = q_n$ but $\delta(q_x, ab) \neq q_n$, $\delta(q_{n-1}, a) = q_n$ but $\delta(q_x, a) \neq q_n$ and $\delta(q_{n-2}, a) \neq q_n$. This proves that all the states of automaton $\mathcal{A}$ are reachable and inequivalent.

In case the considered defect flip transition affects the transition $\delta(q_{n-1}, b) = q_n$ it is replaced by the transition $\delta(q_n, b) = q_{n-1}$. This leads to the NFA given in Figure 3.4. For this automaton Jirásková and Šebej have already shown in [38] that the equivalent minimal DFA needs exactly 2^n states for $n \geq 2$. □

Theorem 3.1.2 gives a tight bound for at least binary alphabets. For unary alphabets, this bound differs. The following theorem covers the unary case.

Theorem 3.1.3. *Let $\mathcal{A}$ be a minimal DFA with $n \geq 2$ states and unary alphabet, and let $\mathcal{A}'$ be the NFA that is received by the defect flip transition affecting $\mathcal{A}$. Then the equivalent minimal DFA for $\mathcal{A}'$ needs at most $n+1$ states. This bound is tight.*

Proof. It is well known since Chrobak [16, 17] that a complete and minimal DFA over a unary alphabet consists of a connected chain of states followed by a ring of states.

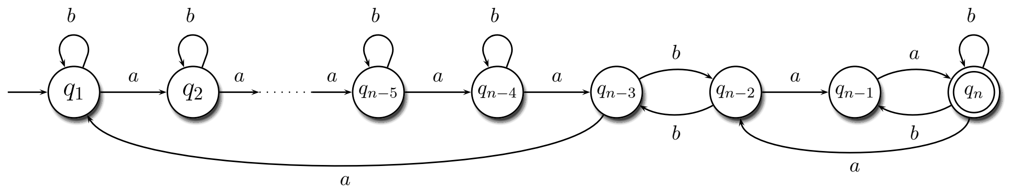


Figure 3.4: NFA of Jirásková and Šebej for which the equivalent minimal DFA has exactly 2^n states for $n \geq 2$.

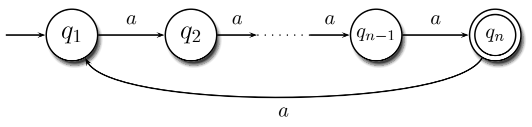


Figure 3.5: DFA $\mathcal{A}$ for the defect flip transition for unary alphabets.

This ring may be a single (accepting or non-accepting) state having a transition to itself on the only symbol of the alphabet. It can also contain more than one state. It is also possible that the chain of states does not exist at all and the automaton only consists of a ring of states. In a DFA, where the ring is a single accepting state, or consists of more than only one state, a rejecting sink state does not exist. In the following, only DFA of this structure will be considered.

The structure of such DFA especially means that there exists only one transition leaving each state and for most states only one transition enters this state. The only exception to this is the last state of the chain, which is the end of the chain and the beginning and ending of the ring at the same time. This state q is the only state that is the target of two transitions.

If the defect flip transition affects a unary minimal DFA, the resulting defective automaton always accepts a finite language. The reason for this is, that the interchanging of the source state s and target state $t \neq q$ of a transition deletes the connection between s and t. Due to the structure of unary DFA, this means at the same time the loss of reachability of all states that are reached from s in the original DFA. The equivalent minimal DFA to this defective automaton needs at most as many states as the original one, since the insertion of a rejecting sink state suffices.

If the target state is q, all states may still be reachable, since there exist two transitions leading into q. This happens, if the transition leading from the last state of the ring r into the first state of the ring q is flipped. In this case, also a finite language is accepted by the defective automaton. The longest word of this language has at most $n-1$ symbols, if the number of states of the original DFA is n. It is well known, that a minimal DFA accepting a (unary) finite language has a number of states, that is given by the length of the longest word plus two. In this case, this gives the upper bound of n+1.

To show the optimality of this bound, let $\mathcal{A} = \langle Q, \{a\}, \delta, q_1, \{q_n\}\rangle$ be the minimal DFA depicted in Figure 3.5, where $Q = \{q_1, q_2, \ldots, q_n\}$ is the set of states and

$$\delta(q_j, a) = \begin{cases} q_{j+1} & \text{if } j \in \{1, 2, \ldots, n-1\}, \\ q_1 & \text{if } j = n \end{cases}$$

is its state transition function.

This DFA is complete. The following considerations provide the minimality of the chosen automaton.

Every state $q_i \in \{q_1, q_2, \ldots, q_n\}$ is reachable from the initial state by the word a^{i-1}.

For indices $x, y \in \{1, 2, \ldots, n-1\}, x < y$, we have that $\delta(q_x, a^{n-y}) \neq q_n$ but $\delta(q_y, a^{n-y}) = q_n$. Thus, all the states of the considered automaton are reachable and inequivalent.

If the defect interchanges the transition from state q_n to q_1, we receive an undefined transition for state q_n and an ambiguity for state q_1. The accepted language is modified to the finite language $\{a, a^{n-1}\}$. It is well known that the minimal DFA accepting this language needs exactly $n+1$ states, n states to count up to the length $n-1$ and one more state to reject all words longer than $n-1$. This proves the optimality of the upper bound. □

For the defect flip transition in the unary case, the next theorem proves that it is possible for the minimal DFA that is equivalent to the defective automaton to have any number of states below the upper bound.

This question is closely related to the one asking for magic numbers. This question was stated by Iwama, Kambayashi, and Takaki [31]. It asks whether there always exists a nondeterministic finite automaton with n states whose equivalent minimal DFA has α states for all integers n and α, with $n \leq \alpha \leq 2^n$.

Iwama, Matsuura, and Paterson [32] called a number α *magic*, if there exists no NFA with n states having an equivalent DFA with α states. In [31] it was shown, that $\alpha = 2^n - 2^k$ or $\alpha = 2^n - 2^k - 1$, for $0 \leq k \leq n/2 - 2$ is non-magic, and in [32] this was shown for $\alpha = 2^n - k$, with $5 \leq k \leq 2n - 2$, and some coprimality condition for k. These results were shown for binary NFA.

The results for binary alphabets were improved in [34], [35], [20], and [49]. The question asking for magic numbers turned out to become easier, if the size of the alphabet is increased. In [34] it was even shown that for exponentially growing alphabets, no magic numbers exist at all. In [20], this result was improved to smaller growing alphabets. In [33] it was shown for four-letter alphabets that there exist no magic numbers, and in [37], this was even proven for ternary alphabets. The magic number problem for unary alphabets was studied in [21]. Further results on magic numbers can be found, for example, in [35], [36], and [25].

Theorem 3.1.4. *Let $n \geq 4$ be a number. Then a minimal DFA $\mathcal{A}$ exists with n states and unary alphabet, such that by the defect flip transition, the equivalent*

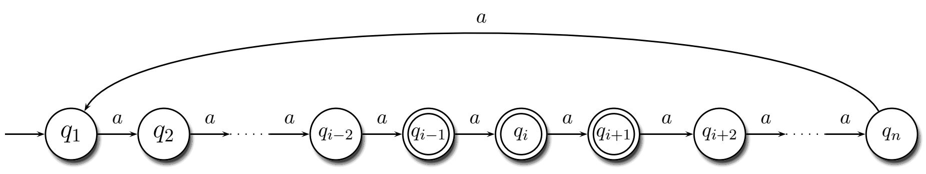


Figure 3.6: DFA for which the defect flip transition can result in any number of states between 2 and $n-1$ for the equivalent minimal DFA.

minimal DFA for the defective automaton needs precisely a number of i states, for all $1 \leq i \leq n+1$.

Proof. The existence of a minimal DFA with $n+1$ states was already shown in Theorem 3.1.3.

To prove the existence of a minimal DFA with a number of states i between 1 and n, that is equivalent to a defective automaton, we use the DFA $\mathcal{A} = \langle Q, \Sigma, \delta, q_1, F \rangle$ depicted in Figure 3.6, where $Q = \{q_1, q_2, \ldots, q_n\}$ is the set of states, its set of accepting states is given by $F = \{q_{i-1}, q_i, q_{i+1}\}$, and the transition function δ is defined by

$$\delta(q_j, a) = \begin{cases} q_{j+1} & \text{if } j \in \{1, 2, \ldots, n-1\}, \\ q_1 & \text{if } j = n \end{cases}.$$

This DFA is complete. It is also minimal, since each state q_j, $1 \leq j \leq n$, is reachable from the initial state q_1 by the word a^{j-1}. The accepting states can be differentiated by the words a and a^2 since then we have $\delta(q_{i-1}, a^2) = q_{i+1}$, $\delta(q_i, a^2), \delta(q_{i+1}, a^2) \notin \{q_{i-1}, q_i, q_{i+1}\}$, and $\delta(q_i, a) = q_{i+1}, \delta(q_{i+1}, a^2) \notin \{q_{i-1}, q_i, q_{i+1}\}$.

For the states with indices $1 \leq j < l \leq i-2$, we have $\delta(q_l, a^{i-l-1}) = q_{i-1}$, but $\delta(q_j, a^{i-l-1}) = q_k$, where $k < i-1$. Thus, the states from the set $\{q_1, q_2, \ldots, q_{i-2}\}$ are pairwise inequivalent. The same is true for the states with indices j and l where $i+1 \leq j < l \leq n$, since $\delta(q_l, a^{n+i-l-1}) = q_{i-1}$ is accepting, but on the other hand we have $\delta(q_j, a^{n+i-l-1}) \notin \{q_{i-1}, q_i, q_{i+1}\}$, and, therefore, non-accepting.

Any state $q_j \in \{q_1, q_2, \ldots, q_{i-2}\}$ is inequivalent to any state $q_l \in \{q_{i+2}, q_{i+3}, \ldots, q_n\}$ since $\delta(q_j, a^{i-j-1}) = q_{i-1}$, but $\delta(q_l, a^{i-j-1}) \notin \{q_{i-1}, q_i, q_{i+1}\}$.

In the following, we distinguish the two cases $2 \leq i \leq n-1$, and $i = 1$ or $i = n$.

To receive a number of states i between 2 and $n-1$ for the minimal DFA equivalent to the defective automaton, we assume that the considered defect affects the transition from state q_{i-1} to q_i of DFA $\mathcal{A}$. Then all states of the set $\{q_i, q_{i+1}, \ldots, q_n\}$ are not reachable anymore. The remaining accepted language is the language $\{a^{i-2}\}$, which is finite and can be accepted by a minimal DFA with i states.

For the case $i = 1$ or $i = n$, we modify the automaton from above. Let q_{n-2}, q_{n-1}, and q_n be the accepting states in this case. If the transition from q_1 to q_2 is defective,

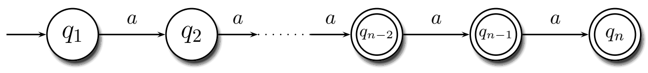


Figure 3.7: An incomplete DFA whose equivalent minimal DFA needs exactly $n+1$ states.

the resulting automaton accepts the empty language. Thus, one state is sufficient for an equivalent minimal DFA. This proves $i = 1$. In case the defect affects the transition from q_{n-1} to q_n, the automaton accepts the finite language $\{a^{n-3}, a^{n-2}\}$, which can be accepted by a minimal DFA with n states. This concludes the proof. □

The three theorems from above considering the defect flip transition show that this defect can result in a very huge minimal DFA for arbitrary alphabets. For unary alphabets, this number is much smaller. This, again, proves the difference between unary alphabet sizes, and at least binary alphabets concerning the state complexity.

In the proof of Theorem 3.1.4, we used the DFA depicted in Figure 3.6. This DFA accepts an infinite language. The defective automata constructed in the proof accept finite languages. This means that the defect can result in a loss of a lot of information about the accepted language. This is motivates the analysis of defects.

The next theorem proves an upper bound for the number of states of an equivalent minimal DFA that is received by the defect delete transition affecting a given DFA.

Theorem 3.1.5. *Let $\mathcal{A}$ be a complete and minimal DFA with $n \geq 4$ states and let the NFA $\mathcal{A}'$ be received from $\mathcal{A}$ by the defect delete transition. Then the equivalent minimal DFA of $\mathcal{A}'$ needs at most $n+1$ states. This bound is tight even for unary alphabets.*

Proof. Even in the worst case, the deletion of a transition in a complete and minimal DFA cannot implement ambiguity in the automaton. It can only lead to undefined transitions. Thus, an equivalent minimal DFA may need at most one more state that represents a rejecting sink which will be the target of every undefined transition. Therefore, a minimal DFA needs at most a number of $n+1$ states.

To prove the optimality of the bound for this defect affecting only a single transition in a minimal DFA with unary alphabet, we will use the DFA depicted in Figure 3.5. We assume the set of accepting states to be $\{q_{n-2}, q_{n-1}, q_n\}$. In the proof of Theorem 3.1.3 it was already shown, that this DFA is complete and minimal.

Let this DFA be defected by deleting the transition $\delta(q_n, a) = q_1$. We receive the incomplete DFA given in Figure 3.7. This automaton accepts the finite language $\{a^{n-3}, a^{n-2}, a^{n-1}\}$, which is known to be accepted by a minimal DFA with $n+1$ states. □

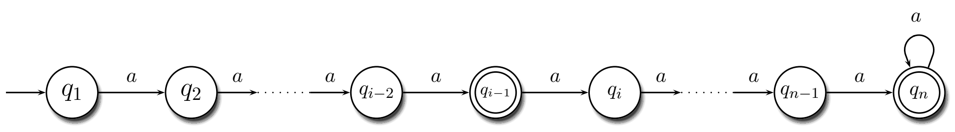


Figure 3.8: DFA to prove the existence of minimal DFA equivalent to defective automata for the defect delete transition with a number of states between 1 and $n-1$.

We now know that a minimal DFA equivalent to the defective automaton needs at most $n+1$ states when deleting one transition of a given DFA with $n \geq 2$ states. In this context the question naturally arises if there exists an equivlant minimal DFA for any number of states between 1 and $n+1$ for the defect delete transition. The following theorem provides the answer.

Theorem 3.1.6. *Let $n \geq 3$ be a number. Then there exist minimal DFA $\mathcal{A}$ with n states, such that by the defect delete transition, the equivalent minimal DFA for the defective automaton needs precisely a number of i states, for all $1 \leq i \leq n+1$. This is true even for unary alphabets.*

Proof. In the proof of Theorem 3.1.5 it has already been shown that there exists a DFA with $n \geq 1$ states that meets the upper bound of $n+1$ states. For the remaining numbers of states, we first consider $i \leq n-1$. Let $\mathcal{A} = \langle Q, \{a\}, \delta, q_1, F \rangle$ be the DFA from Figure 3.8 with $Q = \{q_1, q_2, \ldots, q_n\}$, $F = \{q_{i-1}, q_n\}$ where $i-1=0$ means that q_n is the only accepting state of the automaton, and

$$\delta(q_j, a) = \begin{cases} q_{j+1} & \text{if } j \in \{1, 2, \ldots, n-1\}, \\ q_n & \text{if } j = n \end{cases}.$$

This DFA is complete. The following considerations provide the minimality of automaton $\mathcal{A}$.

Every state $q_j \in \{q_1, q_2, \ldots, q_n\}$ is reachable by a^{j-1}.

For indices $x, y \in \{1, 2, \ldots, n-1\} \setminus \{i-1\}, x < y$, either $\delta(q_x, a^{n-y}) \neq q_{i-1}$ or $\delta(q_x, a^{n-y}) = q_{i-1}$ but $\delta(q_y, a^{n-y}) = q_n$. In the first case, q_x and q_y are inequivalent. In the other case, q_x and q_y can be distinguished by reading $a^{n-y}a$ since then we have $\delta(q_x, a^{n-y}a) = q_i \notin F$ and still $\delta(q_y, a^{n-y}a) = q_n$. To show that states q_{i-1} and q_n are inequivalent, we use the word a and obtain $\delta(q_{i-1}, a) = q_i$, $i < n$ and $\delta(q_n, a) = q_n$. Thus, all the states are reachable and inequivalent.

For $2 \leq i \leq n-1$, we modify the DFA $\mathcal{A}$ by deleting the transition $\delta(q_{i-1}, a) = q_i$. We receive the incomplete DFA depicted in Figure 3.9. Since this automaton accepts the finite language $\{a^{i-2}\}$, an equivalent minimal DFA needs precisely i states.

For $i = 1$, the transition connecting states q_1 and q_2 is deleted. The resulting incomplete DFA accepts the empty language, which is accepted by a minimal DFA with only one state.

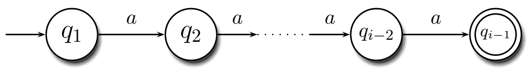


Figure 3.9: Incomplete DFA whose equivalent minimal DFA needs exactly i states.

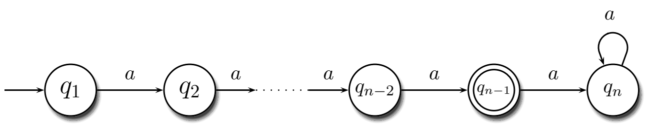


Figure 3.10: DFA that proves the existence of a minimal DFA with n states, that is equivalent to a defective automaton received for the defect delete transition.

For a state complexity of n let $\mathcal{A} = \langle Q, \{a\}, \delta, q_1, F\rangle$ be the DFA depicted in Figure 3.10, where $Q = \{q_1, q_2, \ldots, q_n\}$, $F = \{q_{n-1}\}$ and

$$\delta(q_j, a) = \begin{cases} q_{j+1} & \text{if } j \in \{1, 2, \ldots, n-1\}, \\ q_n & \text{if } j = n \end{cases} .$$

This DFA is complete. The following considerations provide the minimality of the chosen automaton.

Every state $q_j \in \{q_1, q_2, \ldots, q_n\}$ is reachable by a^{j-1}.

For indices $x, y \in \{1, 2, \ldots, n\} \setminus \{n-1\}, x < y$, we have $\delta(q_x, a^{n-1-y}) \neq q_{n-1}$ but $\delta(q_y, a^{n-1-y}) = q_{n-1}$. Thus, all the states are reachable and inequivalent.

We now modify the DFA $\mathcal{A}$ by deleting the transition $\delta(q_{n-1}, a) = q_n$. We receive the incomplete DFA from Figure 3.11. Since this automaton accepts the finite language $\{a^{n-2}\}$, the equivalent minimal DFA has n states. □

Before we turn to the next defect, we prove that the minimal DFA for the NFA depicted in Figure 3.12 needs exactly $2^n - 1$ states. We use this automaton to prove the optimality of Theorem 3.1.7.

Lemma 3.1.1. *The minimal DFA equivalent to the NFA depicted in Figure 3.12 has precisely* $2^n - 1$ *states for* $n \geq 2$.

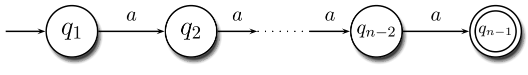


Figure 3.11: Incomplete DFA whose equivalent minimal DFA needs exactly n states.

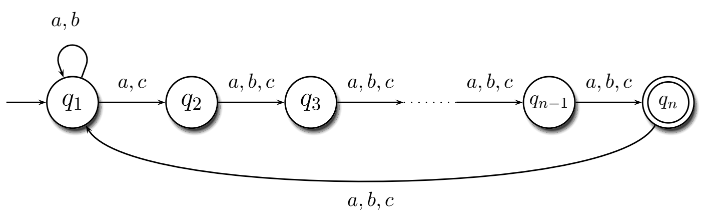


Figure 3.12: NFA whose equivalent minimal DFA needs exactly $2^n - 1$ states for $n \geq 2$.

Proof. Let $\mathcal{A} = \langle Q, \{a, b, c\}, \delta, q_1, \{q_n\}\rangle$ be the NFA with $n \geq 2$ states depicted in Figure 3.12 with state set $Q = \{q_1, q_2, \ldots, q_n\}$. The state transition function δ can be determined from the picture.

At first we consider the reachability of the states of 2^Q. The singletons $\{q_i\}$ for all $q_i \in Q$ are reachable by reading the words c^{i-1} from the initial state $\{q_1\}$. The sets $\{q_1, q_x\}$, where $x \in \{2, 3, \ldots, n\}$ can be reached by the words ab^{x-2}. By induction, the sets $\{q_1, q_{x_1}, q_{x_2}, \ldots, q_{x_k}\}$, where $k \geq 2$, $x_1, x_2, \ldots, x_k \in \{2, 3, \ldots, n\}$, and $x_1 < x_2 < \cdots < x_k$, are reachable from the set $\{q_1, q_{x_2-x_1+1}, q_{x_3-x_1+1}, \ldots, q_{x_k-x_1+1}\}$ of size $k - 2$ by reading the word ab^{x_1-2}. Therefore, all subsets of Q containing state q_1 are reachable.

The sets $\{q_{x_1}, q_{x_2}, \ldots, q_{x_k}\}$, where $k \geq 2$, and where $x_1, x_2, \ldots, x_k \in \{2, 3, \ldots, n\}$ are indices with $x_1 < x_2 < \cdots < x_k$, can also be shown to be reachable. Since all states containing q_1 are reachable we can also reach the set $\{q_{x_1}, q_{x_2}, \ldots, q_{x_k}\}$ by reading the word c^{x_1-1} from the set $\{q_1, q_{x_2-(x_1-1)}, q_{x_3-(x_1-1)}, \ldots, q_{x_k-(x_1-1)}\}$.

Thus, every non-empty set can be reached from the initial state $\{q_1\}$. The only non-reachable set from 2^Q is the empty set. This cannot be reached by any word read from the initial state $\{q_1\}$ since there exists no undefined transition in the given NFA.

In the following we show that these $2^n - 1$ states are inequivalent. Let x and y denote two arbitrary indices of states of $\mathcal{A}$ belonging to $\{1, 2, \ldots, n-1\}$, where $x < y$. In the DFA built by the power set construction the sets $\{q_x\}$ and $\{q_y\}$ can be distinguished by the word a^{n-y}. Therefore, also all the sets $S, T \subseteq \{q_1, q_2, \ldots, q_{n-1}\}$, where $S \neq T$, and $S \neq \emptyset, T \neq \emptyset$, can be distinguished.

This leaves the inequivalence of the accepting states containing q_n. Let S and T be the sets selected as above. The first case we consider is $q_{n-1} \notin S$. Then we have $\delta(S \cup \{q_n\}, a) = S' \subseteq \{q_1, q_2, \ldots, q_{n-1}\}$ and $\delta(T \cup \{q_n\}, a) = T'$ where S', T' are not empty. If $q_{n-1} \in T$ then T' contains q_n and the sets $S \cup \{q_n\}$ and $T \cup \{q_n\}$ are distinguishable.

In case that $q_{n-1} \notin T$, T' is a non-empty subset of $\{q_1, q_2, \ldots, q_{n-1}\}$ and we already know that S' and T' can be distinguished and, thus, also $S \cup \{q_n\}$ and $T \cup \{q_n\}$.

If $q_{n-1} \in S$ we have that $\delta(S \cup \{q_n\}, a) = S' \cup \{q_n\}$, where $S' \subset \{q_1, q_2, \ldots, q_{n-1}\}$, and $\delta(T \cup \{q_n\}, a) = T'$. If $q_{n-1} \notin T$, the considered sets are inequivalent. In case that $q_{n-1} \in T$, we choose $j = \max\{i \mid q_i \in S \setminus T \cup T \setminus S\}$. Without loss of generality, let $q_j \in S$. We then have $\delta(S \cup \{q_n\}, a^{n-j}) = S' \cup \{q_n\}$ where $S' \subseteq \{q_1, q_2, \ldots, q_{n-1}\}$ and $\delta(T \cup \{q_n\}, a^{n-j}) = T' \subseteq \{q_1, q_2, \ldots, q_{n-1}\}$. This proves all sets $S \cup \{q_n\}$ and $T \cup \{q_n\}$ to be inequivalent.

Summarising, we have $2^n - 1$ inequivalent and reachable states in the minimal DFA that accepts the same language as the NFA given in Figure 3.12. □

With the help of Lemma 3.1.1 we can show the following theorem.

Theorem 3.1.7. *Let $\mathcal{A}$ be a minimal DFA with $n \geq 2$ states and let the NFA $\mathcal{A}'$ be received by the defect insert transition. Then the minimal DFA equivalent to $\mathcal{A}'$ needs at most $2^n - 1$ states. This bound is tight for at least ternary alphabets.*

Proof. In the worst case the defect insert transition implements ambiguity in the automaton but does not lead to undefined transitions since the DFA affected by the defect is complete. Thus, an equivalent minimal DFA may need all states generated by the power set construction except the empty set which can only be reached in the DFA if there exists an undefined transition in the NFA. This means a complete and minimal DFA needs at most $2^n - 1$ states.

We now show that this bound can be reached by inserting a single transition in a minimal DFA with ternary alphabet. Let $\mathcal{A} = \langle Q, \{a, b, c\}, \delta, q_1, \{q_n\}\rangle$ be the DFA given in Figure 3.13, where $Q = \{q_1, q_2, \ldots, q_n\}$ and

$$\delta(q_j, a) = \begin{cases} q_1 & \text{if } j = 1 \text{ or } j = n, \\ q_{j+1} & \text{if } j \in \{2, 3, \ldots, n-1\} \end{cases},$$

$$\delta(q_j, b) = \begin{cases} q_1 & \text{if } j = 1 \text{ or } j = n, \\ q_{j+1} & \text{if } j \in \{2, 3, \ldots, n-1\} \end{cases},$$

and

$$\delta(q_j, c) = \begin{cases} q_{j+1} & \text{if } j \in \{1, 2, \ldots n-1\}, \\ q_1 & \text{if } j = n \end{cases}.$$

This DFA is complete. It is also minimal since every state $q_i \in \{q_2, q_3, \ldots, q_n\}$ can be reached from the initial state q_1 by reading ca^{n-i-1} and state q_1 is reached by reading symbol a. The states are also inequivalent, since for states $q_k, q_l \in \{q_1, q_2, \ldots, q_{n-1}\}$ with $k < l$, we have that $\delta(q_k, a^{n-l}) \neq q_n$ but $\delta(q_l, a^{n-l}) = q_n$.

We modify this DFA by inserting the transition $\delta(q_1, a) = q_2$. We receive the NFA depicted in Figure 3.12. For this automaton we in Lemma 3.1.1 it was already shown that the equivalent minimal DFA needs precisely $2^n - 1$ states. □

Since the upper bound stated in the former theorem needs a ternary alphabet, the following theorem gives the upper bound for unary alphabet. For binary alphabets, we were not able to prove a tight upper bound, yet. In this thesis, this is left over for future work.

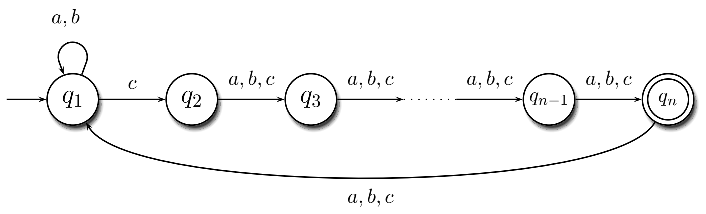


Figure 3.13: DFA for which the equivalent minimal DFA of the defective automaton with inserted transition from q_1 to q_2 on a needs $2^n - 1$ states.

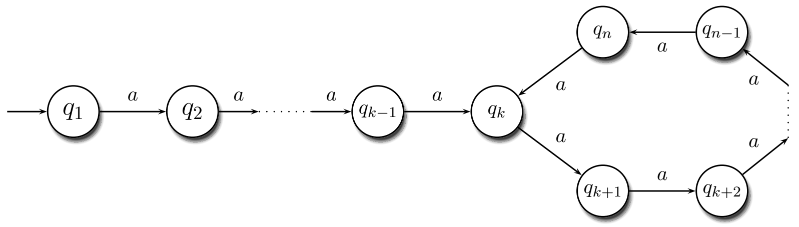


Figure 3.14: General structure of a DFA that accepts a unary regular language. The acceptance property is omitted.

Theorem 3.1.8. *Let $\mathcal{A}$ be a complete and minimal DFA with $n \geq 2$ states and unary alphabet, and let $\mathcal{A}'$ be the NFA that results from the defect insert transition. Then the equivalent minimal DFA for $\mathcal{A}'$ has at most $n^2 - 2n + 2$ states. This bound is tight.*

Proof. It is well known since Chrobak [16, 17], that minimal DFA for unary alphabets consist of a chain of states followed by a ring of states. There exists only one state belonging to both of these parts. We want to call this state the connecting state, and denote it by q_k, like in the DFA depicted in Figure 3.14.

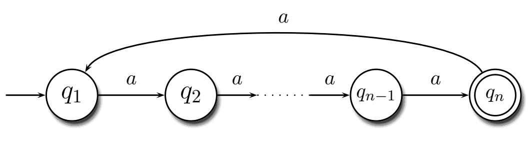


Figure 3.15: DFA for which the defect insert transition inserting a transition on a from q_2 to q_n results in a minimal DFA with $n^2 - 2n + 2$ states.

The insertion of one additional transition to one of the states causes an ambiguity for precisely this one state. Let q_a denote this state. The defective automaton consists of as many states as the original DFA since the defect does not lead to a loss of any states. Therefore, the number of states for the DFA equivalent to the defective automaton is denoted in terms of the number of states of the original DFA.

In case q_a is not reachable from all of its successors, the ambiguity can be used only once on every path leading through the automaton. For an equivalent minimal DFA constructed by the power set construction this means that the predecessors of q_a are represented by singletons and the successors of this state are represented by sets consisting of at most two states of the original DFA. The restriction to at most two states within such a set is due to that the ambiguity leads into at most two different states, from which state q_a is not reachable anymore.

Let $\mathcal{A}$ denote a unary DFA like depicted in Figure 3.14. The insertion of a transition for any state in $\{q_1, q_2, \ldots, q_{k-1}\}$ to a state with a higher index, leads to the restriction that the ambiguity can only be used once. For example, this is the case when inserting a transition from q_2 to q_4 in $\mathcal{A}$. In the power set automaton for this defective automaton, only the singletons $\{q_1\}$ and $\{q_2\}$ are reachable, and the pairs of states of the form $\{q_i, q_{i+1}\}$, where $3 \leq i \leq n-1$, and $\{q_n, q_k\}$. Once entering state $\{q_n, q_k\}$ in the power set automaton, only the pairs $\{q_i, q_{i+1}\}$ for $k \leq i \leq n-1$, and $\{q_n, q_k\}$ are reachable.

For such an inserted transition, an upper bound for the number of states for the power set automaton is given by $2n$. This includes the possibility to reach all singletons and all pairs of two consecutive states of the DFA. For pairs of non-consecutive states, the number may only decrease.

The inserted ambiguity may also be used more than only once on some paths in the automaton. In this case, the state q_a needs to be reachable from at least one of its successors. This means, there exists at least one path that starts and ends with state q_a. Inserting another transition on a in the DFA depicted in Figure 3.14 for state q_4 leading into state q_2, a path from q_4 to itself is inserted. The shortes such path is given by $q_4q_2q_3q_4$. In this example, only the singletons $\{q_1\}$, $\{q_2\}$, $\{q_3\}$, and $\{q_4\}$ are reachable in the power set automaton for $k \geq 5$.

This is because all states in the power set automaton that are reachable from $\{q_4\}$ are represented by sets of size at least two. These sets are $\{q_2, q_5\}$, $\{q_3, q_6\}$, and $\{q_4, q_7\}$. The reachable triples are $\{q_2, q_5, q_8\}$, $\{q_3, q_6, q_9\}$, and $\{q_4, q_7, q_{10}\}$, if $n = 10$. The indices differ for different numbers n, and also the number of reachable sets of a fixed size.

From this, we can deduce that the more states, also the more sets of states are existent and reachable of a fixed size between 2 and $n-1$. But since n is fixed to some number, the number of reachable sets is maximisedby minimising the length of the path connecting q_a to itself. Then the maximal number of sets of states of a fixed size is reachable in the power set automaton. This means, the insertion of $\delta(q_4, a) = q_3$ would fulfill these requirements.

If the additional transition is inserted for a state within the cycle $q_kq_{k+1}\cdots q_nq_k$, then there exist two different cycles that begin and end with state q_a. If, for example, the additional transition is inserted for state q_{k+1} leading into state q_{k+4}, there exist the cycles $q_{k+1}q_{k+2}\cdots q_nq_kq_{k+1}$, and $q_{k+1}q_{k+4}q_{k+5}\cdots q_nq_kq_{k+1}$ for state q_{k+1}. For the states within the power set automaton this means that all singletons up to $\{q_{k+1}\}$ are reachable. Also all pairs of the form $\{q_i, q_{i+2}\}$ for $k+2 \leq i \leq n-2$, and the pairs $\{q_{n-1}, q_k\}$, $\{q_n, q_{k+1}\}$ are reachable. These are $n-k-2$ many pairs. The reachable sets of three states are $\{q_i, q_{i+2}, q_{i+4}\}$ for $k \leq i \leq n-4$, and the triples $\{q_{n-3}, q_{n-1}, q_k\}$, and $\{q_{n-2}, q_n, q_{k+1}\}$. These are also $n-k-2$ sets.

From this, we can deduce that there exist $n-k-2$ numbers of reachable state sets for each size between 2 and $n-1$. To maximise this number, k needs to be minimised. This is the case, if k is equal to one. Since most of the reachable sets consist of states with indices that have the same fixed distance greater than or equal to one, the number of such sets is maximised, if this distance is fixed to one. Summarising all the maximising conditions, this means the original DFA consists only of a ring of n states and the inserted transition inserts a second cycle from q_a to itself of length $n-1$. All n singletons also need to be reachable, wherefore the state q_a needs to be the state q_n.

In the power set automaton for such a defective automaton, the rejecting sink state cannot be reached, since there do not exist any undefined transitions. Combining the results from above, there exist precisely $n-1$ reachable sets of states in the power set automaton of the defective automaton for each size i between 2 and $n-1$. All in all this gives the following sum for the number of reachable state sets:

$$n + \sum_{i=2}^{n-1} n - 1 = n^2 - 2n + 2.$$

Since it is possible that these state sets are all inequivalent, this gives an upper bound for the number of states of a complete and minimal DFA accepting the language of the defective automaton.

The upper bound is met when the defect insert transition affects the DFA depicted in Figure 3.15. Let $\mathcal{A} = \langle Q, \{a\}, \delta, q_1, \{q_n\}\rangle$ denote this DFA, where its set of states is $Q = \{q_1, q_2, \ldots, q_n\}$, and its state transition function is given by

$$\delta(q_j, a) = \begin{cases} q_{j+1} & \text{if } j \in \{1, 2, \ldots, n-1\}, \\ q_1 & \text{if } j = n \end{cases}.$$

This DFA is complete and each state $q_i \in Q$ is reachable from the initial state q_1 by the word a^{i-1}, where $1 \leq i \leq n$. The states are all inequivalent since for two states $q_i, q_j \in Q$, $1 \leq i < j \leq n-1$, the word a^{n-j} distinguishes the two states. Thus, the depicted DFA is minimal.

If the defect adds the transition $\delta(q_n, a) = q_2$ to this DFA the result is an NFA consisting of two cycles, one of length n and one of length $n-1$.

The states of the equivalent DFA resulting from the power set construction have a special structure due to the structure of $\mathcal{A}$. In this special case, for the sets consisting of more than one state, the states are consecutive. That means if the state with the smallest index within the set is q_i, where $1 \leq i \leq n-1$, then the rest of the states of the set are $q_{i+1}, q_{i+2}, \ldots, q_{i+l}$. This also includes the accepting state consisting of all the states of the NFA. The indices are to be considered modulo n plus one.

The only state containing state q_1 and not q_2 is the singleton $\{q_1\}$. All of the other state sets contain either both, only q_2 or none of these two states.

The accepting states of the power set automaton are the ones containing state q_n, the other state sets are non-accepting.

For this DFA we have that all of these sets are reachable by a certain number of a's to be read. Not all of the reachable states are inequivalent. The two states represented by the sets $\{q_2, q_3, \ldots, q_n\}$ and $\{q_1, q_2, \ldots, q_n\}$ are equivalent, since both states are accepting and by any number of a's read from each of the states the resulting state is $\{q_1, q_2, \ldots, q_n\}$. Thus, they are equivalent.

To prove the inequality of the other states, let X and Y be two arbitrary subsets of $\{q_1, q_2, \ldots, q_n\}$ representing a state of the power set automaton, where $X \neq Y$, q_n belongs to both of the sets, and none of the sets X and Y is equal to $\{q_2, q_3, \ldots, q_n\}$. Let q_x denote the state of the symmetric difference $X \triangle Y$, where we define the number $x = \max\{m \mid q_m \in X \triangle Y\}$ to be the biggest index of all states belonging to only one of the sets X and Y. This especially means $1 \leq x < n$. Without loss of generality, we assume $q_x \in X$. Then the word a^{n-x} shows the inequality of the sets X and Y. If this word is processed from set X, the power set automaton is driven into a state represented by a set containing state q_n, and if it is processed from state set Y into a state set without q_n. This proves the inequality of all accepting states of the power set DFA. The only exception are the two accepting state sets $\{q_2, q_3, \ldots, q_n\}$ and $\{q_1, q_2, \ldots, q_n\}$, which were already proven to be equivalent.

This leaves the proof of the inequality of the reachable non-accepting states of the power set automaton. For this, let $X, Y \subseteq \{q_1, q_2, \ldots, q_{n-1}\}$ now be two arbitrary, non-accepting state sets, where $X \neq Y$. These sets consist of consecutive states since they do not contain state q_n. Let $X = \{q_i, q_{i+1}, \ldots, q_{i+k}\}$ and let $Y = \{q_j, q_{j+1}, \ldots, q_{j+l}\}$, where $1 \leq i, j \leq n-2$, $k, l \geq 0$, and $q_{i+k} \neq q_n, q_{j+l} \neq q_n$. Since these two sets are not equal and consist of consecutive states, their intersection $X \cap Y$ is either empty or contains states, that belong to both of the sets. In the first case, there exists a state q_x in one of the sets X or Y, for which $x = \max\{m \mid q_m \in X \cup Y\}$ is the biggest index of all states belonging to either X or Y. Without loss of generality, we can assume $q_x \in X$. Then the word a^{n-x} proves the inequality of the sets X and Y. If this word is read from set X the automaton is driven into a state containing state q_n, and read from state set Y into a state without q_n.

In case that $X \cap Y$ is not empty, there exists a state q_x in this intersection, where $x = \max\{m \mid q_m \in X \cap Y\}$ is the biggest index belonging to both sets. Processing the word a^{n-x} from X and Y, the power set automaton is driven into two different accepting states, since $X \neq Y$. These accepting states cannot be the two equivalent accepting sets $\{q_2, q_3, \ldots, q_n\}$ and $\{q_1, q_2, \ldots, q_n\}$, since the latter one is only reachable from the former one. The inequivalence of the accepting states also proves the inequivalence of X and Y.

The total number of states for a minimal DFA accepting the same language as the defective automaton constructed above, is given by summing up the numbers of all the reachable state sets of a fixed size between 1 and n that were proven to be inequivalent. This minimal DFA has two state less than the DFA received from the power set construction, since $\{q_2, q_3, \ldots, q_n\}$ and $\{q_1, q_2, \ldots, q_n\}$ are the only equivalent states, and the empty set does not represent one of the reachable states. The rest of the unreachable states were already omitted.

This DFA consists of precisely n different singletons. For each size $2 \leq i \leq n-1$, the minimal DFA contains a state represented by a state set of exactly size i. For each of these sizes there exist exactly $n-1$ such sets. The set $\{q_2, q_3, \ldots, q_n\}$ is equivalent to the last state also belonging to the power set automaton, which is represented by the state set Q. The following sum then gives the number of all states of the minimal DFA for the considered defective automaton:

$$n + 1 + \left(\left(\sum_{i=2}^{n-1} n - 1 \right) - 1 \right) = n + (n-2)(n-1) = n^2 - 2n + 2.$$

This concludes the proof. □

3.2 Defects Occurring on the States

This section deals with defects occurring on the states of deterministic finite automata. The following theorem covers the defect of deletions of accepting states of a given DFA and of all transitions leading into these states and the ones leaving these states.

Theorem 3.2.1. *Let $\mathcal{A}$ be a minimal DFA with $n \geq 2$ states and let the NFA $\mathcal{A}'$ be received by the defect delete accepting state affecting $\mathcal{A}$. Then the equivalent minimal DFA for $\mathcal{A}'$ needs at most n states. This bound is tight even for unary alphabets.*

Proof. For this defect the upper bound is n since the deletion of an accepting state and all its incoming and outgoing transitions only leads to undefined transitions. It does not insert any ambiguity to the automaton. Thus, to receive an equivalent minimal DFA it is sufficient to add a rejecting sink state if it does not already exist.

We show that the automaton given in Figure 3.16 meets the upper bound when deleting the accepting state n. Let $\mathcal{A} = \langle Q, \{a\}, \delta, q_1, \{q_{n-1}, q_n\} \rangle$ denote the depicted DFA, where $Q = \{q_1, q_2, \ldots, q_n\}$ is the set of states, and

$$\delta(q_j, a) = \begin{cases} q_{j+1} & \text{if } j \in \{1, 2, \ldots, n-1\}, \\ q_1 & \text{if } j = n \end{cases}$$

defines its state transition function.

Deleting the accepting state q_n and all transitions leading into or going out of this state results in an NFA that accepts the finite language $\{a^{n-2}\}$. The complete and minimal DFA accepting this language needs n states. □

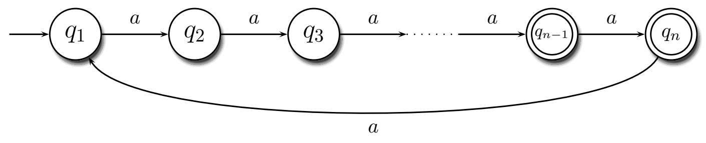


Figure 3.16: DFA for the proof of the optimality of the upper bound for the defect delete accepting state.

The theorem below deals with the question for magic numbers for the defect delet accepting state.

Theorem 3.2.2. *Let $n \geq 2$ be a number. Then there exist minimal DFA A with n states, such that by the defect delete accepting state, the equivalent DFA for the defective automaton needs precisely a number of i states, for all $1 \leq i \leq n$. This is true even for unary alphabets.*

Proof. For $i = n$ we have already shown in the proof of Theorem 3.2.1 that there exists a DFA that meets the upper bound of n states.

Let $1 \leq i \leq n-1$ and $\mathcal{A} = \langle Q, \{a\}, \delta, q_1, F\rangle$ be the DFA depicted in Figure 3.8 with state set $Q = \{q_1, q_2, \ldots, q_n\}$, set of accepting states $F = \{q_{i-1}, q_n\}$ where $i-1 = 0$ means that q_n is the only accepting state of this automaton, and

$$\delta(q_j, a) = \begin{cases} q_{j+1} & \text{if } j \in \{1, 2, \ldots, n-1\}, \\ q_n & \text{if } j = n \end{cases}.$$

This DFA is complete and minimal, which was alredy used in the proof of Theorem 3.1.6.

If the considered defect affecting DFA $\mathcal{A}$ deletes the accepting state q_n and all transitions having q_n as source or target, we receive an incomplete DFA that accepts the finite language $\{a^{i-2}\}$. For this language it is well known that the accepting minimal DFA consists of precisely i states. □

The next possible defect on states we want to consider is the defect delete non-accepting state. In the following theorem an upper bound for the number of states of an equivalent minimal DFA for the defective automaton is presented.

Theorem 3.2.3. *Let $\mathcal{A}$ be a minimal DFA with $n \geq 2$ states and let $\mathcal{A}'$ denote the NFA that is received by the defect delete non-accepting state affecting $\mathcal{A}$. Then the equivalent minimal DFA of $\mathcal{A}'$ has at most n states. This bound is tight even for unary alphabet.*

Proof. Deletion of one non-accepting state and all transitions leading into and leaving this state in a minimal DFA with $n \geq 2$ states does not lead to ambiguity, only to undefined transitions. Thus, this defect results in an incomplete DFA. By adding a rejecting sink state if it does not already exist, the upper bound for the number of states of an equivalent minimal DFA is given by n.

For the proof of tightness of this upper bound, we use the automaton depicted in Figure 3.10 already used in Theorem 3.1.6. Let $\mathcal{A} = \langle Q, \{a\}, \delta, q_1, \{q_{n-1}\}\rangle$ be this DFA for which the state set is $Q = \{q_1, q_2, \ldots, q_n\}$, and its state transition function $\delta : Q \times \{a\} \to Q$ is defined as follows:

$$\delta(q_j, a) = \begin{cases} q_{j+1} & \text{if } j \in \{1, 2, \ldots, n-1\}, \\ q_n & \text{if } j = n \end{cases}.$$

The accepted language is $\{a^{n-2}\}$ and it is well known, that automaton $\mathcal{A}$ is the minimal one accepting this language.

If the defect delete non-accepting state affects state q_n, this results in an incomplete DFA, since q_n is a rejecting sink state. The language accepted by this defective automaton still is $\{a^{n-2}\}$. This shows the tightness of the upper bound, since the minimal DFA accepting this language still needs n states. □

Also for this defect, the question for magic numbers arises in a natural way. The following theorem gives the answer to this question.

Theorem 3.2.4. *Let $n \geq 2$ be a number. Then there exist minimal DFA $\mathcal{A}$ with n states, such that by the defect delete non-accepting state, the equivalent DFA for the defective automaton needs precisely a number of i states, for all $1 \leq i \leq n$. This is true even for unary alphabets.*

Proof. The existence of such an automaton for $i = n$ was already shown in the proof of Theorem 3.2.3. It was used to show that the upper bound of n states for a minimal DFA accepting the language of the defective automaton is met.

For the remaining numbers of states $1 \leq i \leq n-1$, we use the automaton depicted in Figure 3.8. This automaton was already stated to be a minimal DFA in the proof of Theorem 3.1.6. If $n = 2$, this DFA consists of the non-accepting state q_1, and the accepting state q_2. The non-accepting state q_{i-1} does not exist in this automaton.

When deleting state q_i of this automaton for i between 1 and $n-1$, the resulting automaton is disconnected from all the states q_{i+1} up to state q_n. Then the only remaining accepting state is q_{i-1}. In case that $i = 1$ this means, there is no state left at all, especially no accepting state. For each $i \geq 2$, the accepted language is $\{a^{i-2}\}$, where $a^0 = \lambda$. For this language it is well known, that a recognising minimal DFA has exactly i states.

The case $i = 1$ still needs to be considered. The defective automaton does not consist of any state. This means, the accepted language is the empty language, which is accepted by a minimal DFA with only one state, which is a rejecting sink state. This completes the proof. □

Theorems 3.2.1, 3.2.2, 3.2.3 and 3.2.4 show that the defects deleting any type of state in a minimal DFA with $n \geq 2$ states may result in an automaton accepting a language such that a minimal DFA accepting the same language needs a number of states between 1 and n.

Another defect to consider for automata is the loss of the accepting property for a state within a minimal DFA. The following theorems will cover some of the questions concerning the state complexity of this type of defect.

Theorem 3.2.5. *Let $\mathcal{A}$ be a minimal DFA with $n \geq 3$ states and let $\mathcal{A}'$ denote the NFA that is received by the defect remove acceptance. Then the equivalent minimal DFA for $\mathcal{A}'$ needs at most n states. This bound is tight even for unary alphabets.*

Proof. In case there only exists one accepting state in the DFA the defect results in a DFA accepting the empty language. In this case n is an upper bound for the state complexity of a minimal DFA accepting the empty language. Considering a DFA $\mathcal{A}$ with at least two accepting states, the defect does not affect the completeness property of the automaton since there exists no added ambiguity or results in any undefined transition. It is even possible that the minimality is preserved. The only difference between the two automata is the accepted language since the defective one accepts less words than the other one. Thus, in any case there is no need to add new states to the defective automaton $\mathcal{A}'$. This is why at most n states are needed for an equivalent minimal DFA.

To prove the tightness of this bound for $n \geq 3$, we use the automaton depicted in Figure 3.8 again, which we already used in the proof of Theorem 3.1.6. The accepting states of this automaton are chosen to be q_{n-2} and q_n, which means $i = n - 1$.

In case the defect only affects the acceptance of state q_{n-2} of this automaton, the defective automaton still is complete and minimal, since it accepts the language $\{a\}^{n-1}\{a\}^*$. The minimal DFA accepting this language is already the defective automaton. Since this DFA consists of n states, the statement of the theorem is proven. □

As for the former defects affecting states, the next theorem covers the question for the possible state complexities of minimal DFA accepting the language of the defective automaton for the defect remove acceptance.

Theorem 3.2.6. *Let $n \geq 3$ be a number. Then there exist minimal DFA A with n states, such that by the defect remove acceptance, the equivalent DFA for the defective automaton needs precisely a number of i states, for all $1 \leq i \leq n$. This is true even for unary alphabets.*

Proof. Again, we use the automaton $\mathcal{A}$ depicted in Figure 3.8 for $n \geq 3$.

For $2 \leq i \leq n$, if the defect affects state q_{i-1} of this automaton, the resulting automaton accepts the language $\{a\}^{n-1}\{a\}^*$, which is accepted by a complete and minimal DFA with exactly n states. This was already shown to prove the tightness of the upper bound in Theorem 3.2.5.

To show the existence of a DFA with a number of states i between 1 and $n - 1$, that is equivalent to a defective automaton, we assume that the defect affects the acceptance property of state q_n of $\mathcal{A}$. For $i \geq 2$, the accepted language is reduced

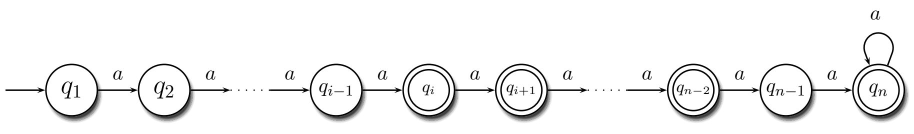


Figure 3.17: DFA used in the proof for the upper bound for the defect of inserting the accepting property to a non-accepting state.

to the single word a^{i-2}. The language consisting of this word only, is accepted by a minimal DFA with exactly i states.

For $i = 1$, state q_{i-1} does not exist at all and the language accepted by the defective automaton is empty. Since this language is accepted by a minimal DFA with one state, the proof is completed. □

The last defect to consider is the defect add acceptance. The following theorems will cover some questions concerning the state complexity for this type of defect.

Theorem 3.2.7. *Let $\mathcal{A}$ be a minimal DFA with $n \geq 3$ states, and let $\mathcal{A}'$ be received from $\mathcal{A}$ by the defect add acceptance. Then the equivalent minimal DFA of $\mathcal{A}'$ needs at most n states. This bound is tight even for unary alphabet.*

Proof. Adding acceptance to already existing non-accepting states in a minimal DFA never leads to undefined or ambiguous transitions. It also does not lead to less states. This defect can only result in a violation of the minimality of the given finite automaton. Thus, it is possible that the minimal DFA accepting the same language as the defective automaton already is the defective one. All these considerations lead to an upper bound of n states for the equivalent DFA.

The optimality of this upper bound is shown by letting the defect affect the minimal DFA depicted in Figure 3.5 we already used to show the optimality of the upper bound for the defect flip transition.

For $n \geq 3$ states, this automaton consists of one accepting and at least two non-accepting states. The accepted language consists of all words over the alphabet $\{a\}$ for which the length is $kn + n - 1$ for some integer $k \geq 0$. The adding of acceptance for one of the non-accepting states of the considered DFA does not lead to any ambiguity or undefined transitions to the automaton. This defect may only affect the inequivalence of the states, and, thus, the minimality of the DFA. We assume that the defect affects state q_{n-1}. Then the two accepting states can be distinguished by the word a, and the defective automaton still is a minimal DFA with n states. □

Lemma 3.2.1. *The finite automaton depicted in Figure 3.17 is a minimal DFA for $n \geq 3$.*

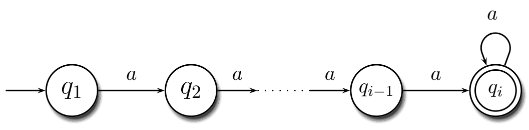


Figure 3.18: Minimal DFA with unary alphabet and i states.

Proof. Let the depicted DFA be denoted by $\mathcal{A} = \langle Q, \{a\}, \delta, q_1, F\rangle$, with state set $Q = \{q_1, q_2, \ldots, q_n\}$, set of accepting states $F = \{q_i, q_{i+1}, \ldots, q_{n-2}, q_n\}$, and state transition function $\delta : Q \times \{a\} \to Q$ defined as follows:

$$\delta(q_j, a) = \begin{cases} q_{j+1} & \text{if } j \in \{1, 2, \ldots, n-1\}, \\ q_n & \text{if } j = n. \end{cases}$$

This automaton $\mathcal{A}$ is a minimal DFA. This is because a state q_j is reachable from the initial state q_1 by the word a^{j-1}. The states $q_1, q_2, \ldots, q_{i-1}$ and also q_{n-1} are inequivalent to the states $q_i, q_{i+1}, \ldots, q_{n-2}$ and q_n. Two states q_j and q_k from the subset $\{q_i, q_{i+1}, \ldots, q_{n-2}, q_n\}$, with $j < k$, can be shown to be inequivalent by the word a^{n-j}, since $\delta(q_j, a^{n-j}) = q_{n-1} \notin F$, but $\delta(q_k, a^{n-j}) = q_n \in F$.

Two states q_j and q_k from the set $\{q_1, q_2, \ldots, q_{i-2}\}$, where $j < k$, are inequivalent since $\delta(q_j, a^{n-j-1}) = q_{n-1} \notin F$ and $\delta(q_k, a^{n-j-1}) = q_n \in F$. Any state q_k of this subset of states can be proven to be inequivalent to the states q_{i-1} and q_{n-1} by the word a.

The only states left to show to be inequivalent are the states q_{i-1} and q_{n-1}. We have that $\delta(q_{i-1}, a^{n-i}) = q_{n-1} \notin F$ and $\delta(q_{n-1}, a^{n-i}) = q_n \in F$.

Summarising the results, we have that all states of automaton $\mathcal{A}$ are reachable from the initial state q_1. They are also inequivalent. Thus, the DFA depicted in Figure 3.17 is a minimal DFA. □

In the following theorem the answer is given to the question for possible numbers of states that a minimal DFA needs to accept the language of the defective automaton for the defect add acceptance.

Theorem 3.2.8. *Let $n \geq 3$ be a number. Then there exist minimal DFA $\mathcal{A}$ with n states, such that by the defect add acceptance affecting A, the equivalent DFA for the defective automaton needs precisely a number of i states, for all $1 \leq i \leq n$. This is true even for unary alphabets.*

Proof. We use the automaton depicted in Figure 3.17, which was shown to be minimal in Lemma 3.2.1.

In case the defect affects state q_{n-1} of this automaton the resulting automaton accepts the language $\{a\}^{i-1}\{a\}^*$, which is accepted by the minimal DFA with exactly i states depicted in Figure 3.18. This automaton is complete and each state q_j

	Reg. languages	Unary reg. languages
Delete Transition	$n+1$	$n+1$
Insert Transition	2^n-1	n^2-2n+2
Flip Transition	2^n	$n+1$
Exchange Symbol	2^n	n

Table 3.1: Bounds for defects affecting the transitions of a minimal DFA. For unary alphabets, these bounds are tight. For arbitrary regular languages, these bound are also tight already for binary alphabets, except for the defect insert transition. For this defect, the alphabet is at least ternary.

	Reg. languages	Unary reg. languages
Delete Non-Acc. State	n	n
Delete Accepting State	n	n
Remove Acceptance	n	n
Add Acceptance	n	n

Table 3.2: Tight bounds for defects affecting the states of a minimal DFA with n states.

is reachable from the initial state by the word a^{j-1}. Two states q_k and q_r belonging to the subset $\{q_1, q_2, \ldots, q_{i-1}\}$ are inequivalent since for $k < l$ it is true that $\delta(q_k, a^{i-k-1}) = q_{i-1}$, which is a non-accepting state, and $\delta(q_r, a^{i-k-1}) = q_i$, which is an accepting state. This completes the proof. □

Tables 3.1 and 3.2 summarise the results of this section.

3.3 Defects in DFA Accepting Finite Languages

This section covers the question of the state complexity for equivalent minimal DFA for a defective automaton that is the result of a defect affecting a minimal DFA accepting a finite language.

In general, results concerning the state complexity for finite languages differ from those for arbitrary regular languages. In [11] it was shown that some bounds of the state complexity for simple operations like for the concatenation of two finite languages are strictly lower for finite languages than the one for general regular languages. In some sence, a defect may be considered as an operation that is not applied to the regular language directly, but to the deterministic finite automaton that accepts the language. This motivates the separate investigation of the finite languages also for defects.

The first result proves an upper bound for the defect exchange symbol.

Theorem 3.3.1. *Let $\mathcal{A}$ be a minimal DFA with $n \geq 2$ states that accepts a finite language, and let $\mathcal{A}'$ be the NFA that results from the defect exchange symbol. Then the equivalent minimal DFA of $\mathcal{A}'$ needs at most $\frac{n^2}{2} - \frac{n}{2} + 1$ states.*

Proof. Let $Q = \{q_1, q_2, \ldots, q_n\}$ denote the state set of the minimal DFA $\mathcal{A}$ accepting a finite language. Since the accepted language is finite and the DFA is complete there exists a rejecting sink state. Without loss of generality, let q_n denote this sink state. In such a DFA, there also exists one accepting state for which each transition leads into this sink state. Let this state be denoted by q_{n-1}. All of the other accepting or non-accepting states of the DFA have at least one transition that does not lead into this sink state q_n. If there would exist another accepting state for which all transitions lead into the rejecting sink state, this state and q_{n-1} would be equivalent which is a contradiction to the minimality of DFA $\mathcal{A}$. Similar is true for any non-accepting state q with transitions leading only into q_n. Then q and q_n are equivalent. In addition to these facts, there also do not exist any cycles or loops in such a DFA except for one loop on the rejecting sink state. Otherwise the accepted language would not be finite anymore.

Due to the last fact, the states of such a DFA can be ordered by the maximal number of symbols that need to be read to reach state q_{n-1}. This property is unique for each state of the DFA. The only state that cannot be ordered in this way is the rejecting sink state q_n, because there exists no path from q_n to q_{n-1}. Therefore, this state is not considered when ordering the states.

Let $k = \max\{|w| \mid w \in L(\mathcal{A})\}$ denote the length of the longest word accepted by DFA $\mathcal{A}$. Then the initial state has distance k from state q_{n-1} since that many symbols have to be read from the initial state to reach q_{n-1}. This is because the longest words need to be accepted by q_{n-1}. Otherwise they would be accepted by another accepting state for which there exists a transition leading into a different state than the sink state. For this state there needs to exist a path to another accepting state, since otherwise DFA $\mathcal{A}$ would not be minimal. But this would be a contradiction to the choice of k.

The other states all have a distance smaller than k to state q_{n-1} since everytime a transition is performed the target state has a smaller distance to state q_{n-1} than the source state. And there always exists at least one transition from a state with distance $1 \leq i \leq k$ to a state with distance $i - 1$ since the longest words need to be accepted.

Let the states having the same longest distance $0 \leq i \leq k$ to q_{n-1} be collected in levels named by their longest distance i. By the facts from above, we can derive that there always exists at least one transition from a level $1 \leq j \leq k$ into the level $j - 1$. It is impossible to have transitions leading from a state of one level to another state in the same level. There also do not exist transitions leading from one level into a level with a bigger index.

Now it is possible to rename the states such that the states of level j always have a smaller index than the ones in the level $j-1$ and levels with smaller indices for all levels $j \in \{1, 2, \ldots, k\}$. By this, the smallest indices are found in the level k and the ones with the biggest indices in level 0.

If in this automaton for exactly one transition the symbol of the transition is exchanged by another one, an ambiguity for only one state is inserted. Any path leading through this defective automaton can visit this state at most once. Thus, the ambiguity is used at most once. For the DFA resulting from the power set construction this means that at most all singletons may be reachable and at most all pairs of states. But not all pairs are possible to reach due to the structure of the defective automaton. The only reachable pairs in the power set automaton consist of states that are reachable from the state that is the source of the defective transition by a path that uses this defective transition. This especially means for a pair of states $\{q_i, q_j\}$, where $i < j$, that in case q_i is on level $k \geq 1$ and q_j on level $l \geq 1$ the successor state in the power set automaton consists of either two states, one from a level $\leq k-1$ and one from a level $\leq l-1$, or of only one state from a level smaller than or equal to $\min\{k-1, l-1\}$.

With this knowledge it is possible to sum up all of the possible numbers of reachable states. There exist $n-1$ possible singletons plus the rejecting sink state. For the pairs there exist the following possible combinations for a state $q_i \in \{q_1, q_2, \ldots, q_{n-1}\}$ of $\mathcal{A}$: It is possible that q_i appears in a pair with any of the other states with a bigger index $> i$. This is due to the renumbering of the states depending on the levels. Thus, there may exist $n-i-1$ pairs containing state q_i. So there may exist at most $n-1-1 = n-2$ pairs containing state q_1, at most $n-2-1 = n-3$ pairs containing state q_2, and so on up to at most $n-(n-2)-1 = 1$ pairs containing state q_{n-2}, and $n-(n-1)-1 = 0$ pairs containing state q_{n-1}. In this listing, all combinations are mentioned and counted only once.

Adding up all the possible numbers of states, the following sum gives the upper bound:

$$n + (n-2) + (n-3) + \cdots + 1 = n + \sum_{i=1}^{n-2} i = \frac{n^2}{2} - \frac{n}{2} + 1$$

□

To prove the tightness of this upper bound, there must exist a possibility within the defective automaton to reach every state from the initial state. In the automaton there also need to exist transitions that can enlarge the distance between two states. For the DFA gained by the power set construction this means that from a pair of consecutive states it is possible to reach another pair of states that are not consecutive anymore but have a fixed distance. This is necessary to have as many as possible reachable pairs within the power set automaton.

The next theorem proves a lower bound for the defect of exchanging the symbol of a transition. The provided lower bound and the upper bound proved in Theorem 3.3.1 do not match but are in the same order of magnitude.

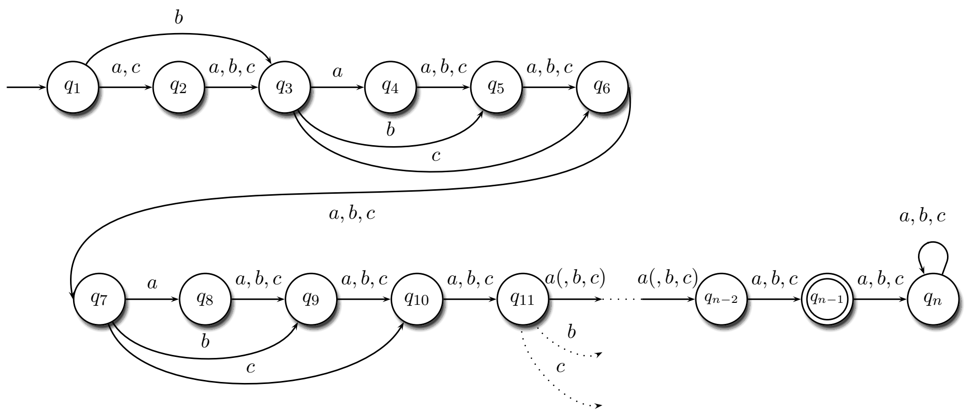


Figure 3.19: DFA for which the defect exchange symbol affecting the transition of q_1 on b by exchanging it into an a results in a minimal DFA with $O(n^2)$ states.

Theorem 3.3.2. *For $n \geq 2$, the upper bound of $\frac{n^2}{2} - \frac{n}{2} + 1$ states for a minimal DFA accepting the same language as an automaton that results from a DFA affected by the defect exchange symbol is tight in the order of magnitude.*

Proof. Let $\mathcal{A} = \langle Q, \Sigma, \delta, q_1, \{q_n\}\rangle$ denote the DFA depicted in Figure 3.19, where $Q = \{q_1, q_2, \ldots, q_n\}$ is its set of states, $\Sigma = \{a, b, c\}$ is its alphabet, and its state transition function is given by

$$\delta(q_j, a) = \begin{cases} q_{j+1} & \text{if } j \in \{1, 2, \ldots, n-1\}, \\ q_n & \text{if } j = n \end{cases},$$

$$\delta(q_j, b) = \begin{cases} q_3 & \text{if } j = 1, \\ q_{j+1} & \text{if } j = 2, j = 4 + 4k, j = 5 + 4k, \text{ or } j = 6 + 4k, \\ q_{j+2} & \text{if } j = 3 + 4k, \\ q_n & \text{if } j = n \end{cases},$$

and

$$\delta(q_j, c) = \begin{cases} q_{j+1} & \text{if } j = 1, j = 2, j = 4 + 4k, j = 5 + 4k, \text{ or } j = 6 + 4k, \\ q_{j+3} & \text{if } j = 3 + 4k, \\ q_n & \text{if } j = n \end{cases},$$

where $k \geq 0$. In case the target state for a transition on b or c of a state q_i does not exist due to the size n, the transition leads into state q_{i+1}.

This automaton is complete. Each of the states $q_i \in Q$ is reachable from the initial state q_1 by the word a^{i-1}. The inequivalence of two different states q_i and q_j of $\mathcal{A}$, where $i < j$, is proven by the word a^{n-1-i}. The DFA accepts when this word is read from the state with the smaller index, and rejects when it is read from the other state. This proves the minimality of $\mathcal{A}$.

In case that the defect exchange symbol affects the transition between the states q_1 and q_3 on symbol b, and b is replaced by a, in the resulting automaton there exists an ambiguity on a and one undefined transition on b for state q_1. In the following we will show that the equivalent minimal DFA for this automaton needs $O(n^2)$ states, which proves the tightness of the upper bound shown in Theorem 3.3.1 in the order of magnitude.

The power set construction applied to this defective automaton gives a DFA with states that are represented by sets consisting of at most two states of the defective automaton. All of the other possible subsets of Q cannot be reached in the power set automaton, since state q_1 is not reachable from any state of $\mathcal{A}$. Therefore, this ambiguity can only be used once. All possible pairs do not contain state q_1. For each state of such a pair, the transition in $\mathcal{A}$ changes the state deterministically.

Since the states in the singletons and pairs are the same states as in DFA $\mathcal{A}$ these sets are subsets of $Q = \{q_1, q_2, \ldots, q_n\}$. All pairs containing state q_n that represent a state of the power set automaton are equivalent to the singleton composed of only the state of the pair unequal to q_n. This is due to the fact that q_n is the rejecting sink state of the defective automaton. Since the initial state q_1 also cannot be part of any pair, the pairs are subsets of $\{q_2, q_3, \ldots, q_{n-1}\}$.

The structure of the defective automaton resulting from the DFA $\mathcal{A}$ depicted in Figure 3.19 causes the pairs to consist of two states of this automaton with a fixed distance. The minimal distance is 1, the maximal distance is $\lfloor \frac{n-1}{2} \rfloor$ if q_{n-1} is reachable from a state with index $5+4k$ or $6+4k$ for some $k \geq 0$ by a symbol b or c, or it is $\lfloor \frac{n-1}{2} \rfloor - 1$ if $n-1$ is computable by $4+4k$ for some $k \geq 0$. The structure is also the reason for the property that the bigger the distance of the states within the pairs gets the higher is the index of the state with the smaller index.

The reachable pairs representing states of the power set automaton are all pairs $\{q_i, q_{i+1}\} \subset \{q_2, q_3, \ldots, q_{n-1}\}$. They are reachable by the words a^{i-1}. Reachable pairs with bigger distances $l \geq 2$ are of the form $\{3+2k+i, 3+2k+i+l\}$ for even distances and $\{3+2k+i, 3+2k+i+l+1\}$ for odd distances, where $i, k \geq 0$. The pairs with even distances are reached by the words $(ac)^{\frac{l}{2}-1}aba^i$, and the pairs with odd distances by the words $(ac)^{\frac{l-1}{2}}a^i$.

No other pairs of states are reachable since the distances are produced by the transitions on b and c by the states with an index computable by $3+4k$, where $k \geq 0$. The distance can be increased on each of these states by either one or two, depending on the transition used for the state. Therefore, it is impossible to have a big distance already for small indices of the states contained in the pair.

The singletons $\{q_i\}$ for $q_i \in \{q_1, q_2, \ldots, q_n\}$ are all reachable by the words ca^{i-2}.

This leaves the proof of the inequality of the reachable states of the power set automaton to show its minimality. All the singletons $\{q_i\} \neq \{q_j\}$ from the set $\{q_1, q_2, \ldots, q_{n-2}\}$ are inequivalent because if $i < j$, the word a^{n-i-1} computed from state $\{q_i\}$ leads into acceptance, and into the rejecting sink state when read from state $\{q_j\}$.

All the reachable non-accepting states represented by two different pairs $\{q_{i_1}, q_{i_2}\}$ and $\{q_{j_1}, q_{j_2}\}$, can be distinguished by the word a^{n-1-i_1}, when $i_1 = \min\{i_1, i_2, j_1, j_2\}$.

To show the inequality of any the non-accepting singletons and any of the non-accepting pairs, let $\{q_i\}$ denote such a reachable singleton and $\{q_{j_1}, q_{j_2}\}$ such a pair. Then the word a^{n-1-m} distinguishes the states, where $m = \min\{i, j_1, j_2\}$ is the smallest of all three indices.

The accepting states can also be shown to be inequivalent. Let $\{q_i, q_{n-1}\}$ and $\{q_j, q_{n-1}\}$ denote two different accepting pairs, and $i < j$. These pairs can be differentiated by the word a^{n-1-i}. This word also differentiates any pair $\{q_i, q_{n-1}\}$ from the accepting singleton $\{q_{n-1}\}$.

This concludes the proof of the minimality of the considered power set automaton and leaves the counting of the states of this DFA. There exist n singletons, one of them represents the rejecting sink state of the DFA. The number of pairs with distance one is $n-3$ since the first such pair is $\{q_2, q_3\}$ from which all of the other such pairs are reached by reading a certain amount of a's.

For the number of pairs of states of the form $\{q_i, q_j\} \subseteq \{q_1, q_2, \ldots, q_{n-1}\}$ having a distance $l \geq 2$ we differentiate between even and odd distances. For even distances $l \geq 2$ there exist exactly $n-1-2l$ pairs reachable, and for odd distances there exist $n-1-2l+1$ such pairs in the minimal DFA accepting the same language as the defective automaton.

In case the maximal distance for the pairs is given by $\lfloor \frac{n-1}{2} \rfloor$, we differentiate between even and odd $n-1$. In case $n-1$ is even, the maximal distance is $\frac{n-1}{2}$. Then the number of states is bounded from below by the following sum:

$$\sum_{l=0}^{\frac{n-1}{2}} n-1-2l = \frac{n^2-1}{4} \in \Theta(n^2)$$

A quite similar lower bound is received for odd $n-1$, where the maximal distance is $\frac{n-1}{2} - 1$.

$$\sum_{l=0}^{\frac{n-1}{2}-1} n-1-2l = \frac{n^2-1}{4} \in \Theta(n^2)$$

If the maximal distance for the pairs is $\lfloor \frac{n-1}{2} \rfloor - 1$, $n-1$ is always even. So the maximal distance is $\frac{n-1}{2} - 1$. Then a lower bound is the same as for odd $n-1$ and the other maximal distance.

This proves that the number of pairs is already at least quadratic.

Considering an upper bound for the number of pairs, the sums from above can be built over the numbers $n-1-2l+1$. This gives a quadratic upper bound. Since the number of singletons is only n, this proves the tightness in the order of magnitude of the upper bound proved in Theorem 3.3.1. □

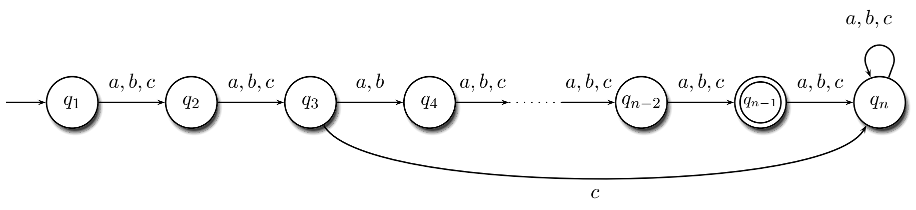


Figure 3.20: DFA for which the defect flip transition affecting the transition of q_1 to q_2 for a results in a minimal DFA with $\Omega(2^n)$ states.

Having a closer look at the structure of the DFA depicted in Figure 3.19, the transitions on the symbols b and c for the states q_3, q_7, and so on are responsible for the reachability of the pairs in the power set automaton with a distance greater than one. The bigger the distance, the bigger is the index of the smallest state having such a distance to another state. If there exist more symbols in the alphabet, it is possible to receive bigger distances much earlier in the automaton. This results in more reachable pairs for the power set automaton and a better lower bound for the defect of exchanging the symbol of a transition. The best result would give an alphabet of a size near to the number of states. But since this number is not fixed, this is not possible.

In the following theorem, we concentrate on the defect flip transition affecting DFA that accept a finite language.

Theorem 3.3.3. *Let $\mathcal{A}$ be a complete and minimal DFA with $n \geq 2$ states that accepts a finite language and let $\mathcal{A}'$ be the NFA that is received from the defect flip transition. Then the equivalent minimal DFA for $\mathcal{A}'$ needs at most 2^n states. This bound is tight in the order of magnitude for at least ternary alphabets.*

Proof. The interchange of the source and the target of a transition introduces an ambiguity to the automaton and results in an undefined transition. In the worst case, this leads to an NFA, for which the equivalent minimal DFA has 2^n states. Therefore, we can assume the upper bound to be the one for arbitrary regular languages, which is 2^n.

To prove the optimality of this bound at least in the order of magnitude, we give a lower bound based on the DFA depicted in Figure 3.20. Let $\mathcal{A} = \langle Q, \Sigma, \delta, q_1, \{q_{n-1}\}\rangle$ denote this DFA, where $Q = \{q_1, q_2, \ldots, q_n\}$ is its set of states, $\Sigma = \{a, b, c\}$ is the underlying alphabet, and its transition function $\delta : Q \times \Sigma \to Q$ is given by

$$\delta(q_j, a) = \begin{cases} q_{j+1} & \text{if } j \in \{1, 2, \ldots, n-1\}, \\ q_n & \text{if } j = n, \end{cases}$$

$$\delta(q_j, b) = \begin{cases} q_{j+1} & \text{if } j \in \{1, 2, \ldots, n-1\}, \\ q_n & \text{if } j = n, \end{cases}$$

and

$$\delta(q_j, c) = \begin{cases} q_{j+1} & \text{if } j \in \{1, 2, 4, 5, \ldots, n-1\}, \\ q_n & \text{if } j \in \{3, n\} \end{cases}.$$

This DFA is minimal since each state $q_i \in Q$ is reachable from the initial state q_1 by reading the word a^{i-1}, and two different states $q_i, q_j \in Q$ can be proved to be inequivalent by the word a^{n-1-i}.

In case the considered defect affects the transition between states q_1 and q_2 on the symbol a, the accepted language of the resulting automaton is infinite. The power set construction produces a DFA with states of a special structure. First of all, each of the singletons is reachable by words consisting of only b's.

Let $\{q_{i_1}, q_{i_2}, \ldots, q_{i_k}\} \subset \{q_1, q_2, \ldots, q_{n-1}\}$ be a set representing a state of the power set automaton, where $1 \leq i_1 < i_2 < \cdots < i_k \leq n-1$, and $k \geq 2$. The first such state reachable from the initial state $\{q_1\}$ of the power set automaton is the state $\{q_1, q_3\}$. It is reached by reading the word ba. By further reading only b's and c's, all pairs are reachable where the indices have a distance of two.

Reading the word ba from state $\{q_1, q_3\}$ leads into the state $\{q_1, q_3, q_5\}$, from which all sets of three states of the same structure are reachable. This means, the indices of the states pairwise differ by either two or four.

From this, we can deduce that all sets of size at least two, that do not contain q_n, and which represent states of the power set automaton, consist of states of $\mathcal{A}$ that have consecutive indices that pairwise differ by a multiple of two. This especially means, for such a set $\{q_{i_1}, q_{i_2}, \ldots, q_{i_k}\}$ the indices can be calculated by i_1,$i_2 = i_1 + 2$, $i_3 = i_2 + 2, \ldots, i_k = i_{k-1} + 2$.

By words of the form $((ab)^k c)^l b^m$ for $k, l, m \geq 0$ it is possible for the sets of states with consecutive even or odd indices to build bigger gaps between the indices. But the distance is always a multiple of 2.

All in all, this gives the possibility to reach every subset of $\{q_1, q_2, \ldots, q_{n-1}\}$, where all the indices of the contained states are either even or odd, but not both in the same set. There is only one exception to this. The subsets of at least size two cannot contain state q_1 without state q_3. This is due to the only possibility of reaching q_1 by reading symbol a in state q_2, which also leads into state q_3.

Counting all the sets containing state q_i, where $i \geq 2$ is even and the biggest of all indices within the set, we have at most $\frac{i}{2} - 1$ states that may be contained in addition to q_i in the same set. This gives exactly

$$\sum_{j=1}^{\frac{i}{2}-1} \binom{\frac{i}{2}-1}{k} = 2^{\frac{i}{2}-1} - 1$$

such sets of states.

If q_i is the state with the biggest index in a set consisting only of states with an odd index, we have

$$2 \cdot \sum_{j=1}^{\frac{i}{2}-1} \binom{\frac{i}{2}-1}{k} = 2^{\frac{i}{2}-1} - 1$$

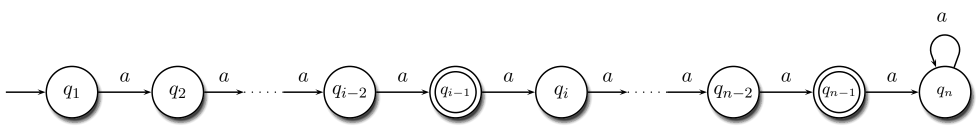


Figure 3.21: DFA used in the proofs of several defects.

many such sets of states. This number excludes the set $\{q_1\}$.

In both of the sums from above the state represented by the empty set is not included. Therefore, for the total number of reachable states in the DFA received by the power set construction, we have the following sum:

$$2 + \sum_{\substack{i \text{ even},\\ i\in\{1,2,\ldots,n-1\}}} 2^{\frac{i}{2}-1} + 2 \cdot \sum_{\substack{i \text{ odd},\\ i\in\{1,2,\ldots,n-1\}}} 2^{\frac{i}{2}-1}$$

For even $n-1$ this is exactly $2^{\frac{n}{2}+1}-1$, and if $n-1$ is odd this is $2^{\frac{n+1}{2}} + 2^{\frac{n-1}{2}} - 1$. All of these reachable sets can be shown to be inequivalent by words consisting only of b's. The exact number is given by $n-1-i$, where i is the biggest index such that state q_i belongs only to one of the two considered state sets. This concludes the proof, since the DFA derived from the power set construction is minimal with a number of states in $\Omega(2^n)$. □

Theorem 3.3.3 shows that even for a minimal DFA accepting a finite language the defect flip transition can lead to a massive blow up in the number of states for the equivalent minimal DFA.

In several of the following theorems, the automaton depicted in Figure 3.21 is used in their proofs. Therefore, let $\mathcal{A} = \langle Q, \{a\}, \delta, q_1, \{q_{i-1}, q_{n-1}\}\rangle$ denote the depicted DFA, where $Q = \{q_1, q_2, \ldots, q_n\}$ denotes its set of states, $1 \le i \le n-1$, and the state transition function $\delta : Q \times \{a\} \to Q$ is defined as follows:

$$\delta(q_j, a) = \begin{cases} q_{j+1} & \text{if } j \in \{1, 2, \ldots, n-1\}, \\ q_n & \text{if } j = n \end{cases}.$$

This DFA is complete and each of the states is reachable for all numbers $n \ge 2$ and $1 \le i \le n-1$. In case that $i = 1$, the only accepting state of $\mathcal{A}$ is state q_{n-1}. The inequivalence of two different states q_j and q_k from $Q \setminus \{q_{i-1}, q_{n-1}\}$ is proved by the word a^{n-j-1}, if $j < k$. The states q_{i-1} and q_{n-1} are proven to be inequivalent by the word a^{n-i-2}. This shows the minimality of the chosen DFA.

The next two theorems will cover the defect delete transition affecting minimal DFA accepting a finite languages. It is shown that this defect may lead to either a massive loss of states or no loss at all. This depends on the structure and the position of the missing transition.

Theorem 3.3.4. *Let $\mathcal{A}$ be a complete and minimal DFA with $n \geq 3$ states that accepts a finite language and let $\mathcal{A}'$ be the incomplete DFA that is received by the defect delete transition affecting $\mathcal{A}$. Then the equivalent minimal DFA of $\mathcal{A}'$ needs at most n states. This bound is tight already for unary alphabets.*

Proof. For any minimal DFA accepting a finite language, the deletion of a transition results in an incomplete finite automaton. This automaton is still deterministic since the deletion of a transition does not insert ambiguity to any of the existing states. In a minimal DFA for finite languages there always exists a rejecting sink state. This would be the only state to be inserted in the worst case to complete the incomplete DFA. Thus, the upper bound is given by n.

To prove the tightness of this bound, let $\mathcal{A}$ denote the minimal DFA depicted in Figure 3.21, and its set of states by $Q = \{q_1, q_2, \ldots, q_n\}$.

In case the defect delete transition affects the transition on a for state q_{n-1}, the resulting defective automaton is an incomplete DFA. The equivalent minimal DFA is automaton $\mathcal{A}$ itself. This proves the optimality of the upper bound for this defect. □

The automaton in Figure 3.21 can also be used to prove that all numbers of states between 1 and n are possible to reach for the minimal DFA that is equivalent to the automaton derived from a minimal DFA accepting a finite language that is affected by the defect delete transition.

Theorem 3.3.5. *Let $n \geq 3$ be a number. Then there exist minimal DFA $\mathcal{A}$ with n states accepting finite languages, such that by the defect delete transition, the equivalent DFA for the defective automaton needs precisely a number of i states, for all $1 \leq i \leq n$. This is true even for unary alphabets.*

Proof. Let $i \in \{1, 2, \ldots, n\}$ be a fixed number and $\mathcal{A} = \langle Q, \{a, b\}, \delta, q_1, \{q_{i-1}, q_{n-1}\}\rangle$ denote the minimal DFA depicted in Figure 3.21, where $Q = \{q_1, q_2, \ldots, q_n\}$ is its set of states. Here, $i = 1$ means, only q_n is accepting.

Let the considered defect delete the transition on a from state q_{i-1} to state q_i. For $2 \leq i \leq n-1$ this results in the loss of the reachability of the states $q_i, q_{i+1}, \ldots, q_n$. The accepted language is $\{a^{i-2}\}$, which is accepted by a minimal DFA with precisely i states.

The case $i = n$ is already shown in Theorem 3.3.4. For $i = 1$, the defect affects the transition on a of state q_1. The resulting defective automaton accepts the empty language, which is known to be accepted by a minimal DFA with only one state. This concludes the proof. □

Just like for arbitrary regular languages, the next defect on transitions to investigate is insert transition adding a transition to the already existing transitions in a minimal DFA accepting a finite language. The next theorem provides an upper bound which is tight in the order of magnitude.

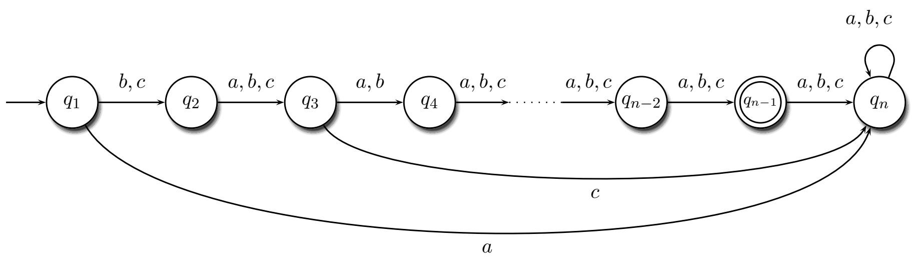


Figure 3.22: DFA for which the defect insert transition inserts a transition from q_1 to q_2 on a results in a minimal DFA with $\Omega(2^n)$ states.

Theorem 3.3.6. *Let $\mathcal{A}$ be a minimal DFA with $n \geq 2$ states that accepts a finite language, and let $\mathcal{A}'$ be the NFA that is received by the defect insert transition affecting $\mathcal{A}$. Then the minimal DFA that is equivalent to $\mathcal{A}'$ needs at most $2^n - 1$ states. This bound is tight in the order of magnitude for at least ternary alphabets.*

Proof. The upper bound of $2^n - 1$ is derived from the arbitrary case proven in Theorem 3.1.7. To prove the optimality in the order of magnitude, the automaton depicted in Figure 3.22 can be modified by the insertion of a transition on a from state q_1 to q_2 to receive an automaton quite similar to the one used in the proof of Theorem 3.3.3. For the two automata the sets of reachable and inequivalent states in the power set automata are the same. Therefore, a lower bound of $\Omega(2^n)$ for the defect of insert transition can be derived from the one for the defect flip transition. This concludes the proof. □

This concludes the investigations for defects on the transitions affecting minimal DFA that accept finite languages. The next theorems will cover the defects on states. The first type of this kind of defect is delete state, either accepting or non-accepting. The following theorem considers the defect delete accepting state.

Theorem 3.3.7. *Let $\mathcal{A}$ be a complete and minimal DFA with $n \geq 2$ states that accepts a finite language, and let $\mathcal{A}'$ be the incomplete DFA that is received by the defect delet accepting state. Then the equivalent minimal DFA of $\mathcal{A}'$ needs at most $n - 1$ states. This bound is tight already for unary alphabet.*

Proof. Since the deletion of states only leads to undefined transitions, the insertion of a rejecting sink state to build the target of the undefined transitions leads to the minimal DFA equivalent to the defective automaton in the worst case. Due to the fact that a rejecting sink is already included in a complete and minimal DFA accepting a finite language, this leads to the upper bound of at most $n - 1$ states.

Let $\mathcal{A}$ denote the DFA depicted in Figure 3.21 with state set $Q = \{q_1, q_2, \ldots, q_n\}$, and $i = n - 1$.

If the considered defect affects the accepting state q_{n-1} the resulting defective automaton only consists of the reachable states $q_1, q_2, \ldots, q_{n-2}$. The non-accepting states q_j and q_k from $\{q_1, q_2, \ldots, q_{n-3}\}$ are still inequivalent, which can be proven by the word a^{n-j-3} in case that $j < k$. Reinsertion of the rejecting sink state q_n and the transition on a from state q_{n-2} to q_n leads to the complete and minimal DFA that is equivalent to the defective automaton since q_n can also be proven to be inequivalent to any state q_j from $\{q_1, q_2, \ldots, q_{n-3}\}$ by the same word a^{n-j-3}.

This DFA has exactly $n - 1$ states, which matches the upper bound. □

To conclude the investigations on the defect delete accepting state, the next theorem covers the question about the possible numbers of states below the upper bound of $n - 1$.

Theorem 3.3.8. *Let $n \geq 3$ be a number. Then there exist minimal DFA $\mathcal{A}$ with n states, each one accepting a finite language, such that by the defect delete accepting state, the equivalent DFA for the defective automaton needs precisely a number of i states, for all $1 \leq i \leq n - 1$. This is true even for unary alphabets.*

Proof. For this we again use the DFA $\mathcal{A}$ that is depicted in Figure 3.21, where $Q = \{q_1, q_2, \ldots, q_n\}$ denotes its set of states, and i is an index between 1 and $n - 1$.

The deletion of state q_{n-1} results in an incomplete deterministic automaton which accepts the language $\{a^{i-2}\}$. If $i = 1$ this is the empty language, and if $i = 2$ this is the language $\{\lambda\}$. For all $1 \leq i \leq n - 1$, this language is accepted by a minimal DFA with precisely i states. □

The next defect of deletion of states affects non-accepting states. There exists a small difference between the deletion of accepting and non-accepting states. One difference is the upper bound for the number of states that a minimal DFA needs to accept the same language as the defective automaton. In the following theorem, this upper bound is shown.

Theorem 3.3.9. *Let $\mathcal{A}$ be a minimal DFA with $n \geq 2$ states that accepts a finite language, and let $\mathcal{A}'$ be the incomplete DFA that is received by the defect delet non-accepting state affecting $\mathcal{A}$. Then the equivalent minimal DFA of $\mathcal{A}'$ needs at most n states. This bound is tight even for unary alphabet.*

Proof. The worst case to happen in case that a non-accepting state is deleted is the deletion of the rejecting sink state. In this case, there exist undefined transitions for some states and a new rejecting sink state needs to be inserted to obtain a minimal DFA. Therefore, the upper bound is given by n states.

To prove the optimality of the upper bound, let $\mathcal{A}$ be the DFA depicted in Figure 3.21, $Q = \{q_1, q_2, \ldots, q_n\}$ denote its set of states, and $i = n - 1$. In case that the defect delet non-accepting state affects the rejecting sink state q_n of $\mathcal{A}$, the equivalent minimal DFA for this defective automaton is precisely $\mathcal{A}$ since the defect did

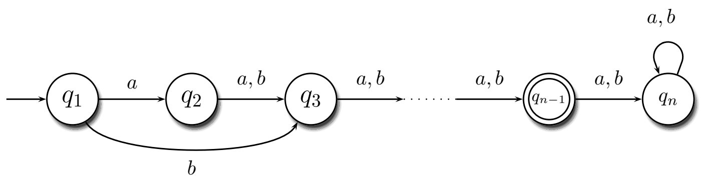


Figure 3.23: DFA for which the defect delete non-accepting state affecting state q_2 results in a minimal DFA with $n-1$ states.

not have any effect on the accepted language. This shows the reachability of the upper bound and concludes the proof. □

The next theorem will give the answer to the question of possible sizes of a DFA accepting the language of a defective automaton that was received by the defect delete non-accepting state.

Theorem 3.3.10. *Let $n \geq 3$ be a number. Then there exist minimal DFA $\mathcal{A}$ with n states, each one accepting a finite language, such that by the defect delete non-accepting state, the equivalent DFA for the defective automaton needs precisely a number of i states, for all $1 \leq i \leq n$. This is true even for unary alphabets.*

Proof. For the number of states $i \in \{1, 2, \ldots, n-2, n\}$, we use the automaton depicted in Figure 3.21 for which we have already shown that this is a minimal DFA. In case that the defect affects state q_i, the states $q_i, q_{i+1}, \ldots, q_n$ are not reachable anymore and the only accepting state is state q_{i-1}. To make this automaton a minimal DFA it is sufficient to add a rejecting sink state that builds the target of all missing transitions. This results in a DFA with exactly i states.

The case $i = n-1$ needs to be proven in a slightly different way because the deletion of state q_i is impossible for this defect since q_{n-1} is accepting. Therefore, let $\mathcal{A} = \langle Q, \{a, b\}, \delta, q_1, \{q_{n-1}\}\rangle$ be the DFA depicted in Figure 3.23, where its set of states is denoted by $Q = \{q_1, q_2, \ldots, q_n\}$, and the state transition function $\delta : Q \times \{a, b\} \to Q$ is defined as follows:

$$\delta(q_j, a) = \begin{cases} q_{j+1} & \text{if } j \in \{1, 2, \ldots, n-1\}, \\ q_n & \text{if } j = n \end{cases},$$

and

$$\delta(q_j, b) = \begin{cases} q_3 & \text{if } j = 1, \\ q_{j+1} & \text{if } j \in \{2, 3, \ldots, n-1\}, \\ q_n & \text{if } j = n \end{cases}.$$

This DFA is minimal since each state $q_j \in \{q_1, q_2, \ldots, q_n\}$ is reachable by the word a^{j-1} and two states $q_j, q_k \in \{q_1, q_2, \ldots q_{n-2}, q_n\}$, where $j < k$, can be shown to be inequivalent by the word a^{n-j-2}.

If the defect delete non-accepting state affects state q_2, the resulting automaton is not complete anymore. The completeness can be received by adding the transition from q_1 on a into state q_n. But it is still minimal since all of the states q_j from the set $\{q_3, q_4, \ldots, q_n\}$ are still reachable from the initial state q_1 by the words ba^{j-3}, and for all states $q_j, q_k \in \{q_3, q_4, \ldots, q_{n-2}, q_n\}$ the word ba^{n-j-3} proves the inequality of q_j and q_k, if $j < k$. Since this DFA consists of exactly $n-1$ states, the proof is concluded. □

The considered defect can only result in a minimal DFA with $n-1$ states if there exists at least one path in the original DFA not containing the deleted state that preserves access to all of the states that are not deleted. This is only possible for at least binary alphabets.

Now that the defects deleting states are covered, we turn come to those defects that change the accepting property of the states. The first defect we are going to investigate is the defect remove acceptance.

Theorem 3.3.11. *Let $\mathcal{A}$ be a complete and minimal DFA with $n \geq 3$ states that accepts a finite language, and let $\mathcal{A}'$ be the incomplete DFA that is received by the defect remove acceptance. Then the equivalent minimal DFA of $\mathcal{A}'$ needs at most n states. This bound is tight even for unary alphabet.*

Proof. The deletion of the acceptance property does not insert any ambiguity to the DFA nor does it delete any transitions. This means, this defect may not change anything on the completeness of the DFA, but may result in a DFA that is not minimal anymore. In the worst case even the minimality is preserved. Thus, the upper bound for the number of states of the minimal DFA that is equivalent to the defective automaton is given by n.

To prove the tightness of this bound, let $\mathcal{A}$ again be the minimal DFA depicted in Figure 3.21, where $Q = \{q_1, q_2, \ldots, q_n\}$ denotes its set of states.

Let i be fixed to the index $n-1$. Then, states $q_{i-1} = q_{n-2}$ and q_{n-1} are different, for $n \geq 3$. In case that the defect removes the accepting property of state q_{i-1}, the resulting DFA accepts the language $\{a^{n-2}\}$. This language is accepted by a minimal DFA with precisely n states already given by the defective automaton. This concludes the proof. □

The proof of Theorem 3.3.11 shows that removing acceptance for a state may result in a DFA that is minimal again. This especially means that it may be difficult to recognise if a given automaton is defective or not. This aspect will be further investigated in the next chapter.

Since the upper bound of n states is the worst case, it is interesting to have a look at other possible numbers of states below this bound. The following theorem proves that the defect remove acceptance may result in another minimal DFA with exactly

$1 \leq i \leq n$ states, where $i = n$ was already shown in the proof of the tightness of the upper bound.

Theorem 3.3.12. *Let $n \geq 3$ be a number. Then there exist minimal DFA $\mathcal{A}$ with n states, each one accepting a finite language, such that by the defect remove acceptance, the equivalent DFA for the defective automaton needs precisely a number of i states, for all $1 \leq i \leq n$. This is true even for unary alphabets.*

Proof. For any number $1 \leq i \leq n$, we use the automaton depicted in Figure 3.21 for which it was already shown that this is a minimal DFA. In case that the defect affects state q_{n-1}, the minimality of this automaton is not preserved, since then the only accepting state is q_{i-1} and all states from $\{q_i, q_{i+1}, \ldots, q_n\}$ are equivalent and can be replaced by a single rejecting sink state. The states from $\{q_1, q_2, \ldots, q_{i-2}\}$ are still reachable and inequivalent. This means that the equivalent minimal DFA for this defective automaton has exactly i states.

This also covers the special case $i = 1$, where only state q_{n-1} is accepting. Removing acceptance for this state results in an automaton recognising the empty language, for which the minimal DFA consists of exactly one rejecting sink state. □

The last defect we want to consider for finite languages also concerns the accepting property of a state. This time, this property is not removed but inserted for some non-accepting state. Theorem 3.3.13 gives a tight upper bound of n for the number of states of the minimal DFA that is equivalent to the defective automaton.

Theorem 3.3.13. *Let $\mathcal{A}$ be a complete and minimal DFA with $n \geq 3$ states, and let $\mathcal{A}'$ be the incomplete DFA that is received by the defect add acceptance. Then the equivalent minimal DFA of $\mathcal{A}'$ needs at most n states. This bound is tight even for unary alphabet.*

Proof. The insertion of the acceptance property to any of the already existing states does not insert any ambiguity. It also does not lead to any undefined transitions. The only thing this defect may change is the property of inequivalence for some of the states. In the worst case, the given DFA can be modified by the considered defect and the minimality property is not affected. This gives an upper bound of n states.

To prove the tightness of this bound, let $\mathcal{A}$ be the minimal DFA depicted in Figure 3.21, where $Q = \{q_1, q_2, \ldots, q_n\}$ denotes its set of states, and $i = 1$, which means that only state q_{n-1} is accepting.

In case that the defect affects state q_1, the minimality of the automaton is preserved for $n \geq 3$. All of the states are still reachable by reading a certain amount of a's, and the non-accepting states can still be distinguished also by a word containing only a's. The only two states that need to be checked to be inequivalent are the two accepting states q_1 and q_{n-1}. But these two states are not equivalent since the word a^{n-2} read from q_1 leads into state q_{n-1}, wherefore q_{n-1} ends up in the rejecting sink state q_n. Since this minimal DFA still consists of n states, this proves the optimality of the bound. □

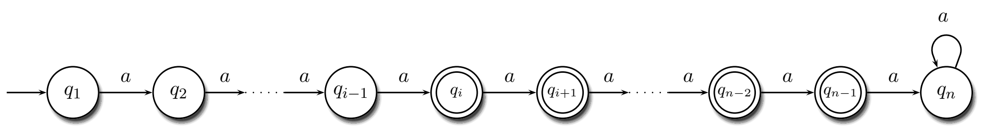


Figure 3.24: DFA for which the defect of insertion of the accepting property for state q_n results in a minimal DFA with i states, where $1 \leq i \leq n$.

Also for this special defect we are interested of the possible numbers of states for a minimal DFA that is equivalent to the defective automaton. The answer to this question is given in the following theorem.

Theorem 3.3.14. *Let $n \geq 3$ be a number. Then there exist minimal DFA $\mathcal{A}$ with n states, each of them accepting a finite language, such that by the defect add acceptance, the equivalent DFA for the defective automaton needs precisely a number of i states, for all $1 \leq i \leq n$. This is true even for unary alphabets.*

Proof. The number of states $i = n$ is already shown in Theorem 3.3.13 for $n \geq 3$.

For $1 \leq i \leq n-1$, let $\mathcal{A} = \langle Q, \{a\}, \delta, q_1, \{q_i, q_{i+1}, \ldots, q_{n-1}\}\rangle$ denote the DFA depicted in Figure 3.24, where its set of states is given by $Q = \{q_1, q_2, \ldots, q_n\}$, and its state transition function $\delta : Q \times \{a\} \to Q$ is defined as follows:

$$\delta(q_j, a) = \begin{cases} q_{j+1} & \text{if } j \in \{1, 2, \ldots, n-1\}, \\ q_n & \text{if } j = n \end{cases}.$$

This DFA is minimal for $n \geq 3$, since the longest accepted word is of length $n - 2$.

In case that the defect adds acceptance for state q_n, all of the accepting states are equivalent. The longest word not accepted then is a^{i-2} for $i \geq 2$. It is well known, that the minimal DFA accepting such a language needs exactly i states. For $i = 1$, the language $\{a\}^*$ is accepted by the defective automaton. A minimal DFA accepting this language consists of only one state. □

Concluding this chapter, the results of the theorems are summarised in Table 3.3.

Delete Transition	n
Insert Transition	$\Omega(2^n) \leq \cdot \leq 2^n - 1$
Flip Transition	$\Omega(2^n) \leq \cdot \leq 2^n$
Exchange Symbol	$\Omega(n^2) \leq \cdot \leq \frac{n(n-1)}{2} + 1$
Delete Non-Acc. State	n
Delete Accepting State	$n - 1$
Remove Acceptance	n
Add Acceptance	n

Table 3.3: Bounds for minimal DFA that are equivalent to a defective automaton received from a minimal DFA that accepts a finite language. The bounds for the defects flip or insert transition, and for the defect exchange symbol are tight in the order of magnitude for at least ternary alphabets. All of the other bounds are tight even for unary alphabets.

4 Recognition and Correction of Defects in DFA – By Example

In the previous chapter, we have seen that the considered defects may lead to large minimal DFA accepting the language of a defective automaton compared to the size of the original DFA. For some defects the size can even be exponential. In practice, the automata used in speech processing often already consist of more than one hundred million states [52]. If such an automaton is affected by a defect, this may be desastrous for its size, and also for the usability.

This leads to the question of the possibility of recognition and also of fixing such a defect. This may help to recognise a defective automaton. In addition, this leads to the knowledge that the given automaton is not the original DFA. If the defect could be recognised, located, and even be fixed, this would be helpful.

In the following, we will give properties of defective automata, that characterise the appearance of the different defects. We already mentioned that more than one appearance of one type of defect, or mixing the appearances of different defects may lead to a different type of defect. This is the reason for restricting the following research to only one appearance of only one type of defect. We will show, that some defects lead to similar properties, and some to different ones. This means, not all defects can be determined precisely by these properties.

We will also prove by example, that only one of the considered defects is fixable, even if it is known, which defect leads to the defective automaton. Concluding from this, it is nearly impossible to fix a defect.

This is the reason, why we turn to the investigation of some subsets of the language that is accepted and rejected by the defective automaton, and the sizes of the minimal DFA accepting these subsets. One of these subsets contains those words that are accepted by the given automaton and which do not use the defect while being processed by the defective automaton. These words were already accepted by the original DFA. Another subset we will consider consists of all the rejected words, that do not use the defect while being processed by the given automaton. This set of words was also rejected by the original DFA. The third subset gathers all the words that are accepted or rejected by the defective automaton that explicitly use the defect while being processed. For these words it is not known, if they were also accepted or rejected by the original DFA. Constructions will be given for automata accepting these languages and upper bounds for the number of states for equivalent minimal DFA.

4.1 Recognition of Defects in a Finite Automaton

At the first sight, it seems to be quite difficult to recognise, if an arbitrary finite automaton is defective or not. Some of the considered defects modify a minimal DFA to an NFA, but for some defect, the automaton still is a minimal DFA. This especially is true for the combination of more than one occurence of one type of defect, and also for a combination of several defects. This is why the following investigations are restricted to defective automata that are the result of only one occurence of a single type of defect.

At first we take a look at the properties of a defective automaton, that is the result of a single defect. For the defect exchange symbol there exists one state having exactly one undefined and one ambiguous transition. The defect flip transition leads to one undefined transition for one state and an ambiguity for another state. These two states are connected at least by the flipped transition. The insertion of one transition leads to an ambiguity for one state. All of these defects result in an NFA.

The defect delete transition results in an automaton with one undefined transition for exactly one state. This automaton is always an incomplete DFA.

The deletion of one accepting state and all transitions having this state as source or target leads to at least one undefined transition for at least one state. The same holds true for the deletion of a non-accepting state and all transitions starting or ending in this state. The received automaton is still deterministic but incomplete.

Both defects that affect the property of acceptance for a state do not add any ambiguity or introduce any undefined transitions to the DFA. It may only happen that the minimality of the DFA gets lost.

Table 4.1 gives an overview of the possible results of the considered defects.

With the help of this knowledge it is now possible to recognise that a defect occurs in a given automaton, assuming that only one defect occurs. In some cases we are even able to decide which of the possible defects is at hand. We need to read the entries of Table 4.1 not from the left to the right but from the right to the left. This means, if the given automaton has only one transition missing for one state, and the same state also has an ambiguity for exactly one symbol, then we know that the defect exchange symbol lead to the given defective automaton. In case that the given automaton misses only one transition for one state and has an ambiguity for one symbol for another state, this automaton is the result of the defect flip transition. If the automaton has no missing transitions but one ambiguity for one state, the defect insert transition has appeared. Having that only one transition is missing for one state, there exists more than one possible defect that may be present. The first is the obvious one delete transition. The other one is delet state, either accepting or non-accepting. This means, that for only one undefined transition it is impossible to decide, which of the three possible defects has occurred.

The last two possible defects that affect the acceptance of states may also not be recognised since the given automaton may still be complete and minimal. In case the automaton is complete but not minimal anymore, we can conclude that a defect of this kind has appeared, since all of the other defects lead to undefined transitions

Defect	Property of the defective automaton
Delete Transition	One undefined transition for precisely one state
Insert Transition	One ambiguous transition for precisely one state
Flip Transition	One ambiguous transition for one state, one undefined transition for one other state
Exchange Symbol	One ambiguous and one undefined transition for one state
Delete Non-Accepting State	At least one undefined transition for at least one state
Delete Accepting State	At least one undefined transition for at least one state
Remove Acceptance	none
Add Acceptance	none

Table 4.1: Characteristic properties of a defective automaton resulting from a minimal DFA for precisely one defect of one of the considered types.

or ambiguities for some states. But, in general, we cannot differentiate between the two defects.

4.2 Correction of Defects in a Finite Automaton

In the previous section, criteria were introduced to decide which of the defects is responsible for the defective automaton. In this section, we give possibilities to correct such a defective automaton.

Given an automaton and the knowledge which of the defects has occurred exactly once to receive the given defective automaton from a complete and minimal DFA, we can try to fix the automaton. This is not that simple for nearly all of the defects since there may exist several possibilities that fix the defect, and lead to a minimal DFA. In this case, it is impossible to decide which of the possible minimal DFA is the original one. This would only be possible, if not only the defective automaton is given but also some more information about the original DFA, like for example its set of accepting states.

In the following, the different defects will be investigated on the subject of how to fix the defect in the given automaton with the knowledge, which defect lead to this automaton and that this defect only appears once within the automaton.

Exchange Symbol: The first defect to discuss is the one that exchanges the symbol of a transition. In the previous section it was shown that this leads to one undefined transition for one state and an ambiguity for the same state. More precise, this state leads into two other states on the same symbol. It is obvious that one of these transitions is the defective one. To repair the automaton it is possible to construct two DFA by exchanging the symbol for one of the ambiguous ones by the symbol of the undefined transition. One of these two DFA is the one that was affected by the defect and resulted in the given automaton. This DFA then is minimal and complete. But also the other automaton may be complete and minimal. So we may not be able to decide which of the two constructed DFA is the original one.

The two following examples will show that it is both possible and impossible for some cases to fix the defect in a given automaton if it is known that the symbol of exactly one transition was exchanged.

Example 4.2.1. Let the automaton from Figure 3.2 be the given defective automaton. The exchange of the symbol needs to have affected the transition on b with source state q_n. In order to fix this defect, we need to change the symbol from a into b for one of the two transitions on a starting in state q_n. This leads either to the minimal DFA depicted in Figure 3.1 or the one depicted in Figure 4.1, which is also minimal. Thus, it is impossible to decide, which of these DFA is the one the given defective automaton was derived from. ⌜⌝

Example 4.2.2. For the automaton depicted in Figure 4.2, we have two possibilities to correct the defect. Either the transition from q_{n-1} to itself on b or the one

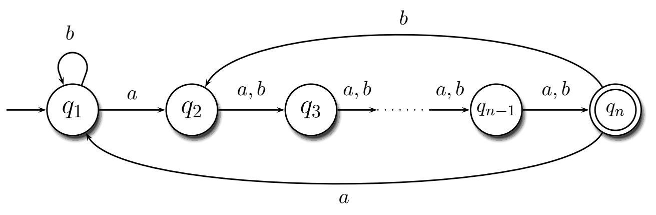


Figure 4.1: Minimal DFA used in Example 4.2.1 for the defect exchange symbol.

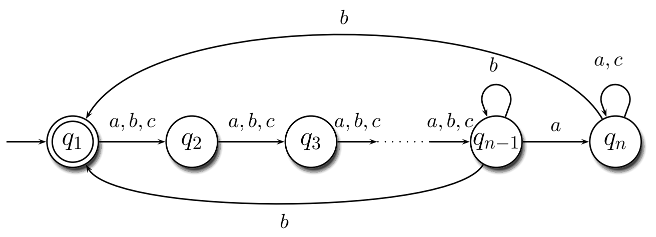


Figure 4.2: Defective automaton used in Example 4.2.2 for the defect exchange symbol.

from q_{n-1} to q_1 on b is replaced by one on c. The second possibility leads to a minimal DFA since all states are reachable by reading a certain number of a's from the initial state and can be distinguished by reading a word from $\{a\}^+\{b\}$.

The other fixed DFA is the one depicted in Figure 4.3. Here states q_{n-1} and q_n are equivalent and, thus, this DFA is not minimal. Therefore it is possible to decide from which the defective automaton was derived. ⌜⌟

Flip Transition: The defect that interchanges the source and the target of a transition causes an ambiguity for one state and an undefined transition for another state,

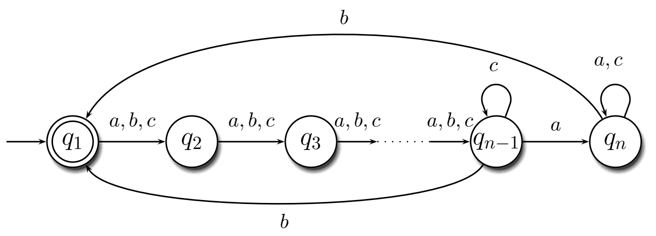


Figure 4.3: Not minimal DFA used in Example 4.2.2 for the defect exchange symbol.

in case that the flipped transition does not have the same source and target. In fact, these states are connected by the ambiguous transition before and after the defect takes effect. This means, when trying to fix the defect, the target of the defective transition is exactly the state with the undefined transition on the symbol of the ambiguous transition. Thus, it is decidable which transition is the defective one and can be fixed. Given a complete and minimal DFA with state set $\{q_1, q_2, \ldots, q_n\}$, and an alphabet containing symbol a. In case that the transition on a from a state q_i to a state q_j is corrupted and there already exists a transition on a from q_j to q_i the defective transition is not merged with the original transition on a even if the source, target, and symbol are the same. This is necessary to recognise the type of defect and also to recognise the defective transition.

Example 4.2.3. Let the automaton from Figure 3.4 be the given defective automaton. It is obvious that there exists an ambiguity for the transition on b for state q_n and an undefined transition on b for state q_{n-1}. Thus, the transition on b connecting the states q_n and q_{n-1} is the defective transition and needs to be fixed. This leads to the DFA depicted in Figure 3.3, which was proven to be complete and minimal in Theorem 3.1.2. This DFA is the one, the given defective automaton was derived from. ⌜⌟

Delete Transition: For the defect delete transition, the state for which this transition is missing can be identified. It is also possible to identify the symbol for this transition. The difficulty in fixing this defect is to determine the target of the missing transition. There exist n different targets for this transition if the given automaton consists of n states. It is possible that not only for one of these targets the resulting DFA is minimal. In general there exist more than only one resulting DFA that are minimal and it is impossible to decide which one is the original DFA the given defective automaton is derived from.

In the following we will give two examples, one for the undecidability for the original DFA, and one for which the only minimal DFA resulting from the fixing of the defect is the original DFA.

Example 4.2.4. If for the minimal DFA depicted in Figure 3.16 its transition from q_n to q_1 on a is deleted, it is simple to identify that a transition on a is missing for state q_n. There exist n possible targets to complete this automaton. One of the resulting DFA is the original one from Figure 3.16, if the missing transition is inserted with target q_1. In the proof of Theorem 3.2.1 it was shown that this DFA is complete and minimal. Another possible fixed DFA is the one with target q_{n-1}. This is a DFA that is not minimal since the two accepting states q_{n-1} and q_n are equivalent.

A third option for the target of the missing transition would be state q_2. The resulting DFA with this transition also is minimal since all states q_i, $1 \leq i \leq n$, are still reachable by reading the word a^{i-1} from the initial state q_1. Two states q_i, q_j, $1 \leq i < j \leq n-2$, can be shown to be inequivalent by the word a^{n-j}. States q_{n-1} and q_n can be differentiated by the word a.

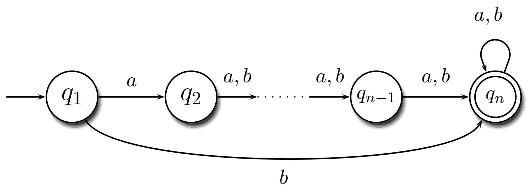


Figure 4.4: Minimal DFA used in Example 4.2.5 for the defect delete transition.

Thus, there exist more that one minimal DFA resulting from the correction and repair of the defect, and it is impossible to decide from which one the given automaton is derived. ⌜⌟

Example 4.2.5. Let the DFA depicted in Figure 4.4 be defected by deleting the transition on a for state q_1. Then only state q_n is still reachable, all of the other states cannot be reached anymore from the initial state q_1 by any word. The only possibility to fix this automaton and receive a minimal DFA where all of the states are reachable again is to insert exactly the transition on a for state q_1 leading into state q_2. All of the other possible targets lead to DFA that are not minimal. Thus, in this case, there only exists one minimal DFA with n states that results from fixing the defect, and is, therefore, decidable. ⌜⌟

Insert Transition: Like discussed in Section 4.1, the defect insert transition leads to an ambiguity for exactly one state. This especially means that for this state there exist transitions on all symbols and for one symbol there exist two targets. In case that these two targets are the same state, the correction is done by merging the two transitions on the same symbol. This leads to the DFA the defective automaton is derived from.

In general, the two targets for the ambiguous transition are different. The deletion of either one of the two transitions on the same symbol leads to two different DFA that may both be minimal. Therefore it is impossible to decide which of the two minimal DFA is the one, the given defective automaton is derived from.

Example 4.2.6. Let the given automaton be the one depicted in Figure 3.12. The inserted transition is either the one on a from state q_1 to q_2 or the one to q_1 itself. Trying to fix this automaton needs the deletion of one of these transitions. The deletion of the transition on a from state q_1 to q_2 results in the DFA depicted in Figure 3.13 for which its minimality was already shown in the proof of Theorem 3.1.7.

The other possible resulting DFA is the one depicted in Figure 4.5, which is received from the defective automaton by deleting the transition on a from state q_1 to itself. This automaton is also minimal since the state $q_i \in \{q_1, q_2, \ldots, q_n\}$ is reachable by reading the word a^{i-1} from the initial state q_1, and two different states $q_i, q_j \in \{q_1, q_2, \ldots, q_{n-1}\}$ can be distinguished for $i < j$ by the word a^{n-j}. Therefore,

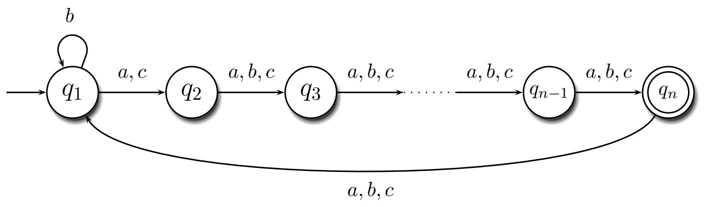


Figure 4.5: Minimal DFA used in Example 4.2.6 for the defect insert transition.

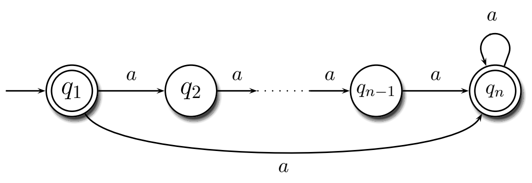


Figure 4.6: Minimal DFA used in Example 4.2.7 for the defect insert transition.

it is impossible to decide, from which of these two minimal DFA the given automaton was derived from without the knowledge of more information of the original DFA. ⌜⌟

Example 4.2.7. Let the DFA depicted in Figure 3.8 be the minimal DFA where $i = 2$. This means, that the two accepting states of this automaton are states q_1 and q_n. Its minimality was already shown in the proof of Theorem 3.1.6. This DFA is corrupted by inserting a transition on the symbol a for state q_1 that leads into state q_n. This automaton is depicted in Figure 4.6.

One of the transitions on a with source q_1 needs to be deleted to correct this defective automaton. The deletion of the transition with target q_n leads exactly to the one, this defective automaton was derived from. But the deletion of the other transition leads to a DFA that is not minimal since the states $q_2, q_3, \ldots, q_{n-1}$ are not reachable anymore.

For this given defective automaton it is obvious which of the corrected automata is the original DFA. ⌜⌟

Delete (Non-)Accepting State: Like for the defect delete transition, also for the defects delete accepting or non-accepting state there exist n different targets for the transitions of the missing state, that needs to be inserted to fix the defect. In case that the given defective automaton consists of $n - 1$ states and an alphabet of size $k \geq 1$, the correction may lead to n^k different DFA. Since this is quite a huge number of automata, not only one of them needs to be minimal. This means that in the

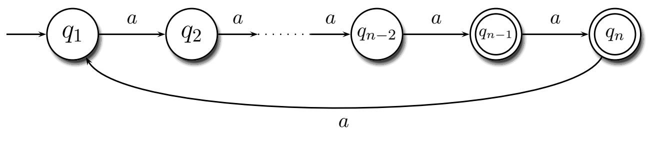


Figure 4.7: Minimal DFA used in Example 4.2.8 for the defect delete accepting state.

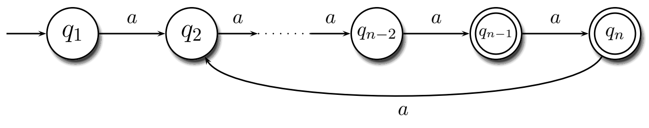


Figure 4.8: Minimal DFA used in Example 4.2.8 for the defect delete accepting state.

general case it is not decidable which of the repaired DFA is the one the defective automaton was received from. The next two examples will give evidence of this fact for both defects.

Example 4.2.8. For the defect delete accepting state, let the given automaton be the one depicted in Figure 3.11. If the defect is known from which this automaton is derived, the target of the missing transition on a for state q_{n-1} needs to be the deleted state. Inserting this state leaves the question where the transition on a for this accepting state ends up. Let q_n denote the inserted accepting state. One of the resulting n DFA are the one depicted in Figure 3.8 for $i = n$. This one is not minimal since the two accepting states q_{n-1} and q_n cannot be distinguished. But the two DFA depicted in Figures 4.7 and 4.8 are also possible to receive for the fixed automaton. These DFA are both minimal since for both automata state $q_i \in \{q_1, q_2, \ldots, q_n\}$ is reachable from the initial state q_1 by reading a^{i-1}, and two states $q_i, q_j \in \{q_1, q_2, \ldots, q_{n-2}\}$ for $i < j$ are proven to be inequivalent by the word a^{n-j}. The two accepting states q_{n-1} and q_n can be proven to be inequivalent by the word a in case that $n \geq 4$. It is impossible to decide from which of these two minimal DFA the given defective automaton was derived from or if it even was one of the other $n - 3$ automata, we did not consider explicitly in this example. ⛶

Example 4.2.9. The DFA depicted in Figure 4.9 is minimal since every state $q_i \in \{q_1, q_2, \ldots, q_n\}$ is reachable by reading the word a^{i-1} from the initial state q_1, and two states q_i, q_j, where $1 \leq i < j \leq n - 1$, can be proven to be inequivalent by the word a^{n-j}.

If in this DFA the non-accepting state q_2 is deleted, the resulting defective automaton contains undefined transitions for states q_1 on a and q_n on a and b. To correct the defect, it is necessary to insert a non-accepting state. For the sake of

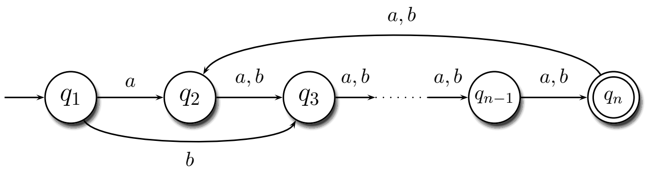


Figure 4.9: Minimal DFA used in Example 4.2.9 for the defect delete non-accepting state.

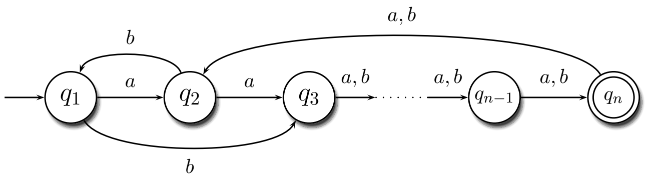


Figure 4.10: Minimal DFA used in Example 4.2.9 for the defect delete non-accepting state.

convenience let this state be called q_2 again. This new state is the target for the three missing transitions for states q_1 and q_n. The targets for the transitions with source state q_2 are still unknown. Since there exist two symbols in the alphabet and n states in the corrected automaton there exist n^2 different combinations for the targets.

One of these possible corrected DFA is the one depicted in Figure 4.9, which is the minimal DFA the defective automaton was derived from. Another minimal DFA is the one depicted in Figure 4.10. A state $q_i \in \{q_1, \ldots, q_n\}$ of this DFA is reachable by reading the word a^{i-1} from the initial state q_1, and two different states q_i and q_j with $1 \leq i < j \leq n-1$ can be shown to be inequivalent by the word a^{n-j}.

This means, there exist at least two minimal DFA that are the results from correcting the defect in the given defective automaton, and it is not possible to decide which of the two is the one, the given automaton was derived from. At least not without further information of the original DFA. ⌜⌝

The following two examples give proof of the possibility to correct the defects of deleting an accepting or a non-accepting state.

Example 4.2.10. Starting from the DFA depicted in Figure 3.8, which was shown to be minimal in the proof of Theorem 3.1.6, where $i = n-1$ and $n \geq 3$, the defect of deleting an accepting state affects state q_{i-1}. In this case, states $q_i, q_{i+1}, \ldots, q_n$ are not reachable anymore.

If it is known that the defect deleted an accepting state, the correction for this given automaton is quite simple. The missing accepting state, lets name it q_{i-1} again, is inserted, and the missing transition for state q_{i-2} is inserted with target q_{i-1}.

The transition on a for state q_{i-1} is still missing to complete the DFA. At the moment, there exists no connection between the states $q_1, q_2, \ldots, q_{i-1}$ and the states q_i up to q_n. This lets conclude, that this last missing transition needs to have the target q_i. The resulting DFA is the original one. Any other possibility of repair leads to a DFA that is not minimal. ⌞⌟

Example 4.2.11. Also for the defect delete non-accepting state, we start from the DFA depicted in Figure 3.8, which was shown to be minimal in the proof of Theorem 3.1.6. For this example, we choose $i = 2$ and $n \geq 3$. In case that the defect of deleting a non-accepting state affects state q_2, states $q_3, q_4, \ldots, q_n$ are not reachable anymore.

If it is known that the defect delete non-accepting state occurred, the correction for the given automaton is the following. The missing non-accepting state, for convenience called again q_2, is inserted. The missing transition for state q_1 with target q_2 is reinserted.

The last transition to insert, is the one on a for state q_2 itself. But since there exists no connection from the part on the automaton consisting of the states q_1 and q_2 and the part containing the states q_3 up to q_n, this transition needs to have target q_3. The resulting DFA is the original DFA. Any other possibility for the target of the transition on a for state q_2 leads to a non-minimal DFA respectively minimal DFA with less than n states. ⌞⌟

Remove Acceptance: The defect remove acceptance leads to a DFA that may not be minimal anymore. In general there exists more than only one possibility to fix this defect. In case that the given defective automaton has state set Q and its set of accepting states is denoted by F, there exist $|Q| - |F|$ many DFA that result from correction. For each of these resulting DFA, for one of the existing non-accepting states of the defective automaton the accepting property is inserted. Just like for the other defects, usually more than only one of these fixed DFA is minimal. The next two examples will show that it is impossible for some automata to determine the DFA from which the defective automaton is derived from, and for some automata only one of the corrected DFA is minimal.

Example 4.2.12. Let the minimal DFA depicted in Figure 3.8 be the one that is affected by the defect remove acceptance on state q_{i-1}, where $2 \leq i \leq n-1$. The minimality of this DFA was already shown in Theorem 3.1.6.

The resulting defective automaton can be fixed by inserting the accepting property for any of the states $q_1, q_2, \ldots, q_{n-1}$. Since we already know that all the DFA resulting from inserting the accepting property to any of the states q_1 up to q_{n-2} are minimal, it is impossible to decide, which of these was the original DFA. ⌞⌟

Example 4.2.13. Let the DFA that will be corrupted be the one depicted in Figure 4.11. This DFA is minimal since a state $q_i \in \{q_1, q_2, \ldots, q_{n-1}\}$ is reachable

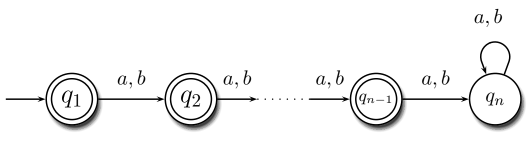


Figure 4.11: Minimal DFA used in Example 4.2.13 for the defect remove acceptance.

from the initial state q_1 by reading the word a^{i-1}, and two states q_i and q_j, where $1 \leq i < j \leq n-1$, can be shown to be inequivalent by the word a^{n-j}.

In case that the defect remove acceptance affects any of the states from the set $\{q_1, q_2, \ldots, q_{n-2}\}$, there exists only one possibility to correct this defect for which the resulting DFA is minimal. ⌜⌟

Add Acceptance: The last defect to consider is the one that adds the accepting property to one of the states of a minimal DFA. This defect can only be corrected by removing the acceptance for one of the accepting states in the defective automaton, that was derived from the minimal DFA. This leads to $|F|$ many different DFA, if F denotes the set of accepting states of the defective automaton. In general, at least two of these resulting DFA are minimal, and it is impossible without more knowledge about the original DFA to decide which of these corrected DFA is the one, the defective automaton was derived from.

In the following, we give an example, where the original DFA cannot be found, and another example where there exists only one corrected DFA that is minimal.

Example 4.2.14. Let the minimal DFA depicted in Figure 3.24 be the one that is defected by adding the accepting property for state q_n, where $1 \leq i \leq n-1$. The minimality of this DFA was already shown in Theorem 3.3.14.

The derived defective automaton can be fixed by removing the accepting property for any of the states $q_i, q_{i+1}, \ldots, q_n$. The two DFA resulting from the removement of the acceptance for either state q_{n-1} or q_n are both minimal, which can easily be seen. Therefore, it is impossible to decide from which of these two DFA the given defective automaton was derived from. ⌜⌟

Example 4.2.15. Starting from the DFA depicted in Figure 3.21, we already known that this DFA is minimal from the proof of Theorem 3.3.7 for all $1 \leq i \leq n-1$. In this example, we choose $i = 1$. Then state q_{n-1} is the only accepting state.

In case that the defect of inserting the accepting property affects state q_{n-2}, there exists only one possibility to correct this defect, and this correction results in the original minimal DFA. ⌜⌟

All the results discussed above are summarised in the following theorem.

Theorem 4.2.1. *Given an automaton and the knowledge, that a defect has appeared in a complete and minimal DFA to receive the given automaton. Then in general for the defects*

- *exchange symbol,*
- *delete transition,*
- *insert transition,*
- *delete accepting state,*
- *delete non-accepting state,*
- *remove acceptance,*
- *add acceptance*

it is impossible to decide, which of the DFA produced by fixing the defect in the given automaton is the one the defect affected originally.

For the defect flip transition, there only exists one possibility for the fixed DFA, which is the DFA that was defected.

This covers the possibility of correcting the defects for a given automaton if it is known that only one defect of one given type is responsible for the automaton.

4.3 Languages Related to Defective Automata

In the last part of this chapter, we will investigate three types of languages and their accepting DFA that can be constructed based on the defective automaton. The first language will be the set that collects all the words that are accepted by the defective automaton, and that do not use the defect while being processed. These are words that are also accepted by the original DFA. This set is called L_+.

The second language is the set of words, which are rejected by the given automaton but do not use the defect while being processed. This set is a subset of the rejected language of the minimal DFA the given automaton is derived from. We will refer to this language by L_-.

The third and last language collects all accepted or rejected words processed by the defective automaton that explicitly use the defect. This language is referred to by $L_?$. For the words of this subset it is unknown if they were accepted or rejected by the original DFA. This is the reason why the investigation of this set of words is of interest.

To be able to consider all of these languages, we need to know which defect occurs in the given DFA. We also need to know which state is affected by this defect. Therefore, in the following, we assume that this information is provided.

4.3.1 Construction of DFA for L_+, L_-, and $L_?$

DFA for L_+: In general, the constructions are quite similar for all of the defects. An automaton accepting L_+ is constructed based on the given defective automaton by removing the existing defect. This especially means to delete all ambiguous transitions since it is unknown which of them is the original one and which one is the defective one, if there exists an ambiguity. In case that the property of acceptance for a state is defective, we assume that the defective state is known. To construct a DFA that accepts L_+ this state is removed. All of these deletions lead to incomplete DFA. To receive a complete DFA in general it is necessary to insert a rejecting sink state. For the defects that delete a state and all the transitions having this deleted state as source or target, the resulting defective automaton is already an incomplete DFA, that accepts L_+. For these defects, it is sufficient to insert a rejecting sink state that builds the target for all missing transitions.

All of these incomplete DFA only consist of states and transitions that already existed in the original DFA the given defective automaton is derived from. This especially includes the accepting states of the original DFA. There may be missing some of the accepting states of the original DFA but there are no additional accepting states in the new automata. Hence, the words accepted by the constructed complete DFA are also accepted by the original DFA.

DFA for L_-: A minimal DFA for the language L_- can be constructed in a quite similar way as the DFA for L_+. The only difference is that before inserting the rejecting sink state to make the DFA complete, the property of acceptance is interchanged for all states. We already observed that the existing states and transitions for the new incomplete DFA where the accepting property was not interchanged already existed in the original DFA. Therefore, by interchanging the accepting property for those states, the now accepted words belong to the rejected language of the original DFA. This is because the DFA accepting the rejected language of the original DFA can be constructed by interchanging the accepting property of the states of the original DFA. This shows that the constructed incomplete DFA accepts only words that belong to the rejected language of the original DFA. This incomplete DFA can be completed by inserting a rejecting sink state as target for the remaining undefined transitions.

DFA for $L_?$: The last DFA that needs to be constructed is the one for the language $L_?$. This automaton is received by modifying the defective automaton. In general, the idea of the construction is the following for defects that affect the transitions or delete a state. All states and transitions of the defective automaton also belong to the constructed DFA, except for the defective ones. In this automaton, all states are non-accepting. A new accepting sink state is inserted. All the missing transitions are now inserted, having this new sink state as target.

The construction is slightly different, if the defect affected the property of acceptance for one state. We assume in this case, that the corrupted state is known. This state is deleted, the acceptance of all the remaining states is removed, and the rest of the construction is similar the one for the other types of defects. By these two constructions, all accepted words use the defect at least once while being processed.

4.3.2 Analysis of the Accepting DFA for the Three Languages

Since the DFA constructed for the languages L_+ and L_- are quite the same before inserting the rejecting sink state except for the property of acceptance of the states, their state complexities are the same for all defects. When interchanging the acceptance property for all states, it does not change anything on the minimality of the DFA. The only possibility is that by changing the accepting property may already introduce a rejecting sink state, such that its insertion is not needed anymore. But in general, this is not the case. The state complexities for the minimal DFA accepting $L_?$ is likely to differ from the state complexities for the languages L_+ and L_-.

Exchange Symbol: Lets start with the defect that exchanges the symbol for a transition. The language L_+ of a defective automaton with n states can be accepted by an automaton based on the given automaton in which the two ambiguous transitions are deleted. In this way, the resulting automaton is deterministic again but not complete. To make it complete it is sufficient to add a rejecting sink state, that builds the target for the two missing transitions, and also for the transitions beginning in this new state. In general, this DFA may be minimal.

Therefore an upper bound for the minimal DFA accepting L_+ for the defect that exchanges the symbol of a transition is given by $n + 1$.

For the language L_- the construction of an accepting minimal DFA was already explained above. In general, the upper bound for such a DFA affected by this type of defect is also a number of $n + 1$ states.

The construction of a minimal DFA accepting the language $L_?$ for the defect exchange symbol is as follows. In the defective automaton, there exists one state having an ambiguous transition for only one symbol. The same state is missing a transition on another symbol. The ambiguous transitions are deleted, so that the automaton now contains a state with undefined transitions for two different symbols. This automaton indeed is an incomplete DFA. To modify this DFA to accept the language $L_?$ of the defective automaton, the acceptance is removed for all existing states. After that, a new accepting sink state is inserted. The yet missing transitions are inserted having this new sink state as target. This DFA is complete and may even be minimal. This gives an upper bound of $n + 1$, just like for the languages L_+ and L_-.

Insert Transition: For the defect insert transition, the deletion of the ambiguous transition is the first step in the construction of a DFA for all languages L_+, L_-, and $L_?$. This automaton does not contain any defective transition. All the transitions and states already existed in the original DFA, the defective automaton was derived from by the defect. This means, the constructed incomplete DFA already accepts the language L_+. To receive a complete DFA for L_+, a new rejecting sink state is inserted. Also the missing transitions are added to the automaton, having this new sink state as target.

This DFA has $n+1$ states. It also may be minimal, wherefore this number is an upper bound for a minimal DFA accepting L_+ for this defect.

For the language L_-, the incomplete DFA is modified by interchanging the accepting and non-accepting states. This automaton already accepts L_- of the defective automaton, but it is still incomplete. As target for the two missing transitions, a new rejecting sink state is inserted, similar to the construction for L_+.

This DFA also consists of $n+1$ states, and may be minimal. This gives the upper bound of $n+1$ also for the language L_- for the insertion of a transition.

The language $L_?$ is not yet accepted by the current incomplete DFA. This language will be accepted by the following automaton. For all of the states of the constructed incomplete DFA, the acceptance is removed. A new accepting sink state is inserted, and the missing transitions are inserted having this new state as target. This complete DFA accepts all words, that are accepted or rejected by the defective automaton, and use the ambiguous transition while being processed. This is precisely language $L_?$.

Since this constructed DFA may be minimal, an upper bound for $L_?$ is given by $n+1$ states for the currently considered type of defect.

Flip Transition: The next defect to consider is the one that interchanges the source and the target of a transition. The minimal DFA that accepts the language L_+ and also the minimal DFA accepting L_- are received in a similar way like for the defect of exchanging the symbol of a transition.

Therefore an upper bound for the minimal DFA accepting L_+ for the defect flip transition is given by $n+1$. The upper bound for the minimal DFA that accepts the language L_- is also $n+1$.

The construction of the minimal DFA accepting the language $L_?$ for this defect also delivers an upper bound of $n+1$. The construction for this language is also similar to the construction for $L_?$ for the defect exchange symbol.

Delete Transition: The language L_+ for the defect delete transition is the one accepted by the defective automaton. The accepted language is already the subset L_+ of the language accepted by the minimal DFA the given automaton is derived from. Since the defective automaton is incomplete but deterministic, it suffices to insert a new rejecting sink state, that builds the target for the missing transition. This DFA may already be minimal, and, therefore, the upper bound for the number of states for a minimal DFA accepting L_+ is $n+1$.

A minimal DFA accepting the language L_- has at most $n+1$ states. The language L_- is accepted by the incomplete DFA that is the result of inverting the acceptance for the defective automaton. Similar to the construction for L_+, a complete DFA accepting L_- is received by inserting a new rejecting sink state as the target for the missing transitions.

The language $L_?$ is defined to be the language that consists of those words that are accepted or rejected by the defective automaton that, in addition, use the defect while being processed. For the defect that deletes one transition, $L_?$ consists of all words that use this deleted transition while being processed by the original DFA. This means, when removing the acceptance for all states of the defective automaton, and adding a new accepting sink state as the new target of the missing transition, we receive a complete DFA that accepts $L_?$. Since this DFA may also be minimal, this gives an upper bound of $n + 1$ states.

Delete Accepting State: The language L_+ of a defective automaton that is received by the defect delete accepting state affecting a minimal DFA is accepted by a DFA that is constructed by inserting a rejecting sink state that is the new target for all missing transitions. This DFA may already be minimal, otherwise the equivalent minimal one has less states. This leads to an upper bound of $n + 1$ states, if the defective automaton has n states.

Just like before, the minimal DFA accepting L_- also has at most as many states as the one accepting L_+. This means, also for the language L_- the upper bound for the number of states of a minimal DFA is $n + 1$.

A DFA for the language $L_?$ for this type of defect is received by removing the acceptance for all states of the defective automaton, and inserting a new accepting sink state to be the target for all missing transitions. All words accepted by this new complete DFA were accepted or rejected by the defective automaton, and, additionally, used the defect while being processed. This also gives an upper bound of $n + 1$ states for a minimal DFA accepting $L_?$.

Delete Non-Accepting State: For the defect of deleting a non-accepting state, we have similar constructions as for the defect that deletes an accepting state for the minimal DFA that accept the languages L_+, L_-, and $L_?$. This leads to the same upper bounds.

Remove Acceptance: For the defects that remove or add acceptance for a state in a minimal DFA, we assume for the construction of the DFA accepting the languages L_+, L_-, and $L_?$ that it is known which of the states is the defective one. Otherwise the constructions cannot be done and the languages can not be determined only from the knowledge of the defective automaton and the defect that occurs in this automaton.

The defect delete acceptance already leads to the acceptance of a subset of the language that is accepted by the original DFA. Therefore, the defective automaton itself accepts L_+. Since the considered defect does not insert any ambiguity to the DFA, and does not delete any transitions, the defective one is still complete and deterministic. In general the defective automaton may also be minimal. Thus, an

	L_+	L_-	$L_?$
Delete Transition	$n+1$	$n+1$	$n+1$
Insert Transition	$n+1$	$n+1$	$n+1$
Flip Transition	$n+1$	$n+1$	$n+1$
Exchange Symbol	$n+1$	$n+1$	$n+1$
Delete Non-Acc. State	$n+1$	$n+1$	$n+1$
Delete Accepting State	$n+1$	$n+1$	$n+1$
Remove Acceptance	n	n	n
Add Acceptance	n	n	n

Table 4.2: Upper bounds for the numbers of states of minimal DFA accepting the languages L_+, L_-, respectively $L_?$ for the different defects. Here, n is the number of states of the original automaton.

upper bound for the minimal DFA accepting the language L_+ is given by n, if the defective automaton consists of n states.

Also for this defect, the minimal DFA accepting the language L_- needs precisely as many states as the defective automaton. This is because when reverting the acceptance for all states except for the defective one, the language L_- is accepted. Therefore, the upper bound for L_- is also given by n states.

To construct a DFA accepting $L_?$ for this type of defect, the acceptance of all states of the defective automaton is removed. Then only the defective state, which is known, gets accepting. This new complete DFA accepts language $L_?$. Since this DFA may be minimal, an upper bound for a minimal DFA accepting $L_?$ is given by n.

Add Acceptance: For the defect add acceptance, we have similar constructions like for the defect remove acceptance for minimal DFA that accept the languages L_+, L_-, respectively $L_?$. This leads to the same upper bounds.

Table 4.2 summarises the results of this section.

5 Distances

In Chapter 3 it was shown that, starting from a minimal DFA that gets defective, nearly all the defects on the transitions lead to an exponential blow-up for the number of states of the minimal DFA accepting the defective language. It was also shown that this blow-up also happens for defects that lead to only minimal differences in the languages. From the point of view of the languages the defect may be not as bad as one may assume from the blow-up for the number of states for the accepting minimal DFA.

From Chapter 4 we know that it is impossible for most of the defects to correct the defect, even if the type of defect is known and only one appearance of this type of defect leads to the defective automaton. This makes it necessary to have some information about the original DFA, additionally to the information about the defective automaton at hand.

For the rest of this chapter, when dealing with defects, we assume that a defective automaton will be at hand. Additionally also the language accepted by the original DFA is available. This explicitly also includes the language that is accepted by the defective automaton.

In the following, we introduce a measure for distances between two languages. This measure is based on already existing measures for distances between words. Such measures were already investigated for example by [13]. Extending the distances between words, there exists the problem that, in general, languages consist of infinitely many words. Therefore, we restrict the distance between languages to finite subsets of the languages. We do this by restricting the distance to the sets of words having a length shorter than or equal to a fixed maximal length. This fixed length will be a parameter of the distance. Therefore, this distance is called parameterized. This measure and some of the results related to it were already published in [45].

After the introduction of this new distance measure, we apply it to regular languages and give upper and lower bounds for its range. Since we want to use this measure for the defects we already considered in the previous chapters, we also take a closer look at the distances below the upper bounds. For the decidability of the order of the distance, the census function is introduced. This function is closely related to a variant of the density function that is already investigated in [65]. We will show that the order of the census function is decidable. This result is used to decide the order of the distance.

The chapter concludes with a construction that allows to compute the precise distance for two regular languages. This is an improvement since the former results only lead to the order of the distance but do not provide the precise distance.

5.1 Introduction to Distances

Before introducing some specific distances, a general definition of a distance is provided. A distance over a set of words Σ^* is a function $d : \Sigma^* \times \Sigma^* \to \mathbb{N}_0 \cup \{\infty\}$, that satisfies the conditions

1. $d(x, y) = 0$ if and only if $x = y$,
2. $d(x, y) = d(y, x)$, and
3. $d(x, y) \leq d(x, z) + d(z, y)$,

for all $x, y, z \in \Sigma^*$.

One such distance is the so-called prefix distance $d_{\mathrm{pref}} : \Sigma^* \times \Sigma^* \to \mathbb{N}_0$, that determines the distance of two words w_1 and w_2 in Σ^* by summing up the number of all letters of the two words that do not belong to a common prefix. Formally the prefix distance is defined by

$$d_{\mathrm{pref}}(w_1, w_2) = |w_1| + |w_2| - 2\max\{|v| \mid w_1, w_2 \in v\Sigma^*\}.$$

It is a fact, that $d_{\mathrm{pref}}(w_1, w_2) = 0$ if and only if the words w_1 and w_2 are equal, and $d_{\mathrm{pref}}(w_1, w_2) = |w_1| + |w_2|$ if and only if the first letters of w_1 and w_2 are not the same. It is also a fact that the distance between two words can be large if one of their lengths is large and their common prefix is proportionally short.

Another such distance is the suffix distance $d_{\mathrm{suff}} : \Sigma^* \times \Sigma^* \to \mathbb{N}_0$, that is defined quite similar to the prefix distance by summing up the number of all letters of two words that do not belong to a common suffix of them instead of a prefix. Formally this is

$$d_{\mathrm{suff}}(w_1, w_2) = |w_1| + |w_2| - 2\max\{|v| \mid w_1, w_2 \in \Sigma^* v\},$$

for two words $w_1, w_2 \in \Sigma^*$.

Example 5.1.1. The distances defined above for the two words $w_1 = abab$ and $w_2 = abbab$ are given by

$$d_{\mathrm{pref}}(w_1, w_2) = |w_1| + |w_2| - 2\,|ab| = 5,$$

and

$$d_{\mathrm{suff}}(w_1, w_2) = |w_1| + |w_2| - 2\,|bab| = 3.$$

⌜⌝⌞⌟

Distances for words can be extended to distances between a word w and a language L by taking the minimum of the distances between w and all the words of L. For the prefix distance this extension is given by pref-$d : \Sigma^* \times 2^{\Sigma^*} \to \mathbb{N}_0$, which is defined by

$$\text{pref-}d(w, L) = \begin{cases} \min\{d_{\mathrm{pref}}(w, v) \mid v \in L\} & \text{if } L \neq \emptyset \\ \infty & \text{otherwise} \end{cases}.$$

In case w belongs to the language L, their prefix distance is zero, which means that pref-$d(w, L) = 0$.

For the suffix distance, the extension to a measure between a word and a language is defined in a similar way:

$$\text{suff-}d(w, L) = \begin{cases} \min\{d_{\text{suff}}(w, v) \mid v \in L\} & \text{if } L \neq \emptyset \\ \infty & \text{otherwise} \end{cases},$$

for a word w and a language L over some alphabet.

Example 5.1.2. Let $w = abab$ be a word and $L = abaa^*$ be a language. Their prefix distance is

$$\text{pref-}d(w, L) = |w| + |aba| - 2\,|aba| = 1,$$

since the prefix distance to any other word of L leads to a distance of at least 2. For the suffix distance we have

$$\text{suff-}d(w, L) = |w| + |aba| - 2\,|\lambda| = 7.$$

There exists no word in L that ends on b. This means, there does not exist a word in L having a suffix in common with w. Therefore, the whole length of w counts into its suffix distance to L. Also the whole length of the word in L is part of the suffix distance. This means, the distance is minimal for the shortest word aba of L. ⌜⌝⌞⌟

One possibility to extend such a distance between a word and a language to a distance between two languages L_1 and L_2 over an alphabet Σ is by taking the maximum of the suprema of the distances of all words from L_1 to L_2 and vice versa, see for example [13]. In general, it needs infinitely many comparisons to compute this distance since the languages do not need to be finite. This is the reason why we are interested in a parameterized definition, where the distance also depends on the length of the words.

In this thesis, we concentrate on the prefix and suffix distances and extend them to parameterized distances between languages. This has practical reasons. Quite a lot of defects only affect some short suffixes or prefixes. This means, the language of the original and the defective automaton have long common prefixes or suffixes, and only differ on a short part. This effect is respected in the choice of the word distance.

The parameterized prefix distance between two languages is the given by the function pref-$D : \mathbb{N}_0 \times 2^{\Sigma^*} \times 2^{\Sigma^*} \to \mathbb{N}_0 \cup \{\infty\}$ defined by

$$\text{pref-}D(n, L_1, L_2) := \sum_{\substack{w \in L_1, \\ 0 \leq |w| \leq n}} \text{pref-}d(w, L_2) + \sum_{\substack{w \in L_2, \\ 0 \leq |w| \leq n}} \text{pref-}d(w, L_1).$$

The parameterized suffix distance suff-$D : \mathbb{N}_0 \times 2^{\Sigma^*} \times 2^{\Sigma^*} \to \mathbb{N}_0 \cup \{\infty\}$ is defined in a similar way by

$$\text{suff-}D(n, L_1, L_2) := \sum_{\substack{w \in L_1,\\ 0 \le |w| \le n}} \text{suff-}d(w, L_2) + \sum_{\substack{w \in L_2,\\ 0 \le |w| \le n}} \text{suff-}d(w, L_1).$$

Example 5.1.3. Considering the languages $L_1 = \{aba\}\{a\}^*$ and $L_2 = \{aba\}\{b\}^*$, that is the languages that start with aba and end on an arbitrary number of a's respectively b's.

Computing the parameterized prefix distance of these languages up to some length $n \ge 0$ means to determine the distances of all the words of L_1 of length at most n to the language L_2, and vice versa.

All the words in L_1 that also belong to L_2 have prefix distance 0. There exists only the word aba that belongs to both languages. The words of L_1 not belonging to L_2 are summarised in the language described by the regular expression $abaaa^*$. The prefix distance of each of these words to language L_2 is minimal to the word aba. The distance then is computed by summing up all the symbols of the word in $abaaa^*$ minus the length of aba, which is 3. For each length greater than 3, there exists exactly one word in $abaaa^*$. Therefore, summing up all the prefix distances of such words to language L_2 leads to the following sum:

$$\sum_{\substack{w \in L_1 \setminus L_2,\\ 0 \le |w| \le n}} |w| - 3 = \sum_{i=4}^{n} i - 3 = \sum_{i=1}^{n-3} i = \frac{(n-3)(n-2)}{2}.$$

The prefix distance of L_2 to language L_1 is determined by the distances of the words in $ababb^*$ to L_1. The argumentation is similar to the one above for the prefix distance of each word of $L_2 \setminus L_1$ to L_1. Thus, the sum for the prefix distance of all those words to L_1 is

$$\sum_{\substack{w \in L_2 \setminus L_1,\\ 0 \le |w| \le n}} |w| - 3 = \sum_{i=4}^{n} i - 3 = \sum_{i=1}^{n-3} i = \frac{(n-3)(n-2)}{2}.$$

We obtain the parameterized prefix distance of the languages L_1 and L_2 by summing up the two sums from above.

$$\begin{aligned}\text{pref-}D(n, L_1, L_2) &= \sum_{\substack{w \in L_1 \setminus L_2,\\ 0 \le |w| \le n}} |w| - 3 + \sum_{\substack{w \in L_2 \setminus L_1,\\ 0 \le |w| \le n}} |w| - 3 \\ &= 2 \cdot \frac{(n-3)(n-2)}{2} = n^2 - 5n + 6\end{aligned}$$

⌜⌝
⌞⌟

A first result for the parameterized prefix distance is the following proposition. It may be used to analyse and construct regular languages with a certain distance by adding and removing words from given languages to start from.

Proposition 5.1.1. *Let $L_1, L_2 \subseteq \Sigma^*$ be two languages so that $L_1 \subseteq L_2$.*

1. *For a word $v \in L_2 \setminus L_1$, let $L_1' = L_1 \cup \{v\}$ and $L_2' = L_2 \setminus \{v\}$. Then*

$$\begin{aligned}\text{pref-}D(n, L_1, L_2) &> \text{pref-}D(n, L_1', L_2) \text{ and}\\ \text{pref-}D(n, L_1, L_2) &> \text{pref-}D(n, L_1, L_2').\end{aligned}$$

2. *For a word $v \in \Sigma^* \setminus L_2$, let $L_2' = L_2 \cup \{v\}$. Then*

$$\text{pref-}D(n, L_1, L_2) < \text{pref-}D(n, L_1, L_2').$$

3. *For a word $v \in L_1$, let $L_1' = L_1 \setminus \{v\}$. Then*

$$\text{pref-}D(n, L_1, L_2) < \text{pref-}D(n, L_1', L_2).$$

Proof. The parameterized prefix distance of the languages L_1 and L_2 is determined by the words in $L_2 \setminus L_1$ since the distance of any word in L_1 is equal to 0. Thus, the formula for the prefix distance is reduced to

$$\text{pref-}D(n, L_1, L_2) = \sum_{\substack{w \in L_2 \setminus L_1,\\ 0 \le |w| \le n}} \text{pref-}d(w, L_1).$$

1. Adding a word of $L_2 \setminus L_1$ to L_1 does not affect the sum of the prefix distances of the words from L_1 to L_2. This sum is still 0. On the other hand, for words $w \in L_2 \setminus L_1'$ we have

   $$\text{pref-}d(w, L_1') = \min\{\, d_{\text{pref}}(w, w') \mid w' \in L_1' \,\} \le \min\{\, d_{\text{pref}}(w, w') \mid w' \in L_1 \,\}$$

   since the new word v in L_1' can only affect the distance if there is a $w \in L_2$ so that $d_{\text{pref}}(w, v) < \text{pref-}d(w, L_1)$. However, the distance $\text{pref-}d(v, L_1)$ is at least 1, since $v \notin L_1$, and decreases to $\text{pref-}d(v, L_1') = 0$. We conclude $\text{pref-}D(n, L_1, L_2) > \text{pref-}D(n, L_1', L_2)$.

   In L_2' a word is missing from L_2 that may have a contribution to the prefix distance of L_1 and L_2. Thus, $\text{pref-}D(n, L_1, L_2')$ is less than $\text{pref-}D(n, L_1, L_2)$.

2. Similar as above, adding a new word to L_2 does not affect the sum of the prefix distances of the words from L_1 to L_2.

   For the new word $v \in L_2'$ we have $\text{pref-}d(v, L_1) \ge 1$, since v does not belong to L_1. So, $\text{pref-}D(n, L_1, L_2) < \text{pref-}D(n, L_1, L_2')$, because the prefix distances of words from L_2 to L_1 do not change by adding a word to L_2.

3. The removal of a word of the language L_1 results in the same situation as adding a new word to L_2. This is why the proposition follows also in this case.

□

5.2 The Paramerized Prefix Distance

In this section only the parameterized prefix distance is considered. The parameterized suffix distance is discussed in Section 5.3. In the following subsections, we will sometimes omit the word parameterized when talking about the parameterized prefix distance, if it is clear from the context.

5.2.1 Upper and Lower Bounds for the Prefix Distance

For the parameterized prefix distance it is interesting to know upper bounds and whether these bounds are tight, that is, whether they are optimal. We will give witness languages for which their prefix distance reaches the upper bound.

Before we can investigate an upper bound, it is necessary to know the maximally possible prefix distance between a word w and a language L. Since the parameterized prefix distance for two languages $L_1, L_2 \subseteq \Sigma^*$ is the sum of the prefix distances of all the words of L_1 up to a given length to the language L_2, and vice versa, this only requires finitely many comparisons for each word w. The following considerations confirm this fact.

Let w denote a word, and L a language, to which the prefix distance of w shall be computed. In case there exists no word in L that has a common prefix with w the prefix distance is minimised by computing the prefix distance to one of the shortest words in L. Here, all symbols of w and of the chosen shortest word of L sum up to the prefix distance.

If there exists any word in L having a prefix in common with w, not all the symbols of w count into its prefix distance to L. In the worst case, the words v in L having a common prefix with w have a suffix of a length such that the prefix distance of w and v succeeds the prefix distance of w and a shortest word of L. In this case, the minimal prefix distance for w and L is also given by the prefix distance between w and a shortest word of L.

From this we can conclude that in the worst case, the prefix distance between a word w and a language L is given by the length of w plus the length of a shortest word of L.

The following proposition is based on this observation.

Proposition 5.2.1. *Let $L_1, L_2 \subseteq \Sigma^*$ be two non-empty languages,*

$$m = \min\{\min\{\, |w| \mid w \in L_1 \,\}, \min\{\, |w| \mid w \in L_2 \,\}\}, \text{ and}$$

$$M = \max\{\min\{\, |w| \mid w \in L_1 \,\}, \min\{\, |w| \mid w \in L_2 \,\}\}.$$

Then $\text{pref-}D(n, L_1, L_2) \leq \sum_{i=m}^{n} |\Sigma|^i \cdot (i + M).$

Proof. The parameterized prefix distance between L_1 and L_2 is

$$\text{pref-}D(n, L_1, L_2) = \sum_{\substack{w \in L_1, \\ 0 \leq |w| \leq n}} \text{pref-}d(w, L_2) + \sum_{\substack{w \in L_2, \\ 0 \leq |w| \leq n}} \text{pref-}d(w, L_1).$$

We define $m_1 = \min\{\, |w| \mid w \in L_1 \,\}$ and $m_2 = \min\{\, |w| \mid w \in L_2 \,\}$ to be the lengths of the shortest words of L_1 and L_2, respectively. According to the observation made immediately before the proposition we obtain

$$\text{pref-}D(n, L_1, L_2) \leq \sum_{\substack{w \in L_1,\\ 0 \leq |w| \leq n}} |w| + m_2 + \sum_{\substack{w \in L_2,\\ 0 \leq |w| \leq n}} |w| + m_1.$$

Since there exist no words in $L_1 \cup L_2$ shorter than m the sums can start with length m. All words $w \in L_1 \cup L_2$ have a distance that is less or equal to $|w| + M$. Moreover, whenever a word belongs to the intersection of L_1 and L_2 it does not contribute to pref-$D(n, L_1, L_2)$ at all. So we can safely remove it from the sums. Thus, we have

$$\text{pref-}D(n, L_1, L_2) \leq \sum_{\substack{w \in L_1 \setminus L_2,\\ m \leq |w| \leq n}} |w| + M + \sum_{\substack{w \in L_2 \setminus L_1,\\ m \leq |w| \leq n}} |w| + M.$$

Since $(L_1 \setminus L_2) \cup (L_2 \setminus L_1) = (L_1 \cup L_2) \setminus (L_1 \cap L_2)$, there are at most $|\Sigma|^i$ words in $L_1 \cup L_2$ of length $m \leq i \leq n$ that contribute to pref-$D(n, L_1, L_2)$. Thus, we have

$$\text{pref-}D(n, L_1, L_2) \leq \sum_{i=m}^{n} |\Sigma|^i \cdot (i + M).$$

□

The necessary properties for languages matching the upper bound are stated in the following lemma.

Lemma 5.2.1. *Let L_1, L_2 be languages with*

$$m = \min\{\min\{\, |w| \mid w \in L_1 \,\}, \min\{\, |w| \mid w \in L_2 \,\}\} \text{ and}$$

$$M = \max\{\min\{\, |w| \mid w \in L_1 \,\}, \min\{\, |w| \mid w \in L_2 \,\}\}.$$

Then the upper bound of Proposition 5.2.1 is met only if (i) each word $w \in L_1 \cup L_2$ contributes $|w|+M$ to the prefix distance, (ii) $L_1 \cap L_2 = \emptyset$ if $m \geq 1$, and $L_1 \cap L_2 \subseteq \{\lambda\}$ if $m = 0$, and (iii) $L_1 \cup L_2 = \{\, w \in \Sigma^* \mid |w| \geq m \,\}$.

Proof. We assume $m \geq 1$ and $L_1 \cap L_2 \neq \emptyset$, or $m = 0$ and $L_1 \cap L_2$ is not a subset of $\{\lambda\}$. In both cases there exists at least one word w of length greater than or equal to $\max\{1, m\}$ that does not contribute to pref-$D(|w|, L_1, L_2)$. So there must be a word in $L_1 \cup L_2$ that contributes more than $|w| + M$ to the prefix distance. However, this is a contradiction to the choice of M to be the maximum of the sizes of the shortest words. So, (ii) is a necessary condition.

If $L_1 \cup L_2 \neq \{\, w \in \Sigma^* \mid |w| \geq m \,\}$, then there is a word not in $L_1 \cup L_2$ whose length is at least $\max\{1, m\}$. This word can be added to both languages L_1 and L_2 without affecting pref-$D(n, L_1, L_2)$. Since in this case the intersection $L_1 \cap L_2$ contains a non-empty word, we have a contradiction to (ii). This shows (iii).

At last we assume that there exists a word $w \in L_1 \cup L_2$ that contributes less than $|w| + M$ to pref-$D(|w|, L_1, L_2)$. Then there must be a word in $L_1 \cup L_2$ that contributes more than $|w| + M$ to the prefix distance. The same contradiction as for (ii) shows case (i). □

Lemma 5.2.1 shows that it is impossible to reach the upper bound if $m < M$. Let w be a word of a language L_2 with $|w| = M$, which means w is a shortest word of L_2. In case that $m < M$, then pref-$d(w, L_1) \leq |w| + m < |w| + M$. This violates condition (i) of the lemma.

The next proposition shows that the upper bound given in Proposition 5.2.1 is the best possible, which means that there exist languages that match the upper bound. Such languages need to satisfy the conditions stated in Lemma 5.2.1.

Proposition 5.2.2. *For any $M = m \geq 0$, there are binary regular languages $L_1, L_2 \subseteq \{a, b\}^*$ so that* pref-$D(n, L_1, L_2) = \sum_{i=m}^{n} |\Sigma|^i \cdot (i + M)$, *where m is the minimum and M is the maximum of the lengths of the shortest words in L_1 and L_2.*

Proof. For any $m \geq 1$, we use the disjoint regular languages $L_1 = \{a\}\{a, b\}^{m-1}\{a, b\}^*$ and $L_2 = \{b\}\{a, b\}^{m-1}\{a, b\}^*$ as witnesses. In particular, no two words of L_1 and L_2 have a common prefix.

Let $w \in L_1$ be some word. Its prefix distance to L_2 is $|w| + m$. Similarly, the prefix distance of every word $w \in L_2$ to the language L_1 is $|w| + m$. So we have

$$\text{pref-}D(n, L_1, L_2) = \sum_{\substack{w \in L_1 \setminus L_2, \\ m \leq |w| \leq n}} |w| + m + \sum_{\substack{w \in L_2 \setminus L_1, \\ m \leq |w| \leq n}} |w| + m.$$

Since $L_1 \cup L_2 = \{\, w \in \Sigma^* \mid |w| \geq m \,\}$ and $L_1 \cap L_2 = \emptyset$ this in turn is

$$\text{pref-}D(n, L_1, L_2) = \sum_{i=m}^{n} |\Sigma|^i \cdot (i + m).$$

If $m = 0$, then the empty word belongs to both languages. In this case we set $L_1 = \{\lambda\} \cup \{a\}\{a, b\}^*$ and $L_2 = \{\lambda\} \cup \{b\}\{a, b\}^*$ and obtain

$$\text{pref-}D(n, L_1, L_2) = \sum_{i=1}^{n} |\Sigma|^i \cdot i = \sum_{i=0}^{n} |\Sigma|^i \cdot (i + 0) = \sum_{i=m}^{n} |\Sigma|^i \cdot (i + m).$$

□

Up to this point we have only considered languages over at least binary alphabets. In the following we will show that the situation for unary languages is significantly different. The most important observation is that two unary words have a prefix distance to each other that is given only by the difference of their lengths. This leads to the next proposition.

Proposition 5.2.3. *Let $L_1 \subseteq \{a\}^*$ and $L_2 \subseteq \{a\}^*$ be two non-empty unary languages. Then* pref-$D(n, L_1, L_2) \leq \frac{n(n+1)}{2} + 1$.

Proof. By Proposition 5.2.1 we know that there is an upper bound on the parameterized prefix distance between two languages. So, let L_1 and L_2 be unary languages whose distance is as large as possible.

If there is some word w in $L_1 \cap L_2$ then we can remove it from one of the languages, say from L_1, and obtain a larger distance. The reason is that now $w \in L_2$ contributes to the distance. Moreover, the distance of the remaining words from L_1 to L_2 are not affected, and the contributions from words of L_2 can never be decreased, when some word from L_1 is deleted. So, we conclude that $L_1 \cap L_2 = \emptyset$.

Next, let m_1 be the length of the shortest word in L_1 and m_2 be the length of the shortest word in L_2. Without loss of generality let $m_1 \leq m_2$. If $0 < m_1$, then we can add the words $\lambda, a, a^2, \ldots, a^{m_1-1}$ to L_1. In this way the contribution of words from L_2, and those from L_1 are not affected. In addition, the new words contribute to the distance. So, the distance between L_1 and L_2 would be increased. Therefore, we conclude that λ already belongs to one of the languages.

Let $\lambda \in L_1$. Since L_2 is non-empty, $1 \leq m_2$ exists. The contribution of any word w from $L_1 \cup L_2$ whose length is at least $m_2 + 1$ is at most $|w|$. Now we consider the contributions of the words $\lambda, a, a^2, \ldots, a^{m_2} - 1$ that may or may not belong to L_1, and the word $a^{m_2} \in L_2$. If all words do belong to L_1 we obtain the overall contribution of $\sum\limits_{i=1}^{m_2} i$ for the words from L_1 and 1 for a^{m_2}. If one of the words do not belong to L_1 this amount decreases.

Altogether we have

$$\text{pref-}D(n, L_1, L_2) \leq 1 + \sum_{i=1}^{m_2} i + \sum_{i=m_2+1}^{n} i = 1 + \sum_{i=1}^{n} i = \frac{n(n+1)}{2} + 1.$$

□

Just like in the general case, the upper bound for the parameterized prefix distance in the unary case can be matched, that means this bound is tight.

Proposition 5.2.4. *There are unary regular languages $L_1, L_2 \subseteq \{a\}^*$ so that their prefix distance is* pref-$D(n, L_1, L_2) = \frac{n(n+1)}{2} + 1$.

Proof. Let $L_1 = aa^*$ and $L_2 = \{\lambda\}$. These languages are unary, regular, and disjoint. Therefore, the prefix distance of each word in $w \in L_1$ to L_2 is $|w|$. For the only word λ in L_2 its distance to L_1 is $d_{\text{pref}}(\lambda, a) = 1$. So we have

$$\text{pref-}D(n, L_1, L_2) = 1 + \sum_{i=1}^{n} i = \frac{n(n+1)}{2} + 1.$$

□

5.2.2 Distances Below the Upper Bound

In Section 5.2.1 we have investigated upper bounds for languages over either unary or at least binary alphabets. We have given examples for worst case languages which have a parameterized prefix distance that match these upper bounds.

In this section we concentrate on regular languages. We explore which functions are possible to obtain for the parameterized prefix distance for this language class. The next proposition gives an example that every polynomial is exceeded when measuring the parameterized prefix distance for regular languages.

Proposition 5.2.5. *There are regular languages L_1 and L_2 even over a binary alphabet so that* pref-$D(n, L_1, L_2) \in \Theta(n2^n)$.

Proof. Here we can use the witness languages L_1 and L_2 from the case $m = 0$ in the proof of Proposition 5.2.2. There,

$$\text{pref-}D(n, L_1, L_2) = \sum_{i=1}^{n} |\Sigma|^i \cdot i = \sum_{i=1}^{n} 2^i \cdot i.$$

has been shown. This sum is equal to $n2^{n+2} - (n+1)2^{n+1} + 2 \in \Theta(n2^n)$. □

For any constant $c \geq 1$, there exist regular languages with parameterized prefix distance c.

Proposition 5.2.6. *Let $c \geq 1$ be an integer. Then there are unary regular languages L_1 and L_2 so that* pref-$D(n, L_1, L_2) = c$*, for all $n \geq c$.*

Proof. We use the languages $L_1 = \{\lambda\}$ and $L_2 = \{\lambda, a^c\}$ as witnesses. Since $\lambda \in L_1 \cap L_2$ the empty word in L_1 and L_2 does not contribute to the distance between L_1 and L_2. It is that pref-$d(a^c, L_1) = c$ and, thus, pref-$D(n, L_1, L_2) = c$, for all $n \geq c$. □

Note that in the proof of Proposition 5.2.6, for all $n < c$, the distance of L_1 and L_2 is zero.

Another possible class of functions for the parameterized prefix distance are the polynomials. In the following theorem we show that given any arbitrary polynomial p having integer coefficients and a positive leading coefficient, p can be the parameterized prefix distance of two regular languages. Furthermore, in the proof of the theorem we explicitly construct such two regular languages. It does not make sense to consider a negative leading coefficient since such a polynomial would characterise a negative distance that does not exist at all.

Theorem 5.2.1. *Let $p(n) = x_k \cdot n^k + x_{k-1} \cdot n^{k-1} + \cdots + x_0$ be a polynomial of degree $k \geq 0$ with integer coefficients x_i, $0 \leq i \leq k$, and $x_k \geq 1$. Then two regular languages L_1 and L_2 over the alphabet $\{a, b\}$ can effectively be constructed so that* pref-$D(n, L_1, L_2) = p(n)$*, for all $n \geq n_0$, where n_0 is some constant.*

Proof. Proposition 5.2.6 already shows the special case $k = 0$. Therefore, we assume $k \geq 1$. The basic idea of the construction is to start with two languages whose distance is already a polynomial of degree k, but its coefficients may be incorrect. Subsequently, the coefficients are corrected one after the other, from x_k to x_0. When coefficient x_i is corrected, the coefficients x_k to x_{i+1} are not affected while the coefficients x_{i-1} to x_0 may be changed.

In general, language L_1 will always be a subset of L_2. In this way, the words from L_1 never contribute to the distance.

For the corrections of the coefficients a set of equally long prefixes is used. So, we define $P \subseteq \{a, b\}^l$, for some constant l, with $P = \{p_0, p_1, \ldots, p_m\}$. Assume for a moment that $l \geq k$ is large enough to perform the following constructions. Later we will give evidence that it always can be chosen appropriately.

We consider auxiliary languages

$$L_{r,-1} = \{\, p_r \,\} \text{ and } L_{r,-1,b} = L_{r,-1} \cup L_{r,-1}b,$$
$$L_{r,s} = \{\, p_r v \mid v \in \{a, b\}^*, |v|_b = s \,\} \text{ and } L_{r,s,b} = L_{r,s} \cup L_{r,s}b$$

for $s \geq 0$ and $p_r \in P$. There are $\binom{n-|p_r|}{s} = \binom{n-l}{s} \in \Theta(n^s)$ many words of length n in the languages $L_{r,s}$. Considering the distance between $L_{r,-1}$ and $L_{r,-1,b}$ we obtain pref-$D(n, L_{r,-1}, L_{r,-1,b}) = 1$. For the distance between $L_{r,s}$ and $L_{r,s,b}$, all words from $L_{r,s}b$ contribute 1 while the words from $L_{r,s}$ contribute nothing. For $s \geq 1$, we obtain

$$\text{pref-}D(n, L_{r,s-1}, L_{r,s-1,b}) = \sum_{i=1}^{n} \binom{i-l}{s-1} = \binom{n-l+1}{s}$$
$$= \frac{(n-l+1)\cdot(n-l)\cdot(n-l-1)\cdots(n-l-s+2)}{s!}$$

which gives us a term of the form $\frac{n^s + y_{s-1}\cdot n^{s-1} + y_{s-2}n^{s-2} + y_{s-3}n^{s-3} + \cdots + y_0}{s!}$, where a rough and simple estimation yields $|y_i| \leq 3^s \cdot l^s$, $0 \leq i \leq s-1$.

We start the construction by using the union of auxiliary languages with $x_k \cdot k!$ many different prefixes, that is,

$$L_1 = \bigcup_{i=0}^{x_k\cdot k!-1} L_{i,k-1} \text{ and } L_2 = \bigcup_{i=0}^{x_k\cdot k!-1} L_{i,k-1,b}.$$

So, we start with a distance of the form

$$x_k n^k + z_{k-1}n^{k-1} + z_{k-2}n^{k-2} + z_{k-3}n^{k-3} + \cdots + z_0,$$

where x_k is already the correct coefficient and $|z_i| \leq x_k \cdot 3^k \cdot l^k$, $0 \leq i \leq k-1$.

Next we correct the remaining coefficients. Let $x_{max} = \max\{\, x_i \mid 0 \leq i \leq k \,\}$. Concluding inductively, we assume that currently

$$\text{pref-}D(n, L_1, L_2) = x_k n^k + x_{k-1}n^{k-1} + \cdots + x_{k-i+1}n^{k-i+1} + z_{k-i}n^{k-i} + \cdots + z_0,$$

where the coefficients $x_k, x_{k-1}, \ldots, x_{k-i+1}$ are already correct and, moreover, we have that $|z_{k-i}|, |z_{k-i-1}|, \ldots, |z_0| \leq 3^{i-1} \cdot x_{max} \cdot (3^k \cdot l^k)^i$.

In order to obtain the correct coefficient x_{k-i}, we set $d = z_{k-i} - x_{k-i}$ and distinguish the two cases, where d is negative or positive. If $d = 0$, the coefficient x_{k-i} is already correct and nothing has to be done.

If $d < 0$, the distance has to be increased. To this end, the auxiliary languages $L_{j,k-i-1}$ and $L_{j,k-i-1,b}$ are used. We add their unions with $|d| \cdot (k-i)!$ many new different prefixes to L_1 and L_2, that is,

$$\bigcup_{j=0}^{|d|\cdot(k-i)!-1} L_{j,k-i-1} \text{ is added to } L_1 \text{ and } \bigcup_{j=0}^{|d|\cdot(k-i)!-1} L_{j,k-i-1,b} \text{ is added to } L_2.$$

Since all the prefixes p_j are new and $L_1 \subseteq L_2$, again all words from L_2 contribute 1 to the distance while the words in L_1 contribute nothing. In particular, we have added $|d|n^{k-i} + z'_{k-i-1}n^{k-i-1} + z'_{k-i-2}n^{k-i-2} + \cdots + z'_0$ words up to length n to L_2, where $|z'_{k-i-1}|, |z'_{k-i-2}|, \ldots, |z'_0| \leq |d| \cdot 3^{k-i} \cdot l^{k-i} \leq |d| \cdot 3^k \cdot l^k$. This implies

$$\text{pref-}D(n, L_1, L_2) = x_k n^k + x_{k-1}n^{k-1} + \cdots + x_{k-i}n^{k-i} + z_{k-i-1}n^{k-i-1} + \cdots + z_0,$$

where $x_k, x_{k-1}, \ldots, x_{k-i}$ are already correct and $|z_{k-i-1}|, |z_{k-i-2}|, \ldots, |z_0|$ are at most

$$\begin{aligned}
&3^{i-1} \cdot x_{max} \cdot (3^k \cdot l^k)^i + |d| \cdot 3^k \cdot l^k \\
&= 3^{i-1} \cdot x_{max} \cdot (3^k \cdot l^k)^i + (3^{i-1} \cdot x_{max} \cdot (3^k \cdot l^k)^i + x_{max}) \cdot 3^k \cdot l^k \\
&= 3^{i-1} \cdot x_{max} \cdot (3^k \cdot l^k)^i + 3^{i-1} \cdot x_{max} \cdot (3^k \cdot l^k)^i \cdot 3^k \cdot l^k + x_{max} \cdot 3^k \cdot l^k \\
&\leq 3^i \cdot x_{max} \cdot (3^k \cdot l^k)^{i+1}.
\end{aligned}$$

This concludes the first case.

If $d > 0$, the distance has to be decreased. To this end, words from L_2 are added to L_1 so that they do not contribute to the distance anymore. Let $\tilde{p}$ be one of the $x_k \cdot k!$ prefixes used at the beginning of the induction to establish a polynomial distance of degree k. Moreover, we may assume that $\tilde{p}$ has not been used for the current purpose before.

Then, for $r, t \geq 0$ and $s \geq r$, another auxiliary regular language is defined as $\tilde{L}_{\tilde{p},r,s,t} = \{\, \tilde{p}ub^r v \mid uv \in \{a,b\}^*, |u| = t, |uv|_b = s-r \,\}$. Here, we set $\tilde{L}_{\tilde{p},r,r-1,t} = \{\, \tilde{p}b^r \,\}$. In these languages the position of the block b^r is fixed. This means, that the union $\bigcup_{j=0}^{d\cdot(k-i)!-1} \tilde{L}_{\tilde{p},i,k-1,j}$ contains $dn^{k-i} + z'_{k-i-1}n^{k-i-1} + z'_{k-i-2}n^{k-i-2} + \cdots + z'_0$ words up to length n, for $n \geq d \cdot (k-i)! + l + i$, and where it holds true that $|z'_{k-i-1}|, |z'_{k-i-2}|, \ldots, |z'_0| \leq d \cdot 3^{k-i} \cdot (l+i)^{k-i} \leq d \cdot 3^k \cdot l^k$. Now all these words are concatenated with a symbol b and are added to L_1. Since all words do belong to L_2 as well, we obtain

$$\text{pref-}D(n, L_1, L_2) = x_k n^k + x_{k-1}n^{k-1} + \cdots + x_{k-i}n^{k-i} + z_{k-i-1}n^{k-i-1} + \cdots + z_0,$$

where the coefficients $x_k, x_{k-1}, \ldots, x_{k-i}$ are already correct and analogously to the first case $|z_{k-i-1}|, |z_{k-i-2}|, \ldots, |z_0|$ are at most $3^i \cdot x_{max} \cdot (3^k \cdot l^k)^{i+1}$. This concludes the second case.

The construction is concluded by the observation that choosing $n_0 > l + k$ is sufficient for the auxiliary languages applied in the initial step and the correction steps in the first case. For the corrections in the second case

$$d \cdot (k-i)! + l + i \leq 3^{k+1} \cdot x_{max} \cdot (3^{k^2} \cdot l^{k^2}) \cdot k! \leq n_0$$

is sufficient.

Finally, it has to be shown that the prefix length l always can be chosen appropriately. In the first step, $x_k \cdot k!$ many prefixes are used. For the correction steps, no additional prefix is used in the second case, and $|d| \cdot (k-i)!$ prefixes in the first case. The latter is less than

$$(3^{i-1} \cdot x_{max} \cdot (3^k \cdot l^k)^i + x_{max}) \cdot (k-i)! \leq 3^k \cdot x_{max} \cdot 3^{k^2} \cdot l^{k^2} \cdot k!.$$

Therefore, altogether less than $3^k \cdot x_{max} \cdot 3^{k^2} \cdot l^{k^2} \cdot (k+1)!$ many prefixes are necessary. On the other hand, there are 2^l prefixes of length l. So it is sufficient to choose l large enough so that $2^l \geq 3^k \cdot x_{max} \cdot 3^{k^2} \cdot l^{k^2} \cdot (k+1)!$ which is always possible since k and x_{max} are constants and on the right-hand side there is only a polynomial in l. □

5.2.3 Decidability of the Order of the Distances

Now that we know possible functions for the parameterized prefix distance, it is interesting to decide the order of magnitude of the prefix distance for given regular languages. This is also of interest both from the practical and from the theoretical point of view.

For the parameterized prefix distance, only the words belonging to the symmetric difference of the considered languages are important. All of the other words belong to both languages and contribute only 0 to the parameterized prefix distance. Therefore, it is of interest to know the number of words up to a certain length that do not belong to both languages. The function that counts the number of words for a certain length n is the density function. This function is already mentioned and explored amongst others in [65] and [67]. Formally this is the function $\varrho_L : \mathbb{N}_0 \to \mathbb{N}_0$ defined by

$$\varrho_L(n) = |L \cap \Sigma^n| = |\{ w \in L \mid |w| = n \}|.$$

The so called census function $\mathrm{cens}_L : \mathbb{N}_0 \to \mathbb{N}_0$, that is defined as

$$\mathrm{cens}_L(n) = \sum_{i=0}^{n} \varrho_L(i) = |\{ w \in L \mid |w| \leq n \}|,$$

is a function that counts all words up to a fixed length n. These two functions are closely related.

In [65], a regular language $L \subseteq \Sigma^*$ is said to have a polynomial density if $|L \cap \Sigma^n|$ belongs to $O(n^k)$ for some integer $k \geq 0$. This does not include the lower bound, it is only an upper bound. This means, the authors only considered the limes superior but not the limes inferior. Like already stated in [45], this may lead to different densities for different n. An example for this behaviour can be seen for the language $R_k = \{w \in \{a, b\}^* \mid |w|_a = k + 1 \text{ and } |w| \text{ is even}\}$. For the density we have that $\varrho_{R_k}(n) \in O(n^{k+1})$ if n is even, but $\varrho_{R_k}(n) = 0$ if n is odd. This especially means that the density is neither in $O(n^k)$ nor in $\Omega(n^{k+1})$.

To be able to deduce some of the results for the census function, we need to adjust this and consider the factor n when talking about polynomial densities. Therefore, in the following we call a density to be polynomial, if the function mapping n to $\max\{\varrho(i) \mid 0 \le i \le n\}$ is of order $\Theta(n^k)$ for some $k \ge 1$.

The following definition and the two lemmas were already used and shown in [65]. For the sake of completeness we will shortly recall these proofs in a version adapted to our notion of the density function.

In the following definition we exchange the name t-tiered the authors used in the original definition by t-looped.

Definition 5.2.1. Let $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F\rangle$ be a DFA. A word w in Σ^* is said to be t-looped with respect to $\mathcal{A}$ for $t \ge 0$, if the state transition sequence of w is given by

$$\alpha\beta_1^{d_1}\gamma_1 \cdot \beta_2^{d_2}\gamma_2 \cdots \beta_t^{d_t}\gamma_t,$$

where

(i) $\alpha = p_1p_2 \cdots p_l, 0 \le l \le |Q|$, where the p's are states in Q,

and for each i between 1 and t

(ii) $\beta_i = q_{i,0}q_{i,1} \cdots q_{i,k_i}$ and $\gamma_i = q_{i,0}r_{i,1}r_{i,2} \dots r_{i,l_i}, 0 \le k_i, l_i \le |Q|$, where the q's and r's are states in Q,

(iii) $q_{i,0}$ appears only as the first state in β_i and in γ_i,

(iv) $d_i > 0$.

Lemma 5.2.2. *Let L be accepted by a DFA $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F\rangle$. If there exists a word $w \in L$ such that w is k-looped with respect to $\mathcal{A}$, then the density of L is $\Omega(n^{k-1})$.*

Proof. According to the state transition sequence $\alpha\beta_1^{d_1}\gamma_1 \cdot \beta_2^{d_2}\gamma_2 \cdots \beta_k^{d_k}\gamma_k$ of w with respect to $\mathcal{A}$, w can be written as $w = xy_1^{d_1}z_1y_2^{d_2}z_2 \dots y_k^{d_k}z_k$ such that x is processed by sequence α, y_i is processed exactly once by one cycle β_i, and z_i is processed by sequence γ_i for $1 \le i \le k$. We define the length $n = |xz_1z_2 \cdots z_k| + tC$ for an arbitrary integer $t > 0$, and $C = |y_1| \cdot |y_2| \cdot \dots \cdot |y_k|$. Then for any k arbitrary non-negative integers $t_1, t_2, \dots, t_k$ such that $t_1 + t_2 + \dots + t_k = t$, the word

$$xy_1^{\frac{C}{|y_1|}t_1}z_1 \cdot y_2^{\frac{C}{|y_2|}t_2}z_2 \cdots y_k^{\frac{C}{|y_k|}t_k}z_k$$

is in L and of length n. Let $N_k(t)$ denote the set consisting of all such k-tuples $(t_1, t_2, \dots, t_k)$, that is

$$N_k(t) = \{(t_1, t_2, \dots, t_k) \mid t_1, t_2, \dots, t_k \ge 0, t_1 + t_2 + \dots + t_k = t\}.$$

This set consists of $\binom{t+k-1}{k-1}$ tuples, since these k-tuples are simply ordered partitions of the number t (see for example [64]). Since t is linear in n, we have

$$\binom{t+k-1}{k-1} \in \Omega(t^{k-1}) \in \Omega(n^{k-1}).$$

To prove that the number of distinct words in L of length n is also in $\Omega(n^{k-1})$ we need to show that the words produced by different tuples from $N_k(t)$ are different again. This can easily be done by comparing the different state transition sequences with respect to $\mathcal{A}$ and concluding that these sequences differ in the number of appearances of at least one state $q_{i,0}$, $1 \leq i \leq k$. $\square$

For our definition of the density of a regular language, this proves that if there exists a k-looped word, then the density function is bounded from below by a polynomial.

Lemma 5.2.3. *Let L be an arbitrary regular language with $L = L(\mathcal{A})$ for some DFA $\mathcal{A}$. If the density of L is $O(n^k)$ for some $k \geq 0$, then each word $w \in L$ is t-looped with respect to $\mathcal{A}$ for some non-negative integer $t \leq k+1$.*

Proof. Let $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F\rangle$ be a DFA accepting L. The proof is given by induction on the length of a word in L.

If the length of any word of L is shorter than $|Q|$, the lemma holds trivially. So the induction hypothesis is that the lemma holds for all words w in L with $|w| < n$ for some $n \geq |Q|$.

Let $w \in L$ be an arbitrary word of length n. Taking a look at the state transition sequence of w with respect to $\mathcal{A}$, we reverse this sequence and denote by s the first state that repeats. By $s^{(i)}$ we denote the ith appearance of s in the reversed sequence. Let $s^{(1)}, s^{(2)}, \ldots, s^{(h+1)}$ be the sequence of all appearances of s in the non-reversed state transition sequence.

We have that $h \geq 1$ and there exist less than $|Q|$ states between $s^{(h)}$ and $s^{(h+1)}$. Let $w = w_0 v_1 v_2 \cdots v_h w_1$ such that $\delta(s^{(i)}, v_i) = s^{(i+1)}$, for $1 \leq i \leq h$, $w_0, w_1 \in \Sigma^*$, and $v_1, v_2, \ldots, v_h \in \Sigma^+$. Then also all words $w_0 v_1^{n_1} v_2^{n_2} \cdots v_h^{n_h} w_1$ belong to L for arbitrary non-negative integers $n_1, n_2, \ldots, n_h$.

To prove that w is t-looped for some $t \leq k+1$, we need to show $v_1 = v_2 = \cdots = v_h$. Assuming $v_i \neq v_j$ for some $i \neq j$, v_i cannot be a prefix of v_j since then there would exist another appearance of s between $s^{(j)}$ and $s^{(j+1)}$ which is impossible. Similarly, v_j is no prefix of v_i. The language described by the regular expression $(v_i v_j + v_j v_i)^g$ for any non-negative integer g consists of 2^g different words of the same length $|v_i v_j| \cdot g$. All words represented by the regular expression $w_0(v_i v_j + v_j v_i)^g w_1$ have the same length $|w_0 w_1| + |v_i v_j| \cdot g$. They all belong to L. This is a contradiction to the fact that L has only polynomial density. Thus, $v_1 = v_2 = \cdots = v_h$ and also the parts of the state transition sequence of w with respect to $\mathcal{A}$ produced by the v_i, for $1 \leq i \leq h$, are equal.

The word $w' = w_0w_1$ belongs to L. Since $|w'| < n$, w' is t-looped with respect to $\mathcal{A}$ for some $t \leq k+1$ due to the induction hypothesis. Then its state transition sequence with respect to $\mathcal{A}$ satisfies the conditions of Definition 5.2.1. Let $\alpha\beta_1^{d_1}\gamma_1\beta_2^{d_2}\gamma_2 \cdots \beta_t^{d_t}\gamma_t$ denote this sequence. State s from above must appear only in γ_t and no $q_{i,0}$ may appear in the state transition sequence of w_1 with a similar argument as above. This means, all $q_{i,0}, 1 \leq i \leq t$, appear before the first appearance of s. In case that $t < k+1$, the proof is complete. Assuming that $t = k+1$, then w is $(k+2)$-looped, since its state transition sequence with respect to $\mathcal{A}$ consists of one more sequence β^d. By Lemma 5.2.2, the density of L is in $\Omega(n^{k+1})$, which contradicts the premise. Thus, $t = k+1$ is impossible. □

By Lemma 5.2.3 we know for a regular language L with a density in $O(n^k)$ for some $k \geq 0$, that every word of L is t-looped for some $t \leq k+1$. Lemma 5.2.2 then gives us a lower bound of $\Omega(n^k)$. This is because the distance between two lengths with a polynomial lower bound of the same degree is constant. From the proof of Lemma 5.2.2 we can choose the length n to be $|xz_1z_2 \cdots z_{k+1}| + tC$, where $x, z_1, z_2, \ldots, z_{k+1}, t$, and C are defined as in the proof. For this length there exist $\Omega(n^k)$ many different words in the language. Then the length n' set to $|xz_1z_2 \cdots z_{k+1}| + (t+1)C$ can also be defined for which $\Omega((n')^k)$ words belong to the language. This also holds true for a length $|xz_1z_2 \cdots z_{k+1}| + (t+x)C$, where $x \geq 1$. The distance between two of such lengths is simply C, which is a constant. Therefore, there always exists a length in $\Theta(n)$ for arbitrary n, whose lower bound is in $\Omega(n^k)$ if the density of the language belongs to $O(n^k)$.

These two lemmas are necessary to inherit the results from [65] for the density function since their proofs are based on these conclusions. From the results in [65] it follows that it is decidable whether the upper bound for the density function of a regular language is constant, polynomial, or exponential, where exponential means, the density function is of the form $2^{\Omega(n)}$.

The next proposition shows that it is decidable whether the census function of a regular language is ultimately constant.

Proposition 5.2.7. *Let $\mathcal{A}$ be a minimal DFA. Then it is decidable whether* $\text{cens}_{\mathcal{A}}$ *is ultimately constant.*

Proof. The function $\text{cens}_{\mathcal{A}}$ is ultimately constant if and only if $\mathcal{A}$ accepts a finite language. The finiteness of a regular language is decidable by checking whether each accepting path of $\mathcal{A}$ is acyclic. □

In [65] it was shown, that there exist gaps for the density of regular languages. For any $k \geq 0$, there exists no regular language for which the density belongs to $\omega(n^k) \cap o(n^{k+1})$, and no regular language whose density is superpolynomial and in $2^{o(n)}$. This especially means, there exists no density function of order $\Theta(\sqrt{n})$, $\Theta(n \log(n))$, or $\Theta(2^{\sqrt{n}})$.

Due to the proven fact that the density function is either constant, polynomial, or exponential, the next corollary follows.

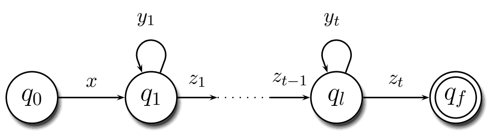


Figure 5.1: Structure of minimal DFA used in the proof of Theorem 5.2.2.

Corollary 5.2.1. *The census function of every regular language is either ultimately constant, polynomial, or exponential.*

Proof. By definition we obtain the census function cens(n) by summing up the densities up to n. Summing up polynomials of degree $k \geq 0$ gives a polynomial at most of degree $k+1$. Similarly, summing up exponential functions of the form $2^{\Omega(n)}$ gives again an exponential function of that form. □

It is possible to retrieve the degree of the density function for a language with polynomial density from the minimal DFA accepting the language. This is not explicitly stated in [65], but can be followed from the results stated there.

Theorem 5.2.2. *Given a language with polynomial density, the degree of the density function can be retrieved from the minimal DFA accepting the language.*

Proof. In [65] it was shown that for a language over an alphabet Σ with polynomial density of degree $k \geq 0$ the language can be described by a finite union of regular expressions of the form

$$xy_1^* z_1 \cdot y_2^* z_2 \cdots y_t^* z_t,$$

where $x, y_1, z_1, y_2, z_2, \ldots, y_t, z_t \in \Sigma^*$, $t \leq k+1$, and all of theirs lengths $|x|$, $|y_1|$, $|z_1|$, $|y_2|$, $|z_2|, \ldots, |y_t|, |z_t|$ are smaller than or equal to $|Q|$, where Q is the set of states of the minimal DFA accepting the language. Each of these regular expressions can be accepted by a DFA of the following form that is depicted in Figure 5.1.

The automaton accepting the whole language consists of finitely many parts of this form. Since we can count the number of loops of all these parts, we can also determine the maximum of all these numbers. From this number we obtain the degree of the density function of the language. □

We can derive directly that the degree of the polynomial for the census function can also be determined by inspecting the structure of the automaton. This is because the degree of the polynomial for the census function is nearly the same as for the density, it is just greater by one.

Corollary 5.2.2. *Given a language with polynomial density, the degree of the census function can be retrieved from the minimal DFA accepting the language.*

The results in [65] imply a decision procedure for the question whether the census function of a regular language is polynomial or exponential. It is even possible to compute the degree, if the census function is polynomial.

Theorem 5.2.3. *Let $\mathcal{A}$ be a DFA. Then it is decidable whether* $\text{cens}_{\mathcal{A}}$ *is exponential or a polynomial. If it is a polynomial, the degree can be computed.*

Proof. If $L(\mathcal{A})$ is a unary language, then $\text{cens}_{\mathcal{A}}$ is either constant or linear. By Theorem 5.2.7 we can decide whether it is ultimately constant. If not by the results in [65] it can be decided whether $\varrho_{\mathcal{A}}$ is exponential or polynomial, where in the latter case the degree of the polynomial is computable. From the orders of the density we can derive the order of $\text{cens}_{\mathcal{A}}$. □

Now that we are able to decide whether the census function is constant, polynomial, or exponential, we can use this knowledge to decide the order of magnitude of the parameterized prefix distance for regular languages. As already mentioned before, the symmetric difference of the languages is crucial for the computation of their prefix distance. And we also already know that for the prefix distance of a word w and a language L it is true that $1 \leq \text{pref-}d(w, L) \leq |w| + |m|$, where m is a shortest word of L, and $w \notin L$.

Theorem 5.2.4. *Let L_1 and L_2 be two regular languages. Then it is decidable whether the parameterized prefix distance* $\text{pref-}D(n, L_1, L_2)$ *is ultimately constant.*

Proof. The family of regular languages is effectively closed under symmetric difference. So, a representation, say a DFA $\mathcal{A}$, accepting $L_1 \oplus L_2$ can effectively be constructed from DFA accepting L_1 and L_2. If $L_1 \oplus L_2$ is finite, then $\text{pref-}D(n, L_1, L_2)$ is ultimately constant. Conversely, if $L_1 \oplus L_2$ is infinite, then $\text{pref-}D(n, L_1, L_2)$ cannot be bounded by a constant, since all the infinitely many words in the symmetric difference contribute at least 1 to the distance. Now the theorem follows from the decidability of finiteness of regular languages. □

Theorem 5.2.5. *Let L_1 and L_2 be two regular languages. Then it is decidable whether the parameterized prefix distance* $\text{pref-}D(n, L_1, L_2)$ *is exponential.*

Proof. As in the proof of Theorem 5.2.4 we may assume without loss of generality that a DFA $\mathcal{A}$ accepting $L_1 \oplus L_2$ can effectively be constructed from L_1 and L_2. Moreover, one can decide whether $\text{pref-}D(n, L_1, L_2)$ is ultimately constant. So, assume that it is not.

Any word $|w|$ in the symmetric difference contributes at least 1 and at most $|w|+|s|$ to the distance, where s is the shortest word in the language w does not belong to. Therefore, we know $\text{cens}_{\mathcal{A}}(n) \leq \text{pref-}D(n, L_1, L_2) \leq (c+n) \cdot \text{cens}_{\mathcal{A}}(n)$, where c is the maximum of the lengths of the shortest words in L_1 and L_2. Since $\text{cens}_{\mathcal{A}}$ can only be ultimately constant, polynomial, or exponential, $\text{pref-}D(n, L_1, L_2)$ is exponential if and only if $\text{cens}_{\mathcal{A}}$ is exponential. Now the theorem follows from the possibility to decide whether $\text{cens}_{\mathcal{A}}$ is exponential. □

Theorem 5.2.6. *Let L_1 and L_2 be two regular languages and $k \geq 1$ be a constant. Then it is decidable whether the parameterized prefix distance* pref-$D(n, L_1, L_2)$ *belongs to $\Omega(n^k) \cap O(n^{k+1})$.*

Proof. First it is decided whether pref-$D(n, L_1, L_2)$ is ultimately constant or exponential. If this is neither of these, both census functions $\text{cens}_{L_1 \setminus L_2}$ and $\text{cens}_{L_2 \setminus L_1}$ are ultimately constant or polynomial. Theorem 5.2.5 shows that the degree k of the polynomial can be computed. With the fact, that each word $|w|$ contributes at least 1 and at most $|w| + |s|$ to the distance, where s is the shortest word in the language w does not belong to, we derive pref-$D(n, L_1, L_2) \in \Omega(n^k) \cap O(n^{k+1})$. □

5.2.4 Computation of the Precise Distance

In the previous section it was shown that the order of the prefix distance of two regular languages is decidable. But due to the inaccuracy of the estimation, this degree cannot be determined exactly by the former results if it is polynomial.

In the rest of this section, a construction will be presented to compute the distance of two given regular languages accurately. Let $L_1, L_2 \subseteq \Sigma^*$ be two arbitrary regular languages whose parameterized prefix distance is to be computed.

Before turning to the constructions used in the computation, we have a look at some examples for the different possibilities for the computation of the shortest prefix distance between a word $w \in L_1$ and a language L_2.

Example 5.2.1. The prefix distance between a word $w \in L_1$ that also belongs to L_2 is 0, since the shortest distance is given by

$$\text{pref-}d(w, L_2) = d_{\text{pref}}(w, w) = |w| + |w| - 2\,|w| = 0.$$

⌜⌝

This, again, emphasises the fact, that only the words belonging to the symmetric difference of the languages L_1 and L_2 affect the prefix distance of these two languages.

Example 5.2.2. For $w = b$ and $L_2 = a^*ab$, the shortest prefix distance is given by

$$\text{pref-}d(w, L_2) = d_{\text{pref}}(w, ab) = |b| + |ab| = 3.$$

The distances of w to all of the other words of L_2 is bigger than that, since there exists no common prefix between w and any word in L_2. Thus, all the symbols of both words count into the prefix distance. In this case, one of the shortest words of the language gives the smallest distance of w and L_2. ⌜⌝

The previous example shows, that the prefix distance between a word w and a language may be given by the distance of w and a shortest word of the language. This especially is the case, if there exists no word in the language having a prefix in common with w. In this case, all symbols of w have a share to the prefix distance, and either do all symbols of the words in the language. This causes the choice of one of the shortest words to minimise the prefix distance.

Example 5.2.3. Given the word $w = a^{42}ba$ and the language $L_2 = a^*bb$, their prefix distance is

$$\text{pref-}d(w, L_2) = d_{\text{pref}}(w, a^{42}bb) = \left|a^{42}ba\right| + \left|a^{42}bb\right| - 2\left|a^{42}b\right| = 2.$$

To all words of L_2 with a shorter prefix than $a^{42}b$ to w, the distance is bigger, since then not only the size of the suffix a of w counts into the prefix distance but also symbols that belong to the prefix w has in common with the word $a^{42}bb$. For words of L_2 with a shorter common prefix, they are of the form a^ibb, where $0 \leq i \leq 41$. For all these words v the prefix distance to w is

$$d_{\text{pref}}(w, v) = \left|a^{42}ba\right| + |v| - 2\left|a^i\right| = \left|a^{42-i}ba\right| + |bb| = 46 - i,$$

which is greater than or equal to 5, and, thus, bigger than the distance of w and $a^{42}bb$.

All words v having the common prefix a^{42} with w, and not being the word $a^{42}bb$ with the shortest distance to w, are of the form $a^{42+i}bb$, where $i \geq 1$. Their prefix distance to w is given by

$$d_{\text{pref}}(w, v) = \left|a^{42}ba\right| + \left|a^{42+i}bb\right| - \left|a^{42}\right| = |ba| + \left|a^ibb\right| = 4 + i,$$

which is greater than 4.

This shows, that the shortest prefix distance between w and L_2 is between w and $a^{42}bb$. ⌜⌟

Example 5.2.3 shows, that there exist word-language-combinations whose prefix distance is minimised when computing the prefix distance to the word of the language, having the longest prefix in common with w from all the words in the language, and at the same time having the shortest suffix completing this longest prefix to build a word of the language.

Example 5.2.4. Let $w = a^{42}b$ and $L_2 = a^*bb$, then their prefix distance is minimised by the word combination w and $a^{42}bb$. All of the other words of L_2 lead to bigger prefix distances, since here the whole word w is a prefix of some words of L_2. ⌜⌟

In Example 5.2.4 we have a special case for the longest common prefix. Here, this prefix is the word itself.

Example 5.2.5. Having $w = aba$ and $L_2 = \{abb^{10}a^*, b, aa\}$, their prefix distance is given by

$$\text{pref-}d(w, L_2) = d_{\text{pref}}(w, aa) = |aba| + |aa| - 2|a| = 3.$$

The distance between w and b is 4, since these words do not have a common prefix. This distance exceeds the minimal distance. This means, that the distance of w and L_2 is not minimised by the distance to one of the shortest words.

The distance $d_{\text{pref}}(w, v)$ for any word $v \in abb^{10}a^*$ is at least 11, since the longest common prefix of these words is ab. But any such word v has a suffix of length at least 10. This means also to these words the distance of w and L_2 is not minimal. ⌜⌟

Example 5.2.5 from above shows, it is also possible that the prefix distance between a word w and a language is minimised by the distance of w and a word v in the language that has not the longest possible prefix in common with w. It is also not one of the shortest words of the language, and, therefore, this is another possibility to compute such a prefix distance.

Example 5.2.6. For $w = aba$ and $L_2 = \{abb^{10}a^*, b\}$ the longest common prefix of w to any word of L_2 is ab. To the words v in $abb^{10}a^*$, the word w has a prefix distance of at least 11, since the longest common prefix of w and v is ab, and the words v have length at least 12. To the word $b \in L_2$, the prefix distance of w is

$$d_{\text{pref}}(w, b) = |aba| + |b| = 4,$$

which is smaller than the distance between w and any word in $abb^{10}a^*$. ⛶

In Example 5.2.6 it is shown, that there exist word-language-combinations, such that their prefix distance is minimised by choosing the distance between the word and one of the shortest words with no common prefix to the word of the language. This may even be the case, if there exist words in the language that have a non-empty prefix in common with the word, just like in the previous example.

5.2.5 Distinguishing Different Computations of the Prefix Distance

The previous examples show that there exist different possibilities how the prefix distance has to be computed between a single word and a language. In the following, all possible and different cases for the computation of such a prefix distance between two regular languges will be separated. We also give criteria how to differentiate between the cases and how to choose the word to which the prefix distance is minimal. After this, these criteria are used to accurately compute the parameterized prefix distance of all the words of one regular language to another based on the minimal DFA accepting them.

Fact 5.2.1. Given a word w and a regular language L, exactly one of the following cases occurs:
Case 1: $w \in L$,
Case 2: $w \notin L$, and there exists no word in L that has a prefix in common with w,
Case 3: $w \notin L$, and there exists a word in L with a common prefix to w.

Before we distinguish the cases in detail, we give a short example for possible different calculations of the prefix distance.

Example 5.2.7. Let $L = \{a\}\{b\}^{11}\{a\}^* \cup \{aa\}$ be a given language. Considering the word $w_1 = ab^{11}a^5$, the distance of w_1 to L is 0, since w_1 belongs to L. This resembles Case 1. The word $w_2 = ba^{10}$ does not have any prefix in common with any word belonging to L, which is precisely Case 2. This is why its prefix distance to L is calculated by the formula $|w_2| + |aa| = 11 + 2 = 13$.

If the prefix distance of $w_3 = ab^{11}ab^5$ and L needs to be determined, the formula $|w_3| + \left|ab^{11}a\right| - 2\left|ab^{11}a\right| = 18 + 13 - 26 = 5$ gives this distance. The distance of w_3 to any other word in L is greater than this.

For the word $w_4 = ab^{10}$, its distance to L is minimised by choosing the word ab^{11}, which gives $|w_4| + \left|ab^{11}\right| - 2\left|ab^{10}\right| = 11 + 12 - 22 = 1$. This distance is smaller than the distance of w_4 to any other word of L.

Given the word $w_5 = ab^7a$, its prefix distance to L is minimised by choosing the word ab^{11}, which results in $|w_5| + \left|ab^{11}\right| - 2\left|ab^7\right| = 9 + 12 - 16 = 5$. This distance is the smallest of all possible distances of w_5 to any word in L.

The words w_3, w_4, and w_5 are examples for Case 3.

Another possible combination are the word $w_6 = aba$ and the language L given by $\{a\}\{b\}^{11}\{a\}^* \cup \{b\}$. Then the distance of given by $|w_6| + |b| = 3 + 1 = 4$. All of the other possible distances of w_6 and L are greater than this, even though w_6 has a prefix in common with each of the other words of L. This also resembles the situation of Case 3. $\square$

As seen in the previous example, there exist several subcases of Case 3. For each of these subcases, there exists a different formla to receive the distance.

Remark 5.2.1. Given a word w and a regular language L, then one of the cases stated in Fact 5.2.1 applies for this combination.

If Case 1 applies, then the prefix distance is pref-$d(w, L) = 0$.

For Case 2, it is true that pref-$d(w, L) = |w| + |m|$, for a shortest word $m \in L$, that is, $m \in \{v \in L \mid \forall u \in L : |v| \leq |u|\}$.

For Case 3 there exist the following possibilities for the computation of the prefix distance:

Case 3a applies, if w has a common prefix with some word of L, but its distance to any of these words is greater than that to a shortest word of L having no prefix in common with w. Then, the prefix distance is determined by pref-$d(w, L) = |w| + |m|$, where $m \in L$ is a shortest word, that has no common prefix with w.

Case 3b describes the situation, if there exists a word $v \in L$, that is a proper non-empty prefix of w, and the prefix distance between w and v is smaller than the distance of w to any other word of L. Then the formula for the prefix distance is pref-$d(w, L) = |w| - |v|$, where $v \in L$ is the longest prefix of w that belongs to L. For all of the other proper prefixes of w belonging to L, their distance to w is bigger.

If there exists a word in L having w as a prefix, then Case 3c applies if all of the other distances are bigger. Then the distance is derived from the formula pref-$d(w, L) = |v| - |w|$, where the word $v \in L$ is a shortest word having prefix w.

The last possible situation is described in Case 3d. This case covers all of the other possibilities for the prefix distance of Case 3. The formula to determine this distance is given by pref-$d(w, L) = |w| + |v| - 2|p|$, for a word $v \in L$ having a non-empty prefix p in common with w. The choice of this word v is described later.

The Cases 1, 2, 3a, 3b, 3c, and 3d describe all possible situations for the determinisation of the prefix distance between a word and a regular language. In the following, criteria for the cases are defined. They determine for a given word and language uniquely, which case applies for this combination.

To be able to formulate criteria to differentiate the possible cases stated in Remark 5.2.1 for a given word w and a language L, the following definitions are necessary.

Definition 5.2.2. Let w be a word, and L a language over an alphabet Σ.

1. The set of all non-empty strict prefixes of w is denoted by
$$\text{spref}(w) = \{x \in \Sigma^+ \mid w = xu, u \neq \lambda\}.$$
2. The set
$$W_w = \{v \in L \mid v = wu, u \in \Sigma^+\}$$
consists of all the words of L having the strict prefix w.
3. The set
$$P_w = \{v \in L \mid (\text{spref}(v) \cap \text{spref}(w)) \setminus L \neq \emptyset\}$$
collects all words v of L having a non-empty, strict prefix in common with w.

The set spref(w) is used to separate Case 3b from the other cases, and also to decide if Case 3a applies.

Set W_w gathers all words that possibly need to be investigated when deciding if Case 3c or another case applies. The following lemma shows, that we can restrict the investigation to a subset of W_w.

Lemma 5.2.4. *Let w be a word and L a language. If the prefix distance of w and L is minimised by a word $v \in W_w$, then this word already belongs to the subset $W'_w = \{v \in W_w \mid |v| = \min\{|u| \mid u \in W_w\}\}$.*

Proof. Let $v = wu, v' = wu' \in W_w$ with $|u| < |u'|$. It then holds true that $|v| < |v'|$ by definition. For the prefix distances of w to these words we have
$$d_{\text{pref}}(w, v) = |u| < \left|u'\right| = d_{\text{pref}}(w, v').$$
□

Notice that this set W'_w is finite since there only exist finitely many suffixes u such that wu belongs to L, and for which the length is fixed to the minimum of such suffixes.

A word of the set P_w is chosen if Case 3d applies for w to compute its shortest prefix distance to L. This set can also be restricted to a subset, when searching for a word within P_w that has the shortest distance to w.

Lemma 5.2.5. *Let w be a word and L a language, and let $v \in L$ be the word to which w has the smallest prefix distance.*

If the word in L to which w has the shortest prefix distance belongs to the set P_w, this word is to be found in the subset

$$\begin{aligned} P'_w = \{v \in P_w \mid \exists p \in (\mathrm{spref}(v) \cap \mathrm{spref}(w)) \setminus L :& \\ w = pxs, v = pys', x, y \in \Sigma, x \neq y, s, s' \in \Sigma^*& \\ \textit{and } \forall v' \in P_w, v' = pzs'', z \in \Sigma, z \neq x, s'' \in \Sigma^* : |v| \leq \left|v'\right|\}.& \end{aligned}$$

Proof. Let u and v be two words within the set P_w having the same fixed prefix $p \in \mathrm{spref}(w)$, where p is the longest prefix u and v have in common with w. If these two words have different lengths, we can assume without loss of generality $|u| > |v|$.

For the prefix distance of w to these words, we have

$$\begin{array}{lrcl} & |s_u| & > & |s_v| \\ \Leftrightarrow & |u| & > & |v| \\ \Leftrightarrow & d_{\mathrm{pref}}(w,u) = |w| + |u| - 2\,|p| & > & |w| + |v| - 2\,|p| = d_{\mathrm{pref}}(w,v). \end{array}$$

This means, if w has its shortest prefix distance to L for a word belonging to $L \cap P_w$, this word also belongs to the subset

$$\begin{aligned} \{v \in P_w \mid \exists p \in (\mathrm{spref}(v) \cap \mathrm{spref}(w)) \setminus L :& \\ w = pxs, v = pys', x, y \in \Sigma, x \neq y, s, s' \in \Sigma^*& \\ \text{and } \forall v' \in P_w, v' = pzs'', z \in \Sigma, z \neq x, s'' \in \Sigma^* : \left|s'\right| \leq \left|s''\right|\}.& \end{aligned}$$

This set is equivalent to the set P'_w since minimising the suffix of a word v in this set is the same as minimising the whole length of a word in P'_w. □

Let p be a prefix of w, and let $v \in L$ be a word for which p is the longest common prefix with w. This word v belongs to P'_w if and only if its length is minimal for all words belonging to L having p as the longest prefix in common with w.

Notice that this set P'_w is finite since there exist only finitely many prefixes of w unequal to λ and w and there exist only finitely many suffixes for each of these prefixes to build a shortest word in L of the required type. From this set P'_w the word v is chosen if Case 3d applies.

For the rest of this chapter, let m denote the shortest word of L that has no non-empty prefix in common with w. If there exist more than only one such word, let m denote one of them. Formally this means,

$$m \in \{v \in L \mid |v| = \min\{|u| \mid u \in L\}, v \notin w\Sigma^*, \nexists p \in \mathrm{spref}(w) : v \in p\Sigma^*\}.$$

This word is used to compute the distance of w and L for Case 2 and Case 3a. If any of these shortest words has a common prefix with w unequal to λ, another subcase of Case 3 applies that is different from Case 3a.

To find the word in L to which w has the shortest distance, it is sufficient to compare the distances of w to any word in either of the sets W'_w, and P'_w, $\mathrm{spref}(w) \cap L$, and its distance to m. Notice that there only exist finitely many words in all of these sets, and, therefore, there only exist finitely many distances that need to be compared.

There does not exist any other possibility for the distance of w to L than the cases, we separated before. Cases 1 up to 3c all differ from each other. The Case 3d covers all of the other possibilities and is also disjunct from the other cases.

5.2.6 Criteria for the Cases

In the following, criteria will be formulated to determine precisely the cases stated in Remark 5.2.1. These criteria are inequations that need to be fulfilled for the cases.

Let w denote a word, and L a regular language over the same alphabet Σ. To find the formula for the computation of the prefix distance between w and L, it is first tested, if w belongs to L.

Proposition 5.2.8. *If w belongs to L, then Case 1 needs to be applied. If this condition holds true, the distance* $\text{pref-}d(w, L)$ *is equal to* 0.

Proof. Since the word problem for regular languages is decidable, it is possible to check this criterion. This results in the ability to decide Case 1. In this case, the distance of w and L is minimised to zero by calculating the prefix distance to the word itself. □

The next case to decide is the Case 2. The criterion for this case is given in the following proposition.

Proposition 5.2.9. *If there exists no word in L that has a non-empty prefix in common with the word w, then Case 2 needs to be applied. The distance* $\text{pref-}d(w, L)$ *then is given by $|w| + |m|$, where m denotes one of the shortest words of L.*

Proof. Let $v \in L$ be a word that is not a shortest word. If there does not exist a non-empty common prefix between v and w, their prefix distance is given by $d_{\mathrm{pref}}(w, v) = |w| + |v| - 2\,|\lambda| = |w| + |v|$. In case, there exists no word in L having a prefix in common with w that is not λ, the distance of w and L is given by $d_{\mathrm{pref}}(w, m) = |w| + |v|$ for a shortest word m of L. □

Notice that, if there exists no word in L having a non-empty prefix in common with w, this especially means, there exists no word in L beginning with the same symbol as w. Considering the minimal DFA accepting L, the first symbol of w leads into the rejecting sink state. This can be checked, and, therefore, it is decidable if Case 2 applies for a word-language-combination for a regular language.

In the following, criteria to separate the subcases of Case 3 will be given. These criteria only need to be checked, if none of the Cases 1 and 2 apply for the combination of w and L.

Proposition 5.2.10. *If neither Case 1 nor Case 2 applies, and if* $\operatorname{spref}(w) \cap L$ *is empty, and for all words* $wu \in W'_w$*, and* $pu' \in P'_w$ *the inequations*

$$|m| \leq |u| - |w|, \tag{5.1}$$

and

$$|m| \leq |u'| - |p| \tag{5.2}$$

hold true, exactly Case 3a applies for w *and* L*, and their distance* pref-$d(w, L)$ *is given by* $|w| + |m|$*, where* $w = pu''$ *for some suffix* u''*.*

Proof. First, if $\operatorname{spref}(w) \cap L$ is not empty, there exists a word $v \in \operatorname{spref}(w) \cap L$ with

$$d_{\text{pref}}(w, m) = |w| + |m| > |w| - |v| = d_{\text{pref}}(w, v), \tag{5.3}$$

since $|v| \geq 1$. Then, Case 3a cannot apply to w and L, since the distance of w and m is greater than the distance between w and one of its non-empty prefixes belonging to L.

In case there exists no proper prefix of w belonging to L, which means that $\operatorname{spref}(w) \cap L = \emptyset$, Case 3b cannot apply. Then, it is to check if there exists no word $wu \in W'_w$, such that

$$\begin{array}{rlcr} & d_{\text{pref}}(w, m) & \leq & d_{\text{pref}}(w, wu) \\ \Leftrightarrow & |w| + |m| & \leq & |w| + |wu| - 2|w| \\ \Leftrightarrow & |m| & \leq & |u| - |w|, \end{array}$$

is fulfilled. If so, then Case 3c does not apply to w and L, but one of the Cases 3a or 3d may still apply.

If there does not exist any word $pu' \in P'_w$ for which the inequations

$$\begin{array}{rlcr} & d_{\text{pref}}(w, m) & \leq & d_{\text{pref}}(w, pu') \\ \Leftrightarrow & |w| + |m| & \leq & |w| + |pu'| - 2|p| \\ \Leftrightarrow & |m| & \leq & |u'| - |p|, \end{array}$$

are true, Case 3d also does not apply.

This means, if these inequations are fulfilled for all words belonging to W'_w and P'_w, the word w has its shortest prefix distance to the word m. This is exactly Case 3a.

Now, let Case 3a apply for the combination of word w and language L. Then there cannot exist a word $v \in L$ having a non-empty prefix in common with w, such that the prefix distance of w and v is strictly smaller than the distance of w and a shortest word $m \in L$ that has no non-empty common prefix with w. This means, the Inequations 5.1 and 5.2 need to be true for all words v belonging to W'_w and P'_w, which excludes the Cases 3c and 3d. There may also not exist any strict prefix of w belonging to L, since then Inequation 5.3 contradicts the minimality of the distance between m and w. This also excludes Case 3b. Therefore, the conditions stated in the proposition are fulfilled.

If Case 3a applies for w and L, their prefix distance is computed by the formula

$$\text{pref-}d(w, L) = d_{\text{pref}}(w, m) = |w| + |m|\,, \tag{5.4}$$

since for all words $v \in L$ different from any shortest word $m \in L$, where m has no prefix in common with w except for λ, the distance $d_{\text{pref}}(w, v)$ is greater that the distance $d_{\text{pref}}(w, m)$. □

Proposition 5.2.11. *If none of the Cases 1, 2, or 3a applies for a word w and a regular language L, and if for all words $wu \in W'_w$, and $pu' \in P'_w$ the inequations*

$$|w| \quad \leq \quad |u| + |v|\,, \tag{5.5}$$

and

$$|p| \quad \leq \quad |u'| + |v|\,, \tag{5.6}$$

hold true, exactly Case 3b applies for w and L, where v denotes the longest strict prefix of w that belongs to L. The distance pref-$d(w, L)$ *is given by $|w| - |v|$.*

Proof. Let $v \in \text{spref}(w) \cap L$ denote the longest non-empty, strict prefix of w that belongs to L. If v is the word of L to which w has the shortest prefix distance, which means Case 3b applies for w and L, the following inequations need to be true for all words $wu \in W'_w$:

$$\begin{array}{rlcr} & d_{\text{pref}}(w, v) & \leq & d_{\text{pref}}(w, wu) \\ \Leftrightarrow & |w| + |v| - 2\,|v| & \leq & |w| + |wu| - 2\,|w| \\ \Leftrightarrow & |w| & \leq & |u| + |v|\,. \end{array}$$

If this inequation is fulfilled for all words in W'_w, there does not exist a word in L different to v with prefix w, to which w has its shortest distance. This especially means that Case 3c does not apply for w and L.

If the inequation from above holds true for all words of W'_w, there may still exist a word $pu' \in P'_w$ for which the following inequations are not true:

$$\begin{array}{rlcr} & d_{\text{pref}}(w, v) & \leq & d_{\text{pref}}(w, pu') \\ \Leftrightarrow & |w| + |v| - 2\,|v| & \leq & |w| + |pu'| - 2\,|p| \\ \Leftrightarrow & |p| & \leq & |u'| + |v|\,. \end{array}$$

If these inequations are fulfilled, there exists no word in P'_w to which w has a shorter distance than to v. This also excludes Case 3d, and leaves Case 3b to apply for w and L.

If Case 3b applies for w and L, then w has the shortest distance to the longest word $v \in \text{spref}(w) \cap L$. This case only applies, if this set is not empty. Then there does not exist a word in L having w as a strict prefix, to which w has a shorter distance. This excludes Case 3c, and, therefore, all words from the set W'_w. This means, Inequation 5.17 is fulfilled. There also does not exist any word having some non-empty, strict prefix in common with w, to which w has a shorter distance. This especially means, Inequation 5.18 is fulfilled.

In case, the distance between a word w and a regular language L is given by the distance $d_{\text{pref}}(w, v)$ for the longest word $v \in \text{spref}(w) \cap L$, the precise formula is given by

$$\text{pref-}d(w, L) = d_{\text{pref}}(w, v) = d_{\text{pref}}(vw_v, v) = |vw_v| - |v| = |w_v| \,, \tag{5.7}$$

where $w = vw_v$, and $|w_v| \geq 1$. This is true, since if Case 3b applies for w and L, there does not exist any other word in L, having or not having a prefix in common with w to which w has a shorter distance. This can be seen by the inequations that need to be fulfilled if Case 3b applies. □

If none of the Cases 1 and 2 applies, and in case that w has its shortest prefix distance to L not to the word m and also not to any of its proper prefixes belonging to L, at least one of the Inequations 5.1, and 5.2 and of 5.17, and 5.18 are not fulfilled.

Then, the word to which w has the shortest distance to, belongs to one of the sets W'_w and P'_w. If this word v belongs to W'_w, it is of the form $v = wu$ for some suffix $u \in \Sigma^+$. This corresponds to Case 3c, where the prefix distance of w and L is given by

$$d_{\text{pref}}(w, v) = d_{\text{pref}}(w, wu) = |w| + |wu| - 2\,|w| = |u| \,. \tag{5.8}$$

Otherwise, the word v belongs to the set P'_w. This corresponds to Case 3d. Then $w = pw_p$ and $v = pv_p$ for a proper prefix p of w and two suffixes w_p and v_p. Both of these suffixes are unequal to the empty word, and v_p does not begin with a symbol that enlarges the prefix p to another prefix of w. The prefix distance pref-$d(w, L)$ then is computed by

$$d_{\text{pref}}(w, v) = |pw_p| + |pv_p| - 2\,|p| = |w_p| + |v_p| \,. \tag{5.9}$$

Proposition 5.2.12. *If none of the Propositions 5.2.8, 5.2.9, 5.2.10, or 5.2.11 leads to an applicable case for the distance of a word w and a regular language L, and if for at least one word $wu \in W'_w$, and all words $pu' \in P'_w$ the inequation*

$$|u| - |w| \quad \leq \quad |u'| - |p| \,. \tag{5.10}$$

holds true, exactly Case 3c applies for w and L. Their distance pref-$d(w, L)$ *then is given by Formula 5.8.*

Proof. If W'_w consists of more than only one word, any of the words $wu \in W'_w$ can be chosen for the following investigations, since all possible suffixes are of the same length, and this length is the only information that has a share to the prefix distance.

If none of the Cases 1, 2, 3a, and 3b applies for w and L, and if the following inequation holds true for all words $pu' \in P'_w$, w has its shortest distance to a word from W'_w, and Case 3c applies:

$$\begin{array}{rrcr} & d_{\text{pref}}(w, wu) & \leq & d_{\text{pref}}(w, pu') \\ \Leftrightarrow & |w| + |wu| - 2\,|w| & \leq & |w| + \left|pu'\right| - 2\,|p| \\ \Leftrightarrow & |u| - |w| & \leq & \left|u'\right| - |p|\,. \end{array}$$

To show the other direction, let Case 3c be the case that is to apply for the distance of w and L. Then, the distance of w to a word $wu \in W'_w$ is smaller than (or equal to) the distance of any word of $L \setminus W'_w$. This especially means that the conditions and inequations stated in Propositions 5.2.8, 5.2.9, 5.2.10, and 5.2.11 are not fulfilled, and none of the Cases 1, 2, 3a, and 3b can be applied for the distance of w and L.

Also for any word v from L, having a prefix in common with w that is neither λ nor w itself, and v not being a proper prefix of w, the distance of v and w is at least as big as the distance between wu and w. This especially leads to Inequation 5.10, that is fulfilled. □

Of course, it is not necessary to check the Inequations 5.1, 5.17, and 5.10 if set W'_w is empty. The same holds true for the Inequations 5.2, 5.18, and 5.10 for an empty set P'_w.

Concluding this section, the last proposition gives a criterion, when to apply Case 3d for a given word w and a regular language L.

Proposition 5.2.13. *If none of the Propositions 5.2.8, 5.2.9, 5.2.10, 5.2.11, or 5.2.12 lead to an applicable case for the distance of a word w and a regular language L, exactly Case 3d applies for w and L. Their distance* pref-$d(w, L)$ *then is given by Formula 5.9.*

Proof. If none of the Cases 1, 2, 3a, 3b, and 3c gives the shortest distance of w and L, then w does not belong to L, there exist words in L having a non-empty prefix in common with w, and no non-empty, proper prefix of w belonging to L gives a shortest distance to w. Also no word with prefix w from L has a shortest distance to w, and one of the shortest words of L having no non-empty prefix in common with w minimises the distance between w and L.

This leaves all words v of L having a common prefix with w that belongs to $\text{spref}(v) \cap \text{spref}(w) \setminus L$. These words have all a longest common prefix with w. Collecting all these words v leads precisely to the set P_w. Like already shown, the search for the word within P_w to which w has its shortest distance, we can restrict to the set P'_w. Since this set is finite, only finitely many distances need to be computed to find the minimal distance. The distance is calculated by Formula 5.9 for the word v of P'_w that minimises the distance. □

Since all the sets $\text{spref}(w)$, W'_w, and P'_w are finite, all the criteria for the Cases 3a, 3b, 3c, and 3d can be tested.

5.2.7 Construction of an Automaton to Determine the Prefix Distance

Now that we have criteria to differentiate between the possibilities for the computation of the prefix distance of a word to a language, these criteria can be used to compute the parameterized prefix distance of two languages $L(\mathcal{A}_1)$ and $L(\mathcal{A}_2)$ accepted by the minimal DFA $\mathcal{A}_1 = \langle Q_1, \Sigma, \delta_1, s_{01}, F_1\rangle$ and $\mathcal{A}_2 = \langle Q_2, \Sigma, \delta_2, s_{02}, F_2\rangle$ more precisely.

As a recall, the formula for the parameterized prefix distance is given by

$$\text{pref-}D(n, L(\mathcal{A}_1), L(\mathcal{A}_2)) = \sum_{\substack{w \in L(\mathcal{A}_1),\\ 0 \le |w| \le n}} \text{pref-}d(w, L(\mathcal{A}_2)) + \sum_{\substack{w \in L(\mathcal{A}_2),\\ 0 \le |w| \le n}} \text{pref-}d(w, L(\mathcal{A}_1)),$$

where $n \geq 0$ is fixed. In the following, these sums are computed separately. The procedure is the same for each of the sums. Therefore, it is explained only for the first of the two sums, that means only the precise approach of the computation of the sum of the prefix distances of the words of $L(\mathcal{A}_1)$ to the language $L(\mathcal{A}_2)$ will be given in detail.

To be able to do this, the criteria from above need to be decided by using information provided by the automata $\mathcal{A}_1$ and $\mathcal{A}_2$. This especially includes the ability to distinguish between productive and non-productive states. Some state s of an automaton is said to be productive if there exists at least one word $w \in \Sigma^*$ such that the automaton is driven into acceptance beginning the processing of w in state s. Formally this means for a deterministic automaton $\mathcal{A}$ and its transition function δ, there exists at least one word w such that $\delta(s, w)$ is an accepting state. If there exists no such word, state s is called non-productive.

To compute the prefix distances for all the words of $L(\mathcal{A}_1)$ up to a certain length n to the language $L(\mathcal{A}_2)$, a new automaton H is constructed based on the automata $\mathcal{A}_1$ and $\mathcal{A}_2$, that stores the following information in each state:

- As in the cross product automaton every state stores one state of $\mathcal{A}_1$ and one state of $\mathcal{A}_2$. These stored states are exactly those states, in which the automata $\mathcal{A}_1$ and $\mathcal{A}_2$ end up after processing the same word as automaton H.

- For every state s of H, a number is stored. If the state of $\mathcal{A}_1$ stored in s is productive, this number indicates the length of a shortest word that needs to be processed to reach this state s. Otherwise, this number is the length of a prefix of the currently processed word. This prefix is the last prefix processable by $\mathcal{A}_1$ that does not end up in the rejecting sink state of $\mathcal{A}_1$.

  In the following we will refer to this number by calling it read or read$_s$ for a considered state s of automaton H. This number will be limited by $2\,|Q_2|$. If this number would exceed this upper bound, this is indicated by the symbol $> 2\,|Q_2|$. This means for s more than $2\,|Q_2|$ symbols have been processed by automaton H since leaving its initial state.

- Every state s of automaton H also stores a list of at most $2\,|Q_2|$ triples. Each triple consists of one state of automaton $\mathcal{A}_1$, one of automaton $\mathcal{A}_2$, and a number of symbols. Such a list may look like $((s_2, p_2, 2), (s_1, p_1, 1), (s_0, p_0, 0))$, where s_0, s_1, s_2 are states of $\mathcal{A}_1$, p_0, p_1, p_2 are states of $\mathcal{A}_2$ and the number $0, 1$ respectively 2 are the numbers of formerly processed symbols.

Starting in its initial state, automaton H visits several states while processing some word w. This processing ends up in a state s. Like already mentioned before, all of the visited states store two states, one from $\mathcal{A}_1$ and the other from $\mathcal{A}_2$. Additionally, also the number read is stored. The list of triples stores precisely this information of all states previously visited on a path of H. This information is additionally stored in the order of occurrences of the information. This especially means, we can reproduce the paths of the processing of w in the automata $\mathcal{A}_1$ and $\mathcal{A}_2$.

Since it is impossible for a finite automaton to store infinitely many information, the size of the list is limited to $2\,|Q_2|$ triples, where Q_2 is the set of states of automaton $\mathcal{A}_2$. This means, only the last $2\,|Q_2|$ states can be reproduced of the paths in $\mathcal{A}_1$ respectively $\mathcal{A}_2$.

We will refer to this list of triples by the name hist or more precise hist_s, when considering some special state s of H.

- Each state s of H also stores a marker that can be true or false. This marker indicates, if there exists a state in automaton H from which state s is reachable, and which contains an accepting state of $\mathcal{A}_2$ or not. This means, the marker is true, if there exists such a state, otherwise it is false. In the following, it is not wanted to let this predecessor accept the empty word, and, therefore, the set of such predecessors is restricted to the states unequal to the initial state of H.

  This marker will be called accept_2 in the following.

The following definition gives the formal definition of the automaton H.

Definition 5.2.3. Let $\mathcal{A}_1 = \langle Q_1, \Sigma, \delta_1, s_{01}, F_1 \rangle$ and $\mathcal{A}_2 = \langle Q_2, \Sigma, \delta_2, s_{02}, F_2 \rangle$ be two minimal DFA. Based on these two automata, we construct a new automaton $H = \langle Q_H, \Sigma, \delta_H, s_H, F_H \rangle$, where:

- $Q_H = Q_1 \times Q_2 \times \text{read} \times \text{hist} \times \text{accept}_2$ is the set of states. Every state of automaton H stores one state of $\mathcal{A}_1$ and one of $\mathcal{A}_2$. Additionally, a number $\text{read} \in \{0, 1, \ldots, 2\,|Q_2|\,, > 2\,|Q_2|\}$ is stored. If the state of $\mathcal{A}_1$ stored in the considered state s of H is productive, this number indicates the length of a shortest word that needs to be processed to reach this state s. Otherwise, this number is the length of a prefix of processed words that lead into state s. This prefix is the longest prefix that does not drive $\mathcal{A}_1$ into its rejecting sink state.

  hist is a list of at most $2\,|Q_2|$ triples belonging to the set

  $$Q_1 \times Q_2 \times \{0, 1, \ldots, 2\,|Q_2|\,, > 2\,|Q_2|\}.$$

This list is a storage of information about the at most $2\,|Q_2|$ predecessors of the considered state. This information are the states of $\mathcal{A}_1$ and $\mathcal{A}_2$ stored in the predecessors and also their numbers read. Additionally, due to the structure of the list, also the order of the predecessors is stored.

The last information stored in every state of automaton H is accept_2 having a value in $\{\text{true}, \text{false}\}$. This is a marker for the acceptance of $\mathcal{A}_2$. If a state of H stores an accepting state of $\mathcal{A}_2$, all of its successors have this marker set to true.

- $s_H = (s_{01}, s_{02}, 0, (), \text{false}) \in Q_H$ is the initial state.
- The set of accepting states is given by

$$F_H = \{(s_1, s_2, \text{read}, \text{hist}, \text{accept}_2) \in Q_H \mid s_1 \in F_1\}.$$

- $\delta_H : Q_H \times \Sigma \to Q_H$ is the transition function, given by

$$\delta_H((s_1, s_2, \text{read}_s, \text{hist}_s, \text{accept}_2), a) =$$
$$\begin{cases} (\delta_1(s_1,a), \delta_2(s_2,a), \text{read}_s + 1, \text{hist}'_s, \text{accept}_2{}') & \text{if } s_1, s_2 \text{ are productive,} \\ & \text{and } \text{read}_s < 2\,|Q_2| \\ (\delta_1(s_1,a), \delta_2(s_2,a), > 2\,|Q_2|, \text{hist}'_s, \text{accept}_2{}') & \text{if } s_1, s_2 \text{ are productive,} \\ & \text{and } \text{read}_s \in \{2\,|Q_2|, > 2\,|Q_2|\} \\ (\delta_1(s_1,a), s_2, \text{read}_s + 1, \text{hist}_s, \text{accept}_2{}') & \text{if } s_1 \text{ is productive,} \\ & \text{and } s_2 \text{ is non-productive} \\ & \text{and } \text{read}_s < 2\,|Q_2| \\ (\delta_1(s_1,a), s_2, > 2\,|Q_2|, \text{hist}_s, \text{accept}_2{}') & \text{if } s_1 \text{ is productive,} \\ & \text{and } s_2 \text{ is non-productive} \\ & \text{and } \text{read}_s \in \{2\,|Q_2|, > 2\,|Q_2|\} \\ (s_1, \delta_2(s_2,a), \text{read}_s, \text{hist}'_s, \text{accept}_2{}') & \text{if } s_1 \text{ is non-productive,} \\ & \text{and } s_2 \text{ is productive,} \\ (s_1, s_2, \text{read}_s, \text{hist}_s, \text{accept}_2) & \text{if } s_1, s_2 \text{ are non-productive} \end{cases}$$

for all states $(s_1, s_2, \text{read}_s, \text{hist}_s, \text{accept}_2) \in Q_H$ and all symbols $a \in \Sigma$, where $\text{hist}_s = ((p_1, q_1, \text{read}_1), (p_2, q_2, \text{read}_2), \ldots, (p_k, q_k, \text{read}_k))$, for $0 \le k \le 2\,|Q_2|$,

$$\text{hist}'_s = \begin{cases} ((s_1, s_2, 0)) & \text{if } k = 0, \text{ that means} \\ & \text{hist}_s = () \\ ((s_1, s_2, \text{read}_s), (p_1, q_1, \text{read}_1), & \\ \quad (p_2, q_2, \text{read}_2), \ldots, (p_k, q_k, \text{read}_k)) & \text{if } 1 \le k < 2\,|Q_2| \\ ((s_1, s_2, \text{read}_s), (p_1, q_1, \text{read}_1), & \\ \quad (p_2, q_2, \text{read}_2), \ldots, (p_{k-1}, q_{k-1}, \text{read}_{k-1})) & \text{if } k = 2\,|Q_2| \end{cases},$$

and

$$\text{accept}_2{}' = \begin{cases} \text{true}, & \text{if } s_2 \in F_2 \\ \text{accept}_2, & \text{else} \end{cases} .$$

Notice that the initial state of H is never reentered if $\mathcal{A}_1$ accepts at least one non-empty word.

The DFA $\mathcal{A}_1$ and $\mathcal{A}_2$ are finite. Therefore, also automaton H is finite by construction since its number of states is bounded by some constant depending only on the sizes of $\mathcal{A}_1$ and $\mathcal{A}_2$. Automaton H is also deterministic since the transition function of H is based on the deterministic transitions of two minimal DFA. The constructed automaton is not minimal since there may exist several states in H that store the rejecting sink state of $\mathcal{A}_1$. No accepting state of H is reachable from these states. This means these states are all equivalent. For example, for a word w that is accepted by $\mathcal{A}_2$ but not by $\mathcal{A}_1$, if $|w| \geq 2$, and if already the first symbol of w drives automaton $\mathcal{A}_1$ into its rejecting sink state, then in automaton H, there exist several states storing this rejecting sink state of $\mathcal{A}_1$ but maybe different states of $\mathcal{A}_2$.

By definition, automaton H accepts the language $L(\mathcal{A}_1)$, since H accepts if and only if $\mathcal{A}_1$ accepts.

Having all the resources from above, the prefix distances for all the words accepted by the minimal DFA $\mathcal{A}_1$ up to a fixed length $n \geq 0$ to the language accepted by the minimal DFA $\mathcal{A}_2$ can be computed.

The automaton H from above is constructed based on these two minimal automata. For each state s of H it is necessary to be checked, which of the cases described in Remark 5.2.1 applies to words of $L(\mathcal{A}_1)$ ending up in s after being processed by H.

Let $s = (s_1, s_2, \text{read}_s, \text{hist}_s, \text{accept}_2)$ denote a state of H, with a list of triples $\text{hist}_s = ((p_1, q_1, \text{read}_1), (p_2, q_2, \text{read}_2), \ldots, (p_k, q_k, \text{read}_k))$, for $0 \leq k \leq 2\,|Q_2|$, where $k = 0$ means $\text{hist}_s = ()$. The following information is stored either in the state itself, or is obtained by the automata $\mathcal{A}_1$, $\mathcal{A}_2$, or H, or by their accepted languages.

States of $\mathcal{A}_1$ and $\mathcal{A}_2$ The states of the automata $\mathcal{A}_1$ and $\mathcal{A}_2$ are known, in which these DFA would end up after processing the same symbols that lead automaton H to state s, since these states are stored in s.

A shortest word of $L(\mathcal{A}_2)$ By m, one of the shortest words of $L(\mathcal{A}_2)$ is denoted. If there exists more than one such word, m is chosen to be one of them. Its length cannot exceed $|Q_2| - 1$, since when accepting a shortest word, the automaton $\mathcal{A}_2$ visits each of its states at most once.

The number read_s The component read_s is a number between 0 and $2\,|Q_2|$, or the symbol $> 2\,|Q_2|$. By construction it is evident that this, in general, is a counter for the minimal number of symbols that need to be processed by H from its initial state to reach state s. This is true, if the state of $\mathcal{A}_1$ stored in s is producing.

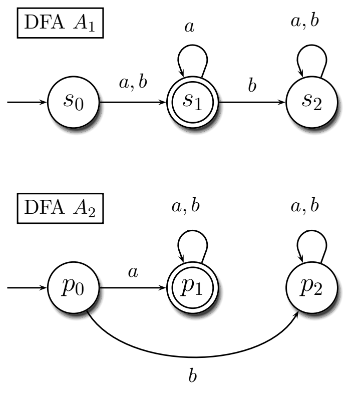


Figure 5.2: Two minimal DFA that accept the languages $L(\mathcal{A}_1) = \{a, b\}\{a\}^*$ respectively $L(\mathcal{A}_2) = \{a\}\{a, b\}^*$.

Example 5.2.8. Let H be the automaton depicted in Figures 5.3 and 5.4. By having a closer look at the connected states, one can see the growth of the number read within states storing productive states of $\mathcal{A}_1$. Each symbol leading from one such state to another changes this number by one.

One can also see, that the history for each state only changes for a successing state, if the stored state of $\mathcal{A}_2$ is productive. ⛶

In case s_1 stored in s is the rejecting sink state of $\mathcal{A}_1$, this number read_s only indicates the number of symbols that could be processed by automaton H until first entering a state also storing the rejecting sink state s_1. This means, $\text{read}_s - 1$ resembles the length of a prefix for all words processed by H leading into state s. This prefix is the longest prefix of words driving H into state s and words in $L(\mathcal{A}_1)$.

The marker accept$_2$ From the marker accept_2 stored in the state the information is obtained, if there exists a state in the automaton H from which state s is reachable, and which stores an accepting state of automaton $\mathcal{A}_2$. This means, accept_2 stores the information of the existence of a word belonging to $L(\mathcal{A}_2)$, that is a non-empty prefix of the words ending up in state s while being processed.

Example 5.2.9. We consider the part of automaton H depicted in Figure 5.3. The first occurrence of true for the marker accept_2 is in the state h_1 that stores the accepting state p_1 of $\mathcal{A}_2$. All of the states reachable from h_1 also store true for this marker. Every state not reachable from one of these states store false for the marker accept_2. This is the case in the part of automaton H depicted in Figure 5.4. ⛶

The function forward$_2$ For each state q of automaton $\mathcal{A}_2$, the length of a shortest word that leads $\mathcal{A}_2$ into acceptance can be retrieved. Formally, this number by is obtained by the function $\text{forward}_2 : Q_2 \to \{0, 1, \ldots, |Q_2| - 1, \infty\}$. For a productive

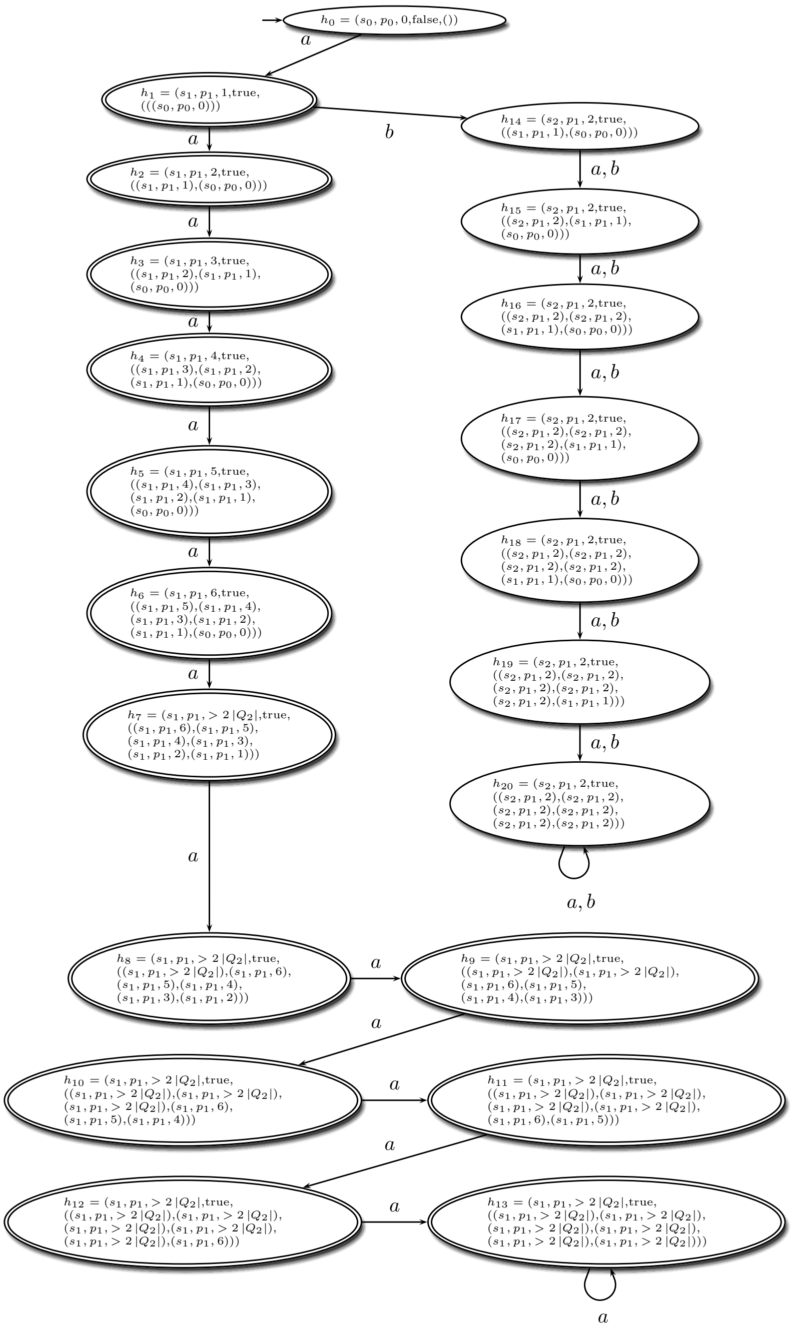
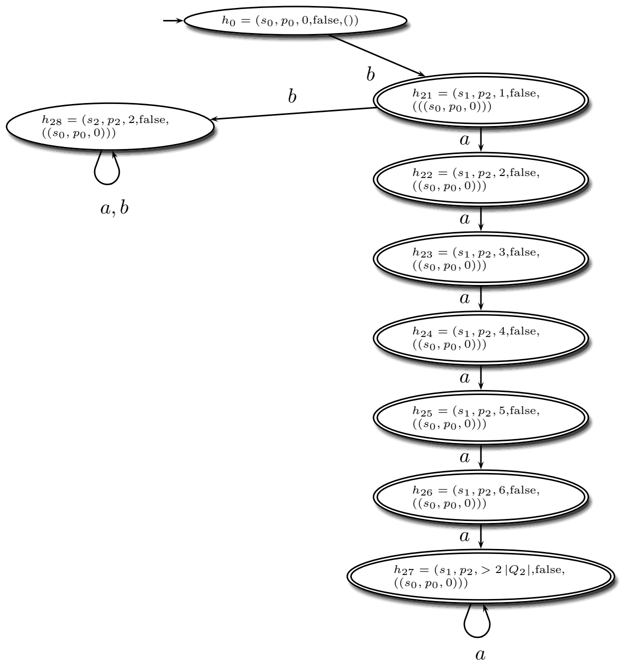

$h_0 = (s_0, p_0, 0, \text{false}, ())$
a
$h_1 = (s_1, p_1, 1, \text{true}, (((s_0, p_0, 0)))$
b
$h_{14} = (s_2, p_1, 2, \text{true}, ((s_1, p_1, 1), (s_0, p_0, 0)))$
a
$h_2 = (s_1, p_1, 2, \text{true}, ((s_1, p_1, 1), (s_0, p_0, 0)))$
a, b
$h_{15} = (s_2, p_1, 2, \text{true}, ((s_2, p_1, 2), (s_1, p_1, 1), (s_0, p_0, 0)))$
a
$h_3 = (s_1, p_1, 3, \text{true}, ((s_1, p_1, 2), (s_1, p_1, 1), (s_0, p_0, 0)))$
a, b
$h_{16} = (s_2, p_1, 2, \text{true}, ((s_2, p_1, 2), (s_2, p_1, 2), (s_1, p_1, 1), (s_0, p_0, 0)))$
a
$h_4 = (s_1, p_1, 4, \text{true}, ((s_1, p_1, 3), (s_1, p_1, 2), (s_1, p_1, 1), (s_0, p_0, 0)))$
a, b
$h_{17} = (s_2, p_1, 2, \text{true}, ((s_2, p_1, 2), (s_2, p_1, 2), (s_2, p_1, 2), (s_1, p_1, 1), (s_0, p_0, 0)))$
a
$h_5 = (s_1, p_1, 5, \text{true}, ((s_1, p_1, 4), (s_1, p_1, 3), (s_1, p_1, 2), (s_1, p_1, 1), (s_0, p_0, 0)))$
a, b
$h_{18} = (s_2, p_1, 2, \text{true}, ((s_2, p_1, 2), (s_2, p_1, 2), (s_2, p_1, 2), (s_2, p_1, 2), (s_1, p_1, 1), (s_0, p_0, 0)))$
a
$h_6 = (s_1, p_1, 6, \text{true}, ((s_1, p_1, 5), (s_1, p_1, 4), (s_1, p_1, 3), (s_1, p_1, 2), (s_1, p_1, 1), (s_0, p_0, 0)))$
a, b
$h_{19} = (s_2, p_1, 2, \text{true}, ((s_2, p_1, 2), (s_2, p_1, 2), (s_2, p_1, 2), (s_2, p_1, 2), (s_2, p_1, 2), (s_1, p_1, 1)))$
a
$h_7 = (s_1, p_1, > 2\,|Q_2|, \text{true}, ((s_1, p_1, 6), (s_1, p_1, 5), (s_1, p_1, 4), (s_1, p_1, 3), (s_1, p_1, 2), (s_1, p_1, 1)))$
a, b
$h_{20} = (s_2, p_1, 2, \text{true}, ((s_2, p_1, 2), (s_2, p_1, 2), (s_2, p_1, 2), (s_2, p_1, 2), (s_2, p_1, 2), (s_2, p_1, 2)))$
a, b
a
$h_8 = (s_1, p_1, > 2\,|Q_2|, \text{true}, ((s_1, p_1, > 2\,|Q_2|), (s_1, p_1, 6), (s_1, p_1, 5), (s_1, p_1, 4), (s_1, p_1, 3), (s_1, p_1, 2)))$
a
$h_9 = (s_1, p_1, > 2\,|Q_2|, \text{true}, ((s_1, p_1, > 2\,|Q_2|), (s_1, p_1, > 2\,|Q_2|), (s_1, p_1, 6), (s_1, p_1, 5), (s_1, p_1, 4), (s_1, p_1, 3)))$
a
$h_{10} = (s_1, p_1, > 2\,|Q_2|, \text{true}, ((s_1, p_1, > 2\,|Q_2|), (s_1, p_1, > 2\,|Q_2|), (s_1, p_1, > 2\,|Q_2|), (s_1, p_1, 6), (s_1, p_1, 5), (s_1, p_1, 4)))$
a
$h_{11} = (s_1, p_1, > 2\,|Q_2|, \text{true}, ((s_1, p_1, > 2\,|Q_2|), (s_1, p_1, > 2\,|Q_2|), (s_1, p_1, > 2\,|Q_2|), (s_1, p_1, > 2\,|Q_2|), (s_1, p_1, 6), (s_1, p_1, 5)))$
a
$h_{12} = (s_1, p_1, > 2\,|Q_2|, \text{true}, ((s_1, p_1, > 2\,|Q_2|), (s_1, p_1, > 2\,|Q_2|), (s_1, p_1, > 2\,|Q_2|), (s_1, p_1, > 2\,|Q_2|), (s_1, p_1, > 2\,|Q_2|), (s_1, p_1, 6)))$
a
$h_{13} = (s_1, p_1, > 2\,|Q_2|, \text{true}, ((s_1, p_1, > 2\,|Q_2|), (s_1, p_1, > 2\,|Q_2|), (s_1, p_1, > 2\,|Q_2|), (s_1, p_1, > 2\,|Q_2|), (s_1, p_1, > 2\,|Q_2|), (s_1, p_1, > 2\,|Q_2|)))$
a

Figure 5.3: An example for a history automaton H, based on the minimal DFA depicted in Figure 5.2. Here only some part of the automaton is depicted for words starting with a. The missing transitions and states can be added like described in the definition of H.



$h_0 = (s_0, p_0, 0, \text{false}, ())$
b
$h_{21} = (s_1, p_2, 1, \text{false}, (((s_0, p_0, 0)))$
b
$h_{28} = (s_2, p_2, 2, \text{false}, ((s_0, p_0, 0)))$
a, b
a
$h_{22} = (s_1, p_2, 2, \text{false}, ((s_0, p_0, 0)))$
a
$h_{23} = (s_1, p_2, 3, \text{false}, ((s_0, p_0, 0)))$
a
$h_{24} = (s_1, p_2, 4, \text{false}, ((s_0, p_0, 0)))$
a
$h_{25} = (s_1, p_2, 5, \text{false}, ((s_0, p_0, 0)))$
a
$h_{26} = (s_1, p_2, 6, \text{false}, ((s_0, p_0, 0)))$
a
$h_{27} = (s_1, p_2, > 2\,|Q_2|, \text{false}, ((s_0, p_0, 0)))$
a

Figure 5.4: An example for a history automaton H, based on the minimal DFA depicted in Figure 5.2. Here only some part of the automaton is depicted for words starting with b. The missing transitions and states can be added like described in the definition of H.

state $q \in Q_2$, this function assigns the length of a shortest word $w \in \Sigma^+$ to the state q, so that $\delta_2(q, w) \in F_2$. Otherwise, the function assigns ∞.

For the productive states, the assigned length is smaller than $|Q_2|$, since for any productive state there always exists an accepting state that is reached by processing less than $|Q_2|$ symbols. This function only assigns the number 0 to a state q, if this state is accepting.

The function pos For each triple (p, q, read) belonging to hist_s, it is of interest to know how many symbols have (at least) been processed since a state of H was entered, that added the considered triple to the list. Such a state needs to be a predecessor of state s.

The number of symbols can be retrieved by counting the position of the considered triple within hist_s. Formally, this is done by the function

$$\text{pos} : \underbrace{(T \times \cdots \times T)}_{\leq 2|Q_2| \text{ times}} \times T \to \{1, 2, \ldots, 2\,|Q_2|\,, \infty\},$$

where $T := Q_1 \times Q_2 \times \{0, 1, \ldots, 2\,|Q_2|\,, > 2\,|Q_2|\}$.

This function assigns a number, that denotes the position of a triple within hist_s. If there exists more than one position for the triple, pos gives the smallest, that is the leftmost position. If the triple does not exist within hist_s, the function maps to ∞.

This position is used to determine the state of DFA $\mathcal{A}_2$, in which a common prefix to a word of $L(\mathcal{A}_1)$ stops to be processed. This is used in the Cases 3b and 3d.

Example 5.2.10. Let hist_s be denoted like above. The indices are chosen such that they denote the position of the triples within this list. This means, if this list of triples does not contain a triple twice, the function pos maps each triple $(p_i, q_i, \text{read}_i)$ to its index i.

If there exist two triples $(p_i, q_i, \text{read}_i)$, and $(p_j, q_j, \text{read}_j)$, where $i \neq j$, but $p_i = p_j$, $q_i = q_j$, and $\text{read}_i = \text{read}_j$, the function pos maps this triple to the smallest index of all the occurrences of this triple. ⌜⌝

The index lc_s Let h be a fixed index of a triple belonging to hist_s, if hist_s is not empty. There always exists a word v belonging to $L(\mathcal{A}_2)$ which has a prefix in common with a word $w \in L(\mathcal{A}_1) = L(H)$ visiting state s while being processed. This prefix is of length read_h, since this number indicates the number of symbols processed by automaton H when entering the state that added the considered triple to hist_s. In fact, this especially means, this prefix can be processed by automaton $\mathcal{A}_2$ and leads into the state q_h stored in the considered triple. In the calculation of the prefix distance of these two words v and w, this prefix does not have a share since it is subtracted from both words. Therefore, even if read_h is equal to $> 2\,|Q_2|$, only the suffixes of v and w count for their distance.

The length of the suffix of v is given by $\text{forward}_2(q_h)$, which resembles the length of the shortest suffix of words in $L(\mathcal{A}_2)$ having a prefix in common with w. In constrast, the whole length of the suffix of w generally cannot be computed from information stored in state s. Only the length of some prefix of the considered suffix of w can be determined. This prefix is computed up to state s since leaving the state of H that added the triple $(q_h, p_h, \text{read}_h)$ to the list hist_s. The length of this prefix is precisely h.

For two different fixed indices h and g of triples within hist_s, where $h < g$, the sums $\text{forward}_2(q_h) + h$ and $\text{forward}_2(q_g) + g$ can be computed and compared. The smaller sum leads to a smaller prefix distance. This is, because the part of the suffix of w that is processed from state s has a share in all prefix distances that have to be compared, and the numbers $\text{forward}_2(q_h) + h$ respectively $\text{forward}_2(q_g) + g$ build the rest of the distance. If this number is minimised, then also the whole prefix distance is minimised.

Based on the function pos, it is possible to retrieve the position of a triple (p, q, read) within hist_s, for which the sum of its position and $\text{forward}_2(q)$ is minimal. This can be done by searching for the minimum of these sums $i + \text{forward}_2(q_i)$ for all triples $(p_i, q_i, \text{read}_i)$, $1 \leq i \leq k$, stored in hist_s. For the number lc_s, we restrict to those triples, where q_i is non-accepting. In this notation, the index i indicates the position of the triple within the list hist_s. This means, $pos(\text{hist}_s, (p_i, q_i, \text{read}_i)) = i$.

The position of the triple of hist_s still needs to be determined, for which the computed sum is minimal. First, all positions of triples are retrieved from hist_s, for which the sum of the above mentioned components is minimal. From all of these positions, the one with the smallest index is chosen. Let $(p_i, q_i, \text{read}_i)$ denote the triple that is stored in hist_s in precisely this chosen index. The sum $\text{forward}_2(q_i) + i$ still needs to be compared with $\text{forward}_2(q_s)$, if q_s is non-accepting. The last number resembles the suffix of a word v, that has a prefix in common with w, which stops being processed precisely in state s. Depending on which of the number is smaller, the index lc_s either is set to i or to 0.

Definition 5.2.4. Let $\text{hist}_s = ((p_1, q_1, \text{read}_1), (p_2, q_2, \text{read}_2), \ldots, (p_k, q_k, \text{read}_k))$, for $0 \leq k \leq 2\,|Q_2|$, where $k = 0$ means $\text{hist}_s = ()$, denote the list of triples stored in a state s of automaton H. The number lc_s is defined to be the smallest index of a triple of hist_s for which the sum $i + \text{forward}_2(q_i)$ is minimised for all triples $(p_i, q_i, \text{read}_i)$ belonging to hist_s, where q_i is non-accepting. In case that $\text{forward}_2(q_s)$ for the non-accepting state q_s of $\mathcal{A}_2$ stored in state s is smaller than all of the considered sums from above, the index lc_s is set to 0.

The index s of lc_s refers to the fact that this number is computed for some fixed state s of automaton H. This number may differ for another state s'.

This index will be used to separate Case 3d from the other cases.

Example 5.2.11. Let the part of the history automaton H be given that is depicted in Figure 5.3. Considering state h_3, the sums $1+\text{forward}_2(p_1)$, $2+\text{forward}_2(p_1)$, and $3 + \text{forward}_2(p_0)$ need to be computed. This leads to the set of numbers $\{1, 2, 4\}$. The minimum of these numbers is 1, and, therefore, the index lc_{h_3} is set to 1, which is the position of the triple $(s_1, p_1, 2)$ delivering the smallest sum for all triples in hist_{h_3}. ⌞⌟

Considering the sums that need to be built to determine lc_s for some state s of automaton H, they have a range between 1 and $3\,|Q_2| - 1$. The part of the sum $\text{forward}_2(p_i)$ is at least 1 and at most $|Q_2| - 1$. This means, for the triple in hist_s at position $2\,|Q_2|$ the sum ranges between $2\,|Q_2|$ and $3\,|Q_2| - 1$. Therefore, there always exists another triple in a different position with a smaller sum. This is why it is sufficient to have a length of at most $2\,|Q_2|$ for hist_s.

The index $\mathbf{la_s}$ If the state q_s of $\mathcal{A}_2$ stored in s is accepting, all words driving automaton H into state s are accepted by automaton $\mathcal{A}_2$. Then the index la_s is set to 0. In case that q_s is non-accepting, quite like for the number lc_s, the sums $\text{forward}_2(q_i) + i$ are compared for all triples $(p_i, q_i, \text{read}_i)$ of hist_s. But this time only those triples of hist_s are considered, where q_i is an accepting state of $\mathcal{A}_2$. In fact, these sums then only consist of the position i, since forward_2 assigns 0 to an accepting state. Then the word processed up to the state that added the considered triple to hist_s also stored this accepting state of $\mathcal{A}_2$. Let s' denote this state. The acceptance of q_i especially means, that all words driving automaton H into the state s' are accepted by automaton $\mathcal{A}_2$.

For a word w leading into state s this means, that there exists at least one word v in $L(\mathcal{A}_2)$ that is a prefix of w. Such a prefix stops being processed by automaton H in state s' repectively s, if q_s is accepting.

The position of the triple that is the smallest number in the set

$$\{\text{pos}((p, q, \text{read}), \text{hist}_s) \mid q \in F_2\},$$

for all triples (p, q, read) belonging to hist_s will be called la_s, if q_s is non-accepting.

By the definition of pos the smallest position is equal to ∞ if there exists no triple in hist_s for which the state from $\mathcal{A}_2$ is accepting. Otherwise it is a number between 1 and $2\,|Q_2|$.

Definition 5.2.5. Let $\text{hist}_s = ((p_1, q_1, \text{read}_1), (p_2, q_2, \text{read}_2), \ldots, (p_k, q_k, \text{read}_k))$, for $0 \leq k \leq 2\,|Q_2|$, where $k = 0$ means $\text{hist}_s = ()$, denote the list of triples stored in a state s of automaton H. If the state of $\mathcal{A}_2$ stored in state s is accepting, the number la_s is defined to be 0. Otherwise, the number la_s is defined to be the smallest index of a triple of hist_s that stores an accepting state of automaton $\mathcal{A}_2$. If there does not exist such a triple, la_s is set to ∞.

This index determines the triple inserted by the state of H in which the longest prefix of a word entering state s is processed, that belongs to the language $L(\mathcal{A}_2)$. This is used to compare Case 3b to the other cases.

Example 5.2.12. Let the part of the history automaton H be given that is depicted in Figures 5.3 and 5.4. For state h_6, the index la_{h_6} is set to 0, since the accepting state p_1 of $\mathcal{A}_2$ is stored in h_6. For state h_{22}, the index $la_{h_{22}}$ is set to ∞, since neither p_2 nor the states of $\mathcal{A}_2$ stored in the triples belonging to hist_{h_1} are accepting. ⌜⌟

5.2.8 Detailed Differentiation of the Cases Based on Automaton H

In the following it will be explained in detail how the cases stated in Remark 5.2.1 can be differentiated with the information retrieved from two minimal DFA $\mathcal{A}_1, \mathcal{A}_2$, and automaton H, which is constructed based on these two automata. By the definition of F_H given above, H accepts a word if and only if $\mathcal{A}_1$ accepts the word. This means that H also accepts the language $L(\mathcal{A}_1)$.

Let $s = (s_1, s_2, \mathrm{read}_s, \mathrm{hist}_s, \mathrm{accept}_2)$ denote a state of automaton H. The states $s_1 \in Q_1$, $s_2 \in Q_2$ belong to $\mathcal{A}_1$ respectively $\mathcal{A}_2$. The number read_s belonging to the set $\{0, 1, \dots, 2\,|Q_2|\,, > 2\,|Q_2|\}$ is the number of symbols that have at least been processed since leaving the initial state of H. For $0 \le k \le 2\,|Q_2|$, the list of triples hist_s is given by $((p_1, q_1, \mathrm{read}_1), (p_2, q_2, \mathrm{read}_2), \dots, (p_k, q_k, \mathrm{read}_k))$, where $k = 0$ means $\mathrm{hist}_s = ()$. The marker accept_2 is true if there exists a state p in H from which state s is reachable, such that p stores an accepting state of $\mathcal{A}_2$, and p was already visited before entering state s for all words processed up to state s.

The first thing to do, is to identify those states of automaton H, for which one of the cases given in Remark 5.2.1 need to be assigned.

Lemma 5.2.6. *All investigations for the parameterized prefix distance of the languages accepted by the two minimal DFA $\mathcal{A}_1$ and $\mathcal{A}_2$ can be restricted to the accepting states s of H.*

Proof. The acceptance of a state s of H is equivalent to the fact that the state of $\mathcal{A}_1$ stored in the considered state is accepting. This means, a word of $L(\mathcal{A}_1)$ is accepted in s. These are precisely the words, for which we already formulated criteria in Section 5.2.6 to decide the formula for their contribution to the parameterized prefix distance of $L(\mathcal{A}_1)$ and $L(\mathcal{A}_2)$. $\square$

For the words accepted by some state s of automaton H, it is decidable which case from Remark 5.2.1 applies. The precise way of decision will be shown in the following. This especially includes to decide which formula is used to compute the contribution to the parameterized prefix distance.

Lemma 5.2.7. *Only for those accepting states of H that store an accepting state of $\mathcal{A}_2$, Case 1 applies. The contribution of words, accepted in these states is equal to zero.*

Proof. In the set of all accepting states those states can be identified, for which also the stored state of $\mathcal{A}_2$ is accepting. This means, the words accepted in such a state are also accepted by automaton $\mathcal{A}_2$. Thus, for these states Case 1 applies for the accepted words. This especially means, the contribution to the parameterized prefix distance of these words is zero.

If Case 1 applies for an accepted word w, then the word w also belongs to $L(\mathcal{A}_2)$. This especially means, that w is accepted in a state of H, that stores and accepting state of $\mathcal{A}_2$. $\square$

Lemma 5.2.8. *Only for those accepting states s of H that store a non-productive state of $\mathcal{A}_2$, and for which* hist_s *consists only of the triple* $(s_{01}, s_{02}, 0)$ *, and also for the initial state of H, if it is accepting, Case 2 applies. The contribution of a word w, accepted in such a state s is equal to $|w| + |m|$, where m is a shortest word of $L(\mathcal{A}_2)$.*

Proof. Let s denote an accepting state of automaton H, where hist_s consists only of the triple $(s_{01}, s_{02}, 0)$, and where the state of $\mathcal{A}_2$ stored in s is non-accepting, and let w denote an arbitrary word that is accepted in state s. This especially excludes the initial state, since in this state the history is empty.

From the structure of hist_s, and the non-productive state of $\mathcal{A}_2$ stored in state s it can be deduced, that already the first letter of w drives automaton $\mathcal{A}_2$ into its rejecting sink state. This especially means, there does not exist a word in $L(\mathcal{A}_2)$ having a non-empty prefix in common with the word w.

Since this is true for all the words accepted in state s, the requirements of Case 2 apply for all of them. From Remark 5.2.1 it is known, that the formula for the contribution of such a word w is given by $|w| + |m|$, where m is a shortest word of $L(\mathcal{A}_2)$ having no prefix in common with w.

If Case 2 applies for the words accepted in state s of H, then there may not exist any word in $L(\mathcal{A}_2)$ that has a prefix in common with any of these words. This especially means, that the first symbol of the words drives automaton $\mathcal{A}_2$ into its rejecting sink state. For automaton H this means, all paths leading into state s start with its initial state followed by a state storing the rejecting sink state of $\mathcal{A}_2$. This also means, all of the states on the paths except for the initial state store the list $((s_{01}, s_{02}, 0))$. Since the path includes state s, the list hist_s also consists only of this single triple and stores the rejecting sink state of $\mathcal{A}_2$.

This leaves the initial state of H to consider, if it is accepting. In this state, the history is empty. This means, there was no letter processed yet. The acceptance indicates, that the empty word λ belongs to $L(\mathcal{A}_1)$. For this word, the prefix distance to $L(\mathcal{A}_2)$ is always minimal to the shortest words of $L(\mathcal{A}_2)$. This is precisely Case 2. □

Example 5.2.13. Let the part of automaton H be given that is depicted in Figure 5.4. All of the words accepted in this part of the automaton start with the symbol b. The language accepted by $\mathcal{A}_2$ does not contain any word starting with b. These are precisely the criteria for Case 2.

Considering for example state h_{22} of H, Case 2 needs to apply. This is true, since the history of h_{22} only contains the triple inserted by the initial state. ⛶

The remaining accepting states s of automaton H, are the states that have not already been considered in Lemmas 5.2.7 or 5.2.8. All of these remaining states have the property that s_1 is always accepting, and s_2 is not accepting, but it may be productive or non-productive. For all such states, the following properties apply. The stored list hist_s does not consist of only the triple $(s_{01}, s_{02}, 0)$, so there exists at least one word in $L(\mathcal{A}_2)$ that has a non-empty common prefix to the words of $L(\mathcal{A}_1)$

accepted in state s. This especially means, that the list hist_s contains at least one triple $(s_i, p_i, \text{read}_i)$, where $\text{read}_i \neq 0$.

From Lemmas 5.2.7 and 5.2.8 it follows that the words accepted in these states it is only possible that one of the subcases of Case 3 applies. In the following, these cases will first be distinguished for those accepting states s of H where $\text{read}_s \leq 2\,|Q_2|$, and afterwards for those states s where read_s is equal to the symbol $> 2\,|Q_2|$.

Proposition 5.2.14. *Let s denote an accepting state of H for which hist_s consists of more than only the triple $(s_{01}, s_{02}, 0)$ and is not equal to the empty list $()$. In addition, let read_s be a number between 1 and $2\,|Q_2|$, and the state of $\mathcal{A}_2$ stored in s be non-accepting. Then one of the subcases of Case 3 given in Remark 5.2.1 applies for the words accepted in s. The criteria stated in Section 5.2.6 can be checked directly with information provided by $H, \mathcal{A}_1, \mathcal{A}_2$, and s, to decide the precise case.*

Proof. Having that $1 \leq \text{read}_s \leq 2\,|Q_2|$, the inequations stated in Section 5.2.6 can be checked directly to distinguish the cases.

Case 3a Checking for Case 3a, the first thing to do is to determine if there exists a common prefix of the words accepted by state s that belongs to $L(\mathcal{A}_2)$. This can be done by simply checking accept_2 for being true. In this case, subcase 3a will never apply for this state s like already mentioned in Section 5.2.6. Otherwise, Cases 3a, 3c, and 3d still need to be separated.

For this separation, the Inequations 5.1 and 5.2 need to be checked. Let w denote a word accepted in state s. For all $wu \in W'_w$, it needs to be tested if Inequation 5.1

$$|m| \leq |u| - |w|$$

holds true, for all words w accepted in state s. If wu is a word of $L(\mathcal{A}_2)$, s_2 is productive. Thus, $|w| = \text{read}_s \leq 2\,|Q_2|$, $|u| = \text{forward}_2(s_2) < |Q_2|$, and also the length $|m| = \min\{|v| \mid v \in L(\mathcal{A}_2)\} < |Q_2|$ are computable, and the inequation can be tested directly. If s_2 is non-productive, there exists not word wu in $L(\mathcal{A}_2)$ and this inequation does not need to be tested, since then Case 3c cannot apply.

For $pu \in L(\mathcal{A}_2)$, $p \neq w$, and p is not the empty word λ, Inequation 5.2 needs to be checked:

$$|m| \leq |u| - |p|\,.$$

It is $|m| < |Q_2|$ as above, $|p| = \text{read}_{lc_s} \leq 2\,|Q_2|$, and $|u| = \text{forward}_2(q_{lc_s}) < |Q_2|$, where lc_s is the index from Definition 5.2.4. This is chosen to find p and u, since in this inequation the distance of w to a shortest word of $L(\mathcal{A}_2)$ is compared to another word of $\mathcal{A}_2$ which has a non-empty prefix in common with w, that is also unequal to w itself. The index lc_s indicates the triple in hist_s for which the distance of w to any of such words of $L(\mathcal{A}_2)$ that need to be considered is minimal. This triple contains the necessary information in the mentioned components. Thus, the inequation can be tested by replacing the variables by the concrete numbers.

If both of these inequations are fulfilled, Case 3a applies for the words accepted in the considered state s, and the formula for their prefix distance is given by Formula 5.4.

If at least one of the inequations is not fulfilled, Case 3a cannot apply. Therefore, the other inequations from above need to be checked to identify the case applying for the words accepted by s. This concludes Case 3a.

Case 3b The Inequations 5.17 and 5.18 are tested to decide if Case 3b needs to be applied. For any word w accepted in state s, Inequation 5.17 is given by

$$|w| \leq |u| + |v|,$$

where v is the longest common prefix belonging to $L(\mathcal{A}_2)$ for any such word w, and $wu \in W'_w$ is also a word of language $L(\mathcal{A}_2)$. Due to this, s_2 must be productive. Thus, it holds true that $|w| = \text{read}_s \leq 2\,|Q_2|$, $|u| = \text{forward}_2(s_2) < |Q_2|$, and $|v| = \text{read}_{la_s} \leq \text{read}_s \leq 2\,|Q_2|$, where la_s is the index from Definition 5.2.5. These numbers are all computable since $\text{read}_s \leq 2\,|Q_2|$, and due to this hist_s also consists of at most $2\,|Q_2|$ triples, all storing a number read unequal to $> 2\,|Q_2|$. This means, that the inequation can be tested. If s_2 is non-productive, Case 3c cannot apply.

For the Inequation 5.18

$$|p| \leq |u| + |v|,$$

we choose v again to be the longest common prefix to the words accepted in state s, and pu to be another word in $L(\mathcal{A}_2)$ having the common prefix p to any of words $w = pu'$ accepted by s. Under the given conditions, this results in the relations $|p| = \text{read}_{lc_s} \leq \text{read}_s \leq 2\,|Q_2|$, $|u| = \text{forward}_2(q_{lc_s}) < |Q_2|$, and also the relation $|v| = \text{read}_{la_s} \leq \text{read}_s \leq 2\,|Q_2|$ is provided, where the indices lc_s and la_s are the ones defined before. These indices exactly indicate those triples in hist_s providing the necessary information for the minimal prefix distance of w to the considered words of the form v and pu. All of these numbers are computable, and, therefore, the inequation can be tested.

If both of the Inequations 5.17 and 5.18 are fulfilled, Case 3b applies for the words accepted in state s. Thus, Formula 5.7 gives the contribution of these words to the parameterized prefix distance.

Otherwise, if at least one of the inequations is not fulfilled, Case 3b does not apply. Since Case 3a is already excluded, the Cases 3c and 3d still need to be separated.

Cases 3c and 3d The separation of the remaining subcases of Case 3 is done by checking the Inequation 5.10

$$|u| - |w| \leq |u'| - |p|,$$

for words wu and pu' in $L(\mathcal{A}_2)$, where w denotes any arbitrary word accepted by state s. Since $wu \in L(\mathcal{A}_2)$, s_2 must be productive. If not, Inequation 5.10 does not need to be checked, and Case 3d applies. If s_2 is productive, we have that

$|w| = \text{read}_s \leq 2\,|Q_2|$, $|p| = \text{read}_{lc_s} \leq \text{read}_s \leq 2\,|Q_2|$, $|u| = \text{forward}_2(s_2) < |Q_2|$, and $|u'| = \text{forward}_2(q_{lc_s}) < |Q_2|$, where lc_s is the index defined before. Since these lengths are all numbers, the inequation can be tested. If it is fulfilled, Case 3c applies to the words accepted in s, and their contribution to the prefix distance is given by Formula 5.8.

Otherwise, Case 3d applies for those words, and Formula 5.9 gives their contribution. □

This leaves the investigation of those accepting states of automaton H, for which hist_s consists of more than only the triple $(s_{01}, s_{02}, 0)$, and where read_s is the symbol $> 2\,|Q_2|$. The criteria from Section 5.2.6 to distinguish between the cases stated in Remark 5.2.1 need to be calculated in a different way than for $\text{read}_s \leq 2\,|Q_2|$. But the general idea of the calculation remains the same like for $\text{read}_s \leq 2\,|Q_2|$.

Proposition 5.2.15. *Let s denote an accepting state of H for which* hist_s *consists of more than only the triple $(s_{01}, s_{02}, 0)$. In addition, let* read_s *be the symbol* $> 2\,|Q_2|$. *Then one of the subcases of Case 3 given in Remark 5.2.1 applies to the words accepted in s. The criteria stated in Section 5.2.6 can be checked with the information provided by H, $\mathcal{A}_1$, $\mathcal{A}_2$, and s, to decide the precise case.*

Proof. In this case, the exact length of the words w accepted in s is unknown since there have been processed more than $2\,|Q_2|$ symbols in automaton H. But also in this situation it is necessary to distinguish between the subcases of Case 3.

Case 3a Like in the finite case for read_s, it is possible to check if accept_2 is set to true. Then Case 3a cannot apply. If accept_2 is false, Case 3a still needs to be separated from the other two Cases 3c and 3d like above, but Case 3b cannot apply. Trying to check the Inequation 5.1

$$|m| \leq |u| - |w|$$

for any word w accepted in state s and for some word $wu \in W'_w$ if it exists, we have that $|w| = \text{read}_s > 2\,|Q_2|$, $|m| < |Q_2|$, and $|u| = \text{forward}_2(s_2) < |Q_2|$, since the word wu only belongs to $L(\mathcal{A}_2)$ if s_2 is productive. By this, the inequation can never be fulfilled, since the left side is zero or positive in every case and the right side is negative in any case. This means, Case 3a cannot apply, if s_2 is productive, and, thus, W'_w is not empty for any of the considered words w.

To separate Case 3a from Case 3d, the Inequation 5.2

$$|m| \leq |u| - |p|$$

needs to be checked for all words w accepted in s, where pu is a word of $L(\mathcal{A}_2)$ having a common prefix p to all considered words, which means $w = pu'$.

It is $|m|$ the length of a shortest word of $L(\mathcal{A}_2)$, so $|m| < |Q_2|$. The length of the prefix p is equal to $\text{read}_{lc_s} \leq \text{read}_s$, and $|u| = \text{forward}_2(q_{lc_s})$, where lc_s is the index from Definition 5.2.4, which gives the index of the triple in hist_s for which the prefix distance of w and $L(\mathcal{A}_2)$ is minimised in case that the word of $L(\mathcal{A}_2)$ is to choose from the set P'_w. If $\text{read}_{lc_s} \leq |u| < |Q_2|$, $|u| - |p|$ needs to be computed, since in this case this difference is a non-negative integer and the inequation may be fulfilled. Otherwise, the right side of the inequation is always negative, the inequation is never fulfilled, and, thus, Case 3a cannot apply.

If this inequation is fulfilled, for the words accepted in state s, and Cases 3b and 3c are already excluded, Case 3a applies for the computation of their share to the parameterized prefix distance. Otherwise, one of the other subcases of Case 3 must apply.

Case 3b If for state s Case 3a does not apply, it first is tested for Case 3b to apply. For this, the word $v \in L(\mathcal{A}_2)$ being the longest common prefix of all words w accepted in state s is considered. This means, $w = vu'$ for some non-empty suffix u'. Checking Case 3b against Case 3c, Inequation 5.17

$$|w| \leq |u| + |v|$$

is necessary to be checked for all words $wu \in W'_w \subseteq L(\mathcal{A}_2)$, if W'_w is not empty. Then we have that $|w| = \text{read}_s > 2\,|Q_2|$, $|u| = \text{forward}_2(q_s) < |Q_2|$, and $|v| = \text{read}_{la_s}$ if $la_s \leq 2\,|Q_2|$ respectively $|v|$ is equal to $> 2\,|Q_2|$ if $la_s = \infty$. Here, la_s is the index from Definition 5.2.5, that indicates the triple in hist_s that contains the information of the word of $L(\mathcal{A}_2)$ that is the longest prefix of any considered word w.

Since wu belongs to $L(\mathcal{A}_2)$, the stored state s_2 is productive. Thus, hist_s contains the last $2\,|Q_2|$ predecessors such that the first triple (and therefore no triple) of hist_s does not contain the rejecting sink state of $\mathcal{A}_2$. With this information, it is possible to compute the difference $|w| - |v|$. This difference is exactly la_s, if $la_s \neq \infty$, and it is possible to check the inequation $la_s \leq \text{forward}_2(s_2)$. In case $la_s = \infty$, it means that the longest prefix of every word accepted in s that belongs to $L(\mathcal{A}_2)$ is at least $2\,|Q_2| + 1$ symbols shorter that the accepted word. Then the distance $|w| + |wu| - 2\,|w| = |u| < |Q_2|$ is smaller than the distance $|w| + |v| - 2\,|v| > 2\,|Q_2|$.

If the inequation from above is not fulfilled, Case 3b cannot apply. Otherwise, Inequation 5.18

$$|p| \leq |u| + |v|\,,$$

needs to be tested, where pu is a word of $L(\mathcal{A}_2)$ and p is a prefix of all words w accepted by state s, which means $w = pu'$ for some non-empty suffix u'.

In this situation we have that $|p| = \text{read}_{lc_s}$, $|v| = \text{read}_{la_s}$ if $la_s \leq 2\,|Q_2|$ respectively $|v| > 2\,|Q_2|$ if $la_s = \infty$, and $|u| = \text{forward}_2(q_{lc_s}) < |Q_2|$, where lc_s and la_s are the indices from Definitions 5.2.4 and 5.2.5. These indices indicate those triples of hist_s providing the necessary information for the considered distances. The difference $|p| - |v| = \text{read}_{lc_s} - \text{read}_{la_s} = lc_s - la_s$ is exactly the number of symbols differentiating the longest common prefix (indicated by index la_s), and the prefix p of $pu \in L(\mathcal{A}_2)$ (indicated by index lc_s). If this number is greater than $|u| = \text{forward}_2(q_{lc_s}) < |Q_2|$, Case 3b cannot apply since then the prefix distance to the longest prefix v of each word accepted in s is greater than its distance to the word pu.

So only the difference $|p| - |v|$ needs to be computed and tested if this difference is smaller than or equal to $|u| < |Q_2|$. If $la_s \leq lc_s$ this especially results in $|v| \geq |p|$, which means that the difference $|p| - |v|$ is smaller than or equal to zero, and, therefore, is smaller than $|u|$. This means Case 3b applies if $la_s \leq lc_s$.

Having $la_s > lc_s$, the difference $|p| - |v| = \text{read}_{lc_s} - \text{read}_{la_s}$ needs to be computed. The difference $\text{read}_{lc_s} - \text{read}_{la_s}$ is the number of symbols read between leaving the state adding the triple $(p_{la_s}, q_{la_s}, \text{read}_{la_s})$ and entering the state adding $(p_{lc_s}, q_{lc_s}, \text{read}_{lc_s})$ to hist_s. This number can also be received by subtracting the positions of the triples in hist_s, which is $la_s - lc_s$. The indices la_s and lc_s are known, so this number is computable.

If la_s is equal to infinity, which means there exists no triple in hist_s with an accepting state of $\mathcal{A}_2$, this difference is ∞ since lc_s is a number smaller than $2\,|Q_2|$. Then Case 3b cannot apply since there exists a word $pu \in L(\mathcal{A}_2)$ to which the prefix distance of all words accepted in state s is smaller than the one to the longest common prefix v. So only the difference $la_s - lc_s$ needs to be computed if la_s is not equal to infinity. Inequation 5.18 proves that this difference needs to be smaller or equal to $|u| = \text{forward}_2(q_{lc_s})$. This can be checked since all numbers can be retrieved from the automata.

If both Inequations 5.17 and 5.18 are fulfilled, and Cases 1, 2, and 3a were already excluded, Case 3b applies for the words accepted by state s. Otherwise this is impossible, and the Cases 3c and 3d need to be compared.

Cases 3c and 3d The Inequation 5.10

$$|u| - |w| \leq \left|u'\right| - |p|,$$

where $wu \in W'_w \subseteq L(\mathcal{A}_2)$ and $pu' \in P'_w \subseteq L(\mathcal{A}_2)$, needs to be checked for all words w accepted in state s to decide which of the two remaining subcases of Case 3 applies. Since wu is a word belonging to $L(\mathcal{A}_2)$, s_2 is productive and hist_s contains $2\,|Q_2|$ triples, and no triple in hist_s contains the rejecting sink state of $\mathcal{A}_2$. In this situation, we have that $|u| = \text{forward}_2(s_2)$, $|u'| = \text{forward}_2(q_{lc_s})$, $|w| = \text{read}_s$, and $|p| = \text{read}_{lc_s}$, where lc_s is the index from Definition 5.2.4 indicating the index of the triple in hist_s providing the necessary information.

The difference $|w| - |p|$ is the number of symbols differentiating w and p. Since p is a prefix of w, this number is given by the position of the triple storing read_{lc_s} in hist_s. This is exactly the index lc_s. Thus, the inequation from above can be checked. If it is fulfilled, and all of the Cases 1, 2, 3a, and 3b were already excluded, Case 3c applies for s, otherwise Case 3d. □

5.2.9 Calculation of the Prefix Distance Based on Automaton H

Now that we can distinguish the cases given in Remark 5.2.1 for every accepting state of automaton H, it is still necessary to know how to compute the contributions to the prefix distance to $L(\mathcal{A}_2)$ of the words of $L(\mathcal{A}_1)$ accepted in such a state in detail.

We differentiate the computation by the cases. For all computations, an accepting state $s \in Q_H$ is given. For this state, it is already known how to compute the contribution of words accepted in s in general. This means, for those words, the formula for the contribution is the one that is stated in Remark 5.2.1.

In the following, copies of the automaton H are defined. These are used to compute the prefix distances for the words accepted by H.

Definition 5.2.6. For an accepting state $s = (s_1, s_2, \text{read}_s, \text{hist}_s, \text{accept}_2) \in Q_H$, a copy of automaton H is constructed, where the only accepting state is s. This automaton is defined by

$$H_s = \langle Q_H, \Sigma, \delta_H, s_H, \{s\}\rangle.$$

To be able to define the other types of automata, the set of predecessing states Pred_s of s is needed. These states can be found by checking for them in automaton H. The processing of the longest common prefix of words in $L(\mathcal{A}_2)$ to the words accepted in state s stops in such a state.

Definition 5.2.7. Let $s = (s_1, s_2, \text{read}_s, \text{hist}_s, \text{accept}_2)$ be an accepting state of automaton H defined in Section 5.2.7. In case that s_2 is productive, the set Pred_s is defined to be $\{s\}$.

For non-productive s_2, the set Pred_s collects all states s_pred of H, such that there exists a shortest path connecting s_pred and s. Let this path be denoted by $s_\text{pred} r_1 r_2 \cdots r_k s$, where $k \geq 0$, and $r_1, r_2, \ldots, r_k$ are states of H. Additionally to the condition that there needs to exist a path connecting states s_pred and s, state s_pred needs to store a productive state of $\mathcal{A}_2$ and state r_1 stores a non-productive state of $\mathcal{A}_2$. Formally, we have

$$\begin{aligned}\text{Pred}_\text{s} = \{q \in Q_H \mid \exists a \in \Sigma, v \in \Sigma^* : \delta_H(q, av) = s,\\ \delta_H(q, a) = (s_1', s_2', \text{read}_s', \text{hist}_s', \text{accept}_2{}'),\ \text{where } s_2' \text{ is non-productive,}\\ \text{and the path connecting } q \text{ and } s \text{ is as short as possible}\}.\end{aligned}$$

All of such states s_pred can be retrieved from the information given in automaton H. Due to the restriction that the considered paths are shortest paths, they consist of at most $|Q_H|$ many states, that is, they have finite length. The same restriction causes that there only exist finitely many such paths. Since automaton H consists of only finitely many states, and not all of them can fulfill the conditions, Pred_s is also finite for each accepting state s.

With the help of the set Pred_s, the second type of automaton used to compute the parameterized prefix distance is defined. This type will be referred to by the name prefix-automaton. This automaton is used to count the number of possible prefixes of accepted words of s, where the prefixes stop being processed in a state of Pred_s.

Definition 5.2.8. Let H denote the automaton constructed by Definition 5.2.3, and let s be an accepting state of H. The prefix-automaton is denoted by $H_{s_\text{pred},\text{pref}}$

for a fixed state $s_{\text{pred}} \in \text{Pred}_{\text{s}}$. This automaton is a copy of automaton H with the only accepting state s_{pred}, and the initial state s_H. Given a state $s_{\text{pred}} \in \text{Pred}_{\text{s}}$, the associated prefix-automaton is formally defined by

$$H_{s_{\text{pred}},\text{pref}} = \langle Q_H, \Sigma, \delta_H, s_H, \{s_{\text{pred}}\}\rangle.$$

The last type of automaton is also constructed based on the set Pred_{s}, and is called suffix-automaton. This type of automaton will be used to count the possible suffixes for a fixed prefix of words accepted in s. This prefix is as long as possible, and stops being processed in a state s_{pred} of Pred_{s}. To refer to this property, this type of automaton is denoted by $H_{s_{\text{pred}},\text{suff}}$, and consists of all states of H that are visited when processing such a suffix starting in s_{pred} and stopping in s.

Definition 5.2.9. Let H denote the automaton constructed by Definition 5.2.3, and let s be an accepting state of H. The suffix-automaton is denoted by $H_{s_{\text{pred}},\text{suff}}$ for a state $s_{\text{pred}} \in \text{Pred}_{\text{s}}$. This automaton is a copy of automaton H with the only accepting state s and the initial state s_{pred}. Another difference is that all transitions of s_{pred} are dismissed that lead into a state of H storing a productive state of $\mathcal{A}_2$.

Formally, this automaton is defined as follows. At first, a set of symbols $N \subset \Sigma$ is defined that collects the symbols for which the transition of state s_{pred} leads into a state storing a non-productive state of $\mathcal{A}_2$. Formally, we have

$$N = \{a \in \Sigma \mid \delta_2(s_{2,\text{pred}}, a) \text{ is non-productive}\}.$$

Here, $s_{2,\text{pred}}$ denotes the state of $\mathcal{A}_2$ that is stored in the state s_{pred}.

The suffix-automaton for state s and a state $s_{\text{pred}} \in \text{Pred}_{\text{s}}$ is defined to be

$$H_{s_{\text{pred}},\text{suff}} = \langle Q_H, \Sigma, \delta_{s_{\text{pred}},\text{suff}}, s_{\text{pred}}, \{s\}\rangle,$$

where $\delta_{s_{\text{pred}},\text{suff}}(q,a) = \delta_H(q,a)$, for all $q \in Q_H \setminus \{s_{\text{pred}}\}$ and $a \in \Sigma$, and

$$\delta_{s_{\text{pred}},\text{suff}}(s_{\text{pred}}, a) = \begin{cases} \delta_H(s_{\text{pred}}, a) & \text{if } a \in N, \\ \text{undefined} & \text{else} \end{cases}.$$

In the suffix-automaton, only paths from state s_{pred} to state s are preserved, that consist of states storing the rejecting sink state of $\mathcal{A}_2$. This especially means, that also state s stores this rejecting sink state. For all of the other states s, where s_2 is productive, the suffix-automaton accepts the empty language.

We will use this automaton for the share of words accepted in states where either Case 3b or Case 3d applies. Observe, that if $s_{\text{pred}} = s$ the suffix-automaton only accepts the empty word, since the only accepting state s is the only state storing a productive state of $\mathcal{A}_2$. All of the other states need to be successors of s and store a non-accepting state of $\mathcal{A}_2$ by definition. This means, s is not reachable from any of the other states.

Based on the definitions from above, the minimal DFA $\mathcal{A}_1$ and $\mathcal{A}_2$ accepting some regular languages, and automaton H built from these DFA like described in Definition 5.2.3, the exact share of words accepted in states of H to the parameterized prefix distance of $L(\mathcal{A}_1)$ and $L(\mathcal{A}_2)$ can be calculated.

The following proposition states the share of words accepted in states s of H, for which Case 1 applies. The share of each of these words is equal to zero, but for the sake of completeness, the correct amount of words having this share is calculated in the proof.

Proposition 5.2.16. *For the accepting states of automaton H where Case 1 applies, the share to the parameterized prefix distance of $L(\mathcal{A}_1)$ and $L(\mathcal{A}_2)$ is given by*

$$\sum_{\substack{s \in F_H, \\ \textit{Case 1 applies}}} 0.$$

Proof. The contribution to the parameterized prefix distance of $L(\mathcal{A}_1)$ and $L(\mathcal{A}_2)$ of words $w \in L(\mathcal{A}_1)$, that are accepted by states, for which the prefix distance is to be computed by the formula of case one, is 0.

The number of words that end up in such a state s are counted by using the copy H_s of the automaton H from above for each of these states. The census function $\text{cens}_{H_s}(n)$ for the given length n in parameterized prefix distance gives this number. The number of words in $L(\mathcal{A}_1)$ of lengths 0 to n, for which the processing by H ends up in state s needs to be multiplied by zero, since this is the contribution of all of these words. Thus, for such a single state s, the following formula is obtained:

$$\text{cens}_{H_s}(n) \cdot 0.$$

These sums are built for all states of H, for which Case 1 applies. Summing up all these numbers obtained by the census function multiplied by zero gives the exact number of the words of $L(\mathcal{A}_1)$ up to the length n, which are accepted in a state of H, where the prefix distance is computed by the formula stated for Case 1, and have multiplied them with their contribution to the considered parameterized prefix distance. This number will be referred to by d_1, which is already known to be equal to zero. □

The result of Proposition 5.2.16 is straightforward. The sum giving the share for the words accepted by automaton H, for which Case 2 applies, is stated in the following theorem.

Proposition 5.2.17. *For all accepting states of automaton H where Case 2 applies, the share to the parameterized prefix distance of $L(\mathcal{A}_1)$ and $L(\mathcal{A}_2)$ is given by*

$$\sum_{i=0}^{n} \varrho_{H_c}(i) \cdot (i + |m|)\,.$$

Automaton H_c is a copy of H that omits all transitions of s_H that lead into states storing a productive state of $\mathcal{A}_2$. Its accepting states are precisely those states for that Case 2 applies.

Proof. The contribution to the parameterized prefix distance of a word having no common prefix to any word of $L(\mathcal{A}_2)$ is the sum of the length of the word plus the length of one of the shortest words of $L(\mathcal{A}_2)$. In automaton H there exists no transition leading back into the initial state. This is because every symbol of Σ drives $\mathcal{A}_2$ into a state different from its initial state since $L(\mathcal{A}_2)$ cannot be the empty language by definition. This transition inserts a triple into the history of the successor states of H. Thus, the history in every state of H is unequal to () except for the initial state.

Only the words of $L(\mathcal{A}_1)$ need to be considered, that begin with a symbol $a \in \Sigma$, such that $\delta_2(s_{02}, a)$ is non-productive. For these words only, Case 2 applies. Their contribution can be computed with the help of a copy H_c of automaton H. In this copy all transitions from the initial state leading into a state storing a productive state of $\mathcal{A}_2$ are omitted, and the only accepting states are the accepting states of H where Case 2 applies.

Then the density function can be used to count the number of words of a certain length i between 0 and n accepted by H_c. This number is multiplied by $i + |m|$, where m denotes a shortest word of $L(\mathcal{A}_2)$. This results in the sum

$$\sum_{i=0}^{n} \varrho_{H_c}(i) \cdot (i + |m|)\,.$$

The number given by this sum is denoted by d_2. □

This leaves the calculation of the share of all the words of $L(\mathcal{A}_1)$, that are accepted in a state of automaton H, for which one of the subcases of Case 3 is applied. These subcases will be considered in the following Propositions.

Proposition 5.2.18. *For the accepting states of automaton H where Case 3a applies, the share to the parameterized prefix distance of $L(\mathcal{A}_1)$ and $L(\mathcal{A}_2)$ is given by*

$$\sum_{\substack{s \in F_H,\\ \text{Case 3a applies}}} \sum_{i=0}^{n} \varrho_{H_s}(i) \cdot (i + |m|).$$

Proof. If for a state s the prefix distance of words of $L(\mathcal{A}_1)$ accepted in s is computed by Formula 5.4 stated for Case 3a, their distance is given by their length plus the length of a shortest word m of $L(\mathcal{A}_2)$. Thus, for such a state s it is counted, how many words of a length i between 0 and n are accepted.

The number of accepted words for state s is retrieved by using the copy H_s of automaton H. The density function ϱ_{H_s} is applied for each of the lengths $0 \le i \le n$, which gives exactly the number of such words accepted by state s. Multiplying this number $\varrho_{H_s} i$ by $i + |m|$ gives the share of these words of $L(\mathcal{A}_1)$.Summing up these numbers for all $0 \le i \le n$, the whole share of words accepted in state s is obtained. This leads to the formula

$$\sum_{i=0}^{n} \varrho_{H_s}(i) \cdot (i + |m|).$$

To receive the contribution to the parameterized prefix distance of $L(\mathcal{A}_1)$ and $L(\mathcal{A}_2)$ for all words of $L(\mathcal{A}_1)$, where Case 3a holds true, it is necessary to sum up over all the states s, where Case 3a is applied. In the following the number given by this sum is called d_{3a}. □

Proposition 5.2.19. *For the accepting states $s = (s_1, s_2, \mathrm{read}_s, \mathrm{hist}_s, \mathrm{accept}_2)$ of automaton H where Case 3b applies, the share to the parameterized prefix distance of $L(\mathcal{A}_1)$ and $L(\mathcal{A}_2)$ is given by the two sums, depending on s_2 productive or non-productive.*

$$\sum_{\substack{s \in F_H, \\ \textit{Case 3b applies}, \\ s_2 \textit{ productive}}} \sum_{s_{\mathrm{pred}} \in \mathrm{Pred}_s} \sum_{i=0}^{n} \sum_{j=0}^{n-i} \varrho_{H_{s_{\mathrm{pred}}, \mathrm{pref}}}(i) \cdot \varrho_{H_{s_{\mathrm{pred}}, \mathrm{suff}}}(j) \cdot (j + la_s)\,,$$

and

$$\sum_{\substack{s \in F_H, \\ \textit{Case 3b applies}, \\ s_2 \textit{ non-productive}}} \sum_{s_{\mathrm{pred}} \in \mathrm{Pred}_s} \sum_{i=0}^{n} \sum_{j=0}^{n-i} \varrho_{H_{s_{\mathrm{pred}}, \mathrm{pref}}}(i) \cdot \varrho_{H_{s_{\mathrm{pred}}, \mathrm{suff}}}(j) \cdot (j + la_s - 1)\,.$$

Proof. Let $s = (s_1, s_2, \mathrm{read}_s, \mathrm{hist}_s, \mathrm{accept}_2)$ denote a state of H that satisfies the conditions from above for Case 3b. First, we consider those accepting states where s_2 is a productive state of $\mathcal{A}_2$. Then the information stored in hist_s was inserted by the last k direct predecessors of state s, where k denotes the length of hist_s. Having the information of hist_s and the automaton H at hand, the paths of length k leading into state s can be retrieved. Let $r_1 r_2 \cdots r_k s$ denote such a path. The information of state r_1 is the information at position k in hist_s, the one of r_2 at position $k-1$ and so on up to the information of r_k that is stored in position 1.

The marker la_s from Definition 5.2.5 gives the position of the triple, where the prefix distance of the words accepted in s to $L(\mathcal{A}_2)$ is minimised for Case 3b. Since this position la_s coincides with a state on the considered path, we know precisely, where the longest prefix belonging to $L(\mathcal{A}_2)$ stops being processed in automaton H. For all possible such paths $r_1 r_2 \cdots r_k s$, we can find this state. Let the set $\mathrm{Pred_s}'$ collect all of these states.

We still need to count, how many words of length at most n are accepted by s. We also need to assume, that for different words accepted in s the length of the longest common prefix may vary. Therefore, we split the considered words into some prefix of a length i at most n and a suffix of length j at most $n - i$. The processing of the prefix of length i starts in s_H and needs to stop in a fixed state s_{pred} of $\mathrm{Pred_s}'$, and the processing of the suffix of length j starts in state s_{pred} and stops in state s.

To count the possible numbers of such prefixes and suffixes, we construct the automata $H_{s_{\mathrm{pred}}, \mathrm{pref}}$ respectively H'_s, which is a modification of H_s, where s_{pred} is the initial state instead of s_H. Their densities applied to i respectively j give the numbers of prefixes respectively suffixes of some fixed length. Multiplying the two densities gives the overall amount of possible words accepted in s.

The last missing component in the prefix distance for words accepted in s is the amount each of the words provides to the distance. This is precisely the length j of the suffix, which is the symbols processed since leaving a state s_{pred}.

Then the prefix distance of words accepted in all such states s if given by the sum

$$\sum_{i=0}^{n}\sum_{j=0}^{n-i} \varrho_{H_{s_{\text{pred}},\text{pref}}}(i) \cdot \varrho_{H'_s}(j) \cdot j,$$

where $s_{\text{pred}} \in \mathrm{Pred_s}'$.

Next, we consider those accepting states where s_2 is a non-productive state of $\mathcal{A}_2$ and where additionally Case 3b applies. Then hist_s does not need to store the information of the direct last k predecessors, where k again denotes the length of hist_s. But it is still possible to retrieve the paths leading into state s having the form $s' r_1 r_2 \cdots r_l s$, such that the state of $\mathcal{A}_2$ stored in s' is still productive and the one on r_1 is non-productive. This also means, all states $r_2, r_3, \ldots, r_l, s$ store a non-productive state of $\mathcal{A}_2$. Due to the definition of H, we then have

$$\mathrm{hist}_{r_1} = \mathrm{hist}_{r_2} = \cdots = \mathrm{hist}_{r_l} = \mathrm{hist}_s \,.$$

All of these states s' are collected in the set $\mathrm{Pred_s}$.

Since on the path from s' to s only automaton $\mathcal{A}_1$ still visited productive states, the longest prefix processable by automaton $\mathcal{A}_2$ stopped being processed in state s' for automaton H. Then all symbols processed on such a path from s' to s have a share to the prefix distance of the words w accepted in s to $L(\mathcal{A}_2)$.

The rest of the share is connected to the number la_s indicating the position in hist_s where the prefix distance in minimised for Case 3b. The information of the triple at this index was inserted by the last predecessor of s' or s' itself that stores an accepting state of $\mathcal{A}_2$. But due to this, the suffix and the number la_s share one processed symbol. Therefore, the precise share is $la_s - 1$.

Just like above, the counting of the amount of the words accepted in s can be done by using the automaton $H_{s',\text{pref}}$ and, this time, the copy $H_{s',\text{suff}}$ for all states $s' \in \mathrm{Pred_s}$. The combination of the densities of these two automata for the prefix lengths $0 \le i \le n$ and the suffix lengths $0 \le j \le n-i$ also gives the number of words accepted by s with precisely these prefix and suffix lengths. All in all, this leads to the sum

$$\sum_{i=0}^{n}\sum_{j=0}^{n-i} \varrho_{H_{s',\text{pref}}}(i) \cdot \varrho_{H_{s',\text{suff}}}(j) \cdot (j + la_s - 1)\,.$$

Taking a closer look at the sum for those states s where s_2 is productive, it can also be written in the form of the sum above for the case that s_2 is non-productive. The length j of the suffix in this case is always fixed to la_s. The state s_{pred} is

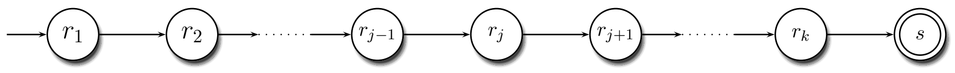


Figure 5.5: One possible path leading into a state $s = (s_1, s_2, \mathrm{read}_s, \mathrm{hist}_s, \mathrm{accept}_2)$, where s_2 is productive and hist_s consists of k triples. Then, the information in position i of hist_s was inserted by state r_{k-i+1}.

precisely the state s, wherefore the density function of automaton $H_{s_{\mathrm{pred}},\mathrm{suff}}$ applied to la_s equals 1, since the only accepting state s is not reachable from any of its successors. This means, $H_{s_{\mathrm{pred}},\mathrm{suff}}$ only accepts the empty word λ. The set $\mathrm{Pred_s}'$ is precisely the set $\mathrm{Pred_s} = \{s\}$.

This means, the sum for states $s = (s_1, s_2, \mathrm{read}_s, \mathrm{hist}_s, \mathrm{accept}_2)$, where Case 3b applies, and where s_2 is productive is given by the formula

$$\sum_{\substack{s\in F_H,\\ \text{Case 3b applies}\\ s_2 \text{ productive}}} \quad \sum_{s_{\mathrm{pred}}\in\mathrm{Pred_s}} \sum_{i=0}^{n}\sum_{j=0}^{n-i} \varrho_{H_{s_{\mathrm{pred}},\mathrm{pref}}}(i)\cdot \varrho_{H_{s_{\mathrm{pred}},\mathrm{suff}}}(j)\cdot (j + la_s)\,,$$

and for s_2 non-productive, it is

$$\sum_{\substack{s\in F_H,\\ \text{Case 3b applies}\\ s_2 \text{ non-productive}}} \quad \sum_{s_{\mathrm{pred}}\in\mathrm{Pred_s}} \sum_{i=0}^{n}\sum_{j=0}^{n-i} \varrho_{H_{s_{\mathrm{pred}},\mathrm{pref}}}(i)\cdot \varrho_{H_{s_{\mathrm{pred}},\mathrm{suff}}}(j)\cdot (j + la_s - 1)\,,$$

We will refer to the number received when adding up the two sums by d_{3b}. □

Example 5.2.14. Let $s = (s_1, s_2, \mathrm{read}_s, \mathrm{hist}_s, \mathrm{accept}_2)$ be a state of some automaton H constructed for two minimal DFA $\mathcal{A}_1$ and $\mathcal{A}_2$ like defined in Section 5.2.7. Having that Case 3b applies for s, we have to differentiate between the two possibilities s_2 is productive and s_2 is non-productive.

In the first case, there exists at least one path $r_1 r_2 \cdots r_k s$ like depicted in Figure 5.5, where k denotes the length of hist_s. Let $r_j = (s_1', s_2', \mathrm{read}_{r_j}, \mathrm{hist}_{r_j}, \mathrm{accept}_2')$ be the state on the path nearest to s, that stores an accepting state of $\mathcal{A}_2$, which means s_2' is accepting.

Then, la_s indicates precisely the triple inserted by r_j. The number of symbols processed when entering state s since leaving state r_j is precisely la_s. All words processed up to r_j are the longest prefixes of words w accepted in s that belong to $L(\mathcal{A}_2)$. This means, if for state s applies Case 3b, all of the symbols processed up to r_j do not have a share to the prefix distance of the words w to $L(\mathcal{A}_2)$. This leaves the share la_s for those words.

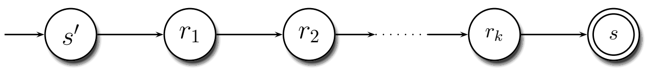


Figure 5.6: One possible path leading into a state $s = (s_1, s_2, \mathrm{read}_s, \mathrm{hist}_s, \mathrm{accept}_2)$, where s_2 is non-productive, and the only state on this path storing a productive state of $\mathcal{A}_2$ is s'. Then, the information in position 1 of hist_s was inserted by s', and the information on position $i \geq 2$ is already stored in position $i-1$ in the history of s'.

Let now s_2 be a non-productive state of $\mathcal{A}_2$. Then there exist paths $s'r_1r_2\cdots r_ks$ in H, like depicted in Figure 5.6. Here, all the states $r_1, r_2, \ldots, r_k, s$ store a non-productive state of $\mathcal{A}_2$. Only state $s' = (s_1', s_2', \mathrm{read}_{s'}, \mathrm{hist}_{s'}, \mathrm{accept}_2')$ stores a state s_2' of $\mathcal{A}_2$ that is productive. Then the information in hist_s stored in position 1 was inserted by state s' by definition of H. The rest of the information in position $i \geq 2$ is the information already stored in $\mathrm{hist}_{s'}$ in position $i-1$.

By definition, $la_{s'}$ then is $la_s - 1$. The share of words w accepted in state s consists of all symbols processed since leaving state s' up to state s. Additionally, also the symbols processed between a predecessor of s' and s' itself also count into this share. The last number is indicated by $la_{s'}$.

Summarising this information leads to precisely the share given in the formula from Proposition 5.2.19 for non-productive s_2. ⛶

Proposition 5.2.20. *For the accepting states of automaton H where Case 3c applies, the share to the parameterized prefix distance of $L(\mathcal{A}_1)$ and $L(\mathcal{A}_2)$ is given by*

$$\sum_{\substack{s \in F_H, \\ \text{Case 3c applies}}} \mathrm{cens}_{H_s}(n) \cdot \mathrm{forward}_2(s_2),$$

where $s = (s_1, s_2, \mathrm{read}_s, \mathrm{hist}_s, \mathrm{accept}_2)$.

Proof. Having the information that Case 3c holds true for words of $L(\mathcal{A}_1)$, that stop being processed in state s, their contribution is given by the number obtained by applying the function $\mathrm{forward}_2$ to the state s_2 of $\mathcal{A}_2$ stored in s. This means, it is needed to count the number of words which end up in state s of all lengths 0 to n. This is done by taking the copy H_s of H. Then, the census function cens_{H_s} for length n is applied. This gives the number of the words of $L(\mathcal{A}_1)$, that stop being processed in state s, which are of length at most n. This number is multiplied by $\mathrm{forward}_2(s_2)$ to obtain the overall contribution for this these words. The contribution for words ending up in state s is given by

$$\mathrm{cens}_{H_s}(n) \cdot \mathrm{forward}_2(s_2).$$

Summing up over all states for which Case 3c can be identified, the contribution for all words of $L(\mathcal{A}_1)$ is retrieved, for which the contribution to the parameterized prefix distance of $L(\mathcal{A}_1)$ and $L(\mathcal{A}_2)$ is computed like in Case 3c. In the following, the number given by the sum will be called d_{3c}. $\square$

Proposition 5.2.21. *For the accepting states $s = (s_1, s_2, \mathrm{read}_s, \mathrm{hist}_s, \mathrm{accept}_2)$ of automaton H where Case 3d applies, the share to the parameterized prefix distance of $L(\mathcal{A}_1)$ and $L(\mathcal{A}_2)$ is given by the following two formulas, where the cases that s_2 is productive respectively non-productive are differentiated. For s_2 productive, we have the formula*

$$\sum_{\substack{s \in F_H, \\ \textit{Case 3d applies} \\ s_2 \textit{ productive}}} \sum_{s_{\mathrm{pred}} \in \mathrm{Pred_s}} \sum_{i=0}^{n} \sum_{j=0}^{n-i} \varrho_{H_{s_{\mathrm{pred}},\mathrm{pref}}}(i) \cdot \varrho_{H_{s_{\mathrm{pred}},\mathrm{suff}}}(j) \cdot \left(j + lc_s + \mathrm{forward}_2(q_{lc_s})\right),$$

and for non-productive s_2, we have

$$\sum_{\substack{s \in F_H, \\ \textit{Case 3d applies} \\ s_2 \textit{non-productive}}} \sum_{s_{\mathrm{pred}} \in \mathrm{Pred_s}} \sum_{i=0}^{n} \sum_{j=0}^{n-i} \varrho_{H_{s_{\mathrm{pred}},\mathrm{pref}}}(i) \cdot \varrho_{H_{s_{\mathrm{pred}},\mathrm{suff}}}(j) \cdot \left(j + lc_s + \mathrm{forward}_2(q_{lc_s}) - 1\right).$$

Proof. To retrieve the share to the distance of $L(\mathcal{A}_1)$ and $L(\mathcal{A}_2)$, the procedure is quite the same as for Case 3b. Let $s = (s_1, s_2, \mathrm{read}_s, \mathrm{hist}_s, \mathrm{accept}_2)$ denote a state of H that satisfies the conditions from above for Case 3d. Again, we first consider those accepting states where s_2 is a productive state of $\mathcal{A}_2$. Then the information stored in hist_s was inserted by the last k predecessors of state s, where k denotes the length of hist_s. Having the information of hist_s and the automaton H at hand, the paths of length k leading into state s can be retrieved. Let $r_1 r_2 \cdots r_k s$ denote such a path. The information of state r_1 is the information at position k in hist_s, the one of r_2 at position $k-1$ and so on up to the information of r_k that is stored in position 1.

This time, the marker lc_s from Definition 5.2.4 gives the position of the triple, for which the prefix distance of the words accepted in s to $L(\mathcal{A}_2)$ is minimised for Case 3d. Since this position lc_s coincides with a state on the considered path, we know precisely where the common prefix of the word v belonging to $L(\mathcal{A}_2)$ and minimising the prefix distance stops being processed in automaton H. For all possible such paths $r_1 r_2 \cdots r_k s$, we can find this state. Let the set $\mathrm{Pred_s}'$ collect all of these states.

We still need to count, how many words of length at most n are accepted by s. We also need to assume, that for different words accepted in s the length of the longest common prefix may vary. Therefore, we split the considered words into some prefix of a length i that is at most n and a suffix of length $j \leq n-i$. The processing of the prefix of length i starts in s_H and needs to stop in a fixed state s_{pred} belonging to $\mathrm{Pred_s}'$. The processing of the suffix of length j starts in state s_{pred} and stops in state s.

To count the possible numbers of such prefixes and suffixes, we construct the automata $H_{s_{\mathrm{pred}},\mathrm{pref}}$ respectively H'_s, which is a modification of H_s, where s_{pred} is the initial state instead of s_H. Their densities applied to i respectively j give the numbers of prefixes respectively suffixes of some fixed length. Multiplying the two densities gives the overall amount of possible words of the fixed length accepted in s.

One last missing component in the prefix distance for words accepted in s is the amount that each of the words provides to the distance. This is precisely the length j of the suffix, which is the symbols processed since leaving a state s_{pred}. Also the share of the words in $L(\mathcal{A}_2)$ that minimise this distance is still missing. This is given by $\mathrm{forward}_2(q_{lc_s})$, like already mentioned when considering the conditions which of the cases applies for a state of H.

Then the prefix distance of words accepted in all such states s if given by the sum

$$\sum_{i=0}^{n}\sum_{j=0}^{n-i} \varrho_{H_{s_{\mathrm{pred}},\mathrm{pref}}}(i)\cdot\varrho_{H'_s}(j)\cdot(j+\mathrm{forward}_2(q_{lc_s}))\,,$$

where $s_{\mathrm{pred}} \in \mathrm{Pred_s}'$.

Next, we consider those accepting states where s_2 is a non-productive state of $\mathcal{A}_2$ and where additionally Case 3d applies. Then hist_s does not need to store the information of the direct last k predecessors, where k again denotes the length of hist_s. But it is still possible to retrieve the paths in H leading into state s having the form $s'r_1r_2\cdots r_ls$, such that the state of $\mathcal{A}_2$ stored in s' is still productive and the one on r_1 is non-productive. This also means, all states $r_2, r_3, \ldots, r_l, s$ store a non-productive state of $\mathcal{A}_2$. Due to the definition of H, we then have

$$\mathrm{hist}_{r_1} = \mathrm{hist}_{r_2} = \cdots = \mathrm{hist}_{r_l} = \mathrm{hist}_s\,.$$

All of these states s' are collected in the already defined set $\mathrm{Pred_s}$.

Since on the path from s' to s only the states of $\mathcal{A}_1$ are still productive, the longest prefix of words w accepted in s to words in $L(\mathcal{A}_2)$ has stopped being processed in state s' for automaton H. This means, all symbols processed on such a path from s' to s have a share to the prefix distance of the words w to $L(\mathcal{A}_2)$.

Part of the rest of the share is linked to the number lc_s indicating the position in hist_s where the prefix distance in minimised for Case 3b. The information of the triple at this index was inserted by the last predecessor of s' or s' itself that stores an accepting state of $\mathcal{A}_2$. The last part of the share is given by $\mathrm{forward}_2(q_{lc_s})$, where q_{lc_s} is a state of automaton $\mathcal{A}_2$. The whole share $lc_s + \mathrm{forward}_2(q_{lc_s} - 1)$ minimises the prefix distance, like already shown before. The subtracted one is due to the fact that the suffix of length j shares precisely one symbol with the part given by lc_s.

Just like above, the counting of the amount of the words accepted in s can be done by using the automaton $H_{s',\mathrm{pref}}$ and, this time, the copy $H_{s',\mathrm{suff}}$ for all states $s' \in \mathrm{Pred_s}$. The second automaton ensures that only path from s' to s are considered such that the second state on the path stores a non-productive state of $\mathcal{A}_2$. The combination of the densities of these two automata for the prefix lengths $0 \leq i \leq n$ and the suffix lengths $0 \leq j \leq n - i$ also gives the number of words accepted by s with precisely these prefix and suffix lengths. All in all, this leads to the sum

$$\sum_{i=0}^{n}\sum_{j=0}^{n-i} \varrho_{H_{s',\mathrm{pref}}}(i) \cdot \varrho_{H_{s',\mathrm{suff}}}(j) \cdot (j + lc_s + \mathrm{forward}_2(q_{lc_s} - 1)) .$$

Taking a closer look at the sum for those states s where s_2 is productive, the sum can be rewritten in the form of the sum above for s_2 non-productive. The length j of the suffix in this case is always fixed to lc_s. The state s_{pred} is precisely the state s, wherefore the density of $H_{s_{\mathrm{pred}},\mathrm{suff}}$ applied to lc_s equals 1, since the only accepting state s is not reachable from any of its successors. This means, $H_{s_{\mathrm{pred}},\mathrm{suff}}$ only accepts the empty word λ. The set $\mathrm{Pred_s}'$ is precisely the set $\mathrm{Pred_s} = \{s\}$.

This means, the sum for states $s = (s_1, s_2, \mathrm{read}_s, \mathrm{hist}_s, \mathrm{accept}_2)$ where Case 3d applies, and where s_2 is productive is given by

$$\sum_{\substack{s \in F_H,\\ \text{Case 3d applies}\\ s_2 \text{ productive}}} \sum_{s_{\mathrm{pred}} \in \mathrm{Pred_s}} \sum_{i=0}^{n}\sum_{j=0}^{n-i} \varrho_{H_{s_{\mathrm{pred}},\mathrm{pref}}}(i) \cdot \varrho_{H_{s_{\mathrm{pred}},\mathrm{suff}}}(j) \cdot \\ (j + lc_s + \mathrm{forward}_2(q_{lc_s})) ,$$

and for s_2 non-productive by

$$\sum_{\substack{s \in F_H,\\ \text{Case 3d applies}\\ s_2 \text{ non-productive}}} \sum_{s_{\mathrm{pred}} \in \mathrm{Pred_s}} \sum_{i=0}^{n}\sum_{j=0}^{n-i} \varrho_{H_{s_{\mathrm{pred}},\mathrm{pref}}}(i) \cdot \varrho_{H_{s_{\mathrm{pred}},\mathrm{suff}}}(j) \cdot \\ (j + lc_s + \mathrm{forward}_2(q_{lc_s}) - 1) .$$

The number given by adding up the two sums is called d_{3d}. □

Example 5.2.15. Let $s = (s_1, s_2, \mathrm{read}_s, \mathrm{hist}_s, \mathrm{accept}_2)$ be a state of some automaton H constructed for two minimal DFA $\mathcal{A}_1$ and $\mathcal{A}_2$ like defined in Section 5.2.7. Having that Case 3d applies for s, we have to differentiate between the two possibilities s_2 is productive and s_2 is non-productive.

In the first case, there exists at least one path $r_1 r_2 \cdots r_k s$ like depicted in Figure 5.5, where k denotes the length of hist_s. Let $r_j = (s_1', s_2', \mathrm{read}_{r_j}, \mathrm{hist}_{r_j}, \mathrm{accept}_2')$ be the state on the path nearest to s, that inserted the information of the triple indicated by lc_s. Then, $s_2' = q_{lc_s}$.

The number of symbols processed when entering state s since leaving state r_j is precisely lc_s. All words processed up to r_j are the longest prefixes of words w accepted in s and words v belonging to $L(\mathcal{A}_2)$. This means, if for state s applies

Case 3d, all of the symbols processed up to r_j do not have a share to the prefix distance of the words w to $L(\mathcal{A}_2)$. This gives a part of the share for those words that is lc_s. The second part of the share for those words w is given by $\text{forward}_2(q_{lc_s})$. This correlates to the amount given in the formula given in Proposition 5.2.21 for productive s_2.

Let now s_2 be a non-productive state of $\mathcal{A}_2$. Then there exist paths $s'r_1r_2\cdots r_ks$ in H, like depicted in Figure 5.6. Here, all the states $r_1, r_2, \ldots, r_k, s$ store a non-productive state of $\mathcal{A}_2$. Only state $s' = (s'_1, s'_2, \text{read}_{s'}, \text{hist}_{s'}, \text{accept}_2{}')$ stores a state s'_2 of $\mathcal{A}_2$ that is productive. Then the information in hist_s stored in position 1 was inserted by state s' by definition of H. The rest of the information in position $i \geq 2$ is the information already stored in $\text{hist}_{s'}$ in position $i-1$.

By definition, $lc_{s'}$ then is $lc_s - 1$. The share of words w accepted in state s consists of all symbols processed since leaving state s' up to state s. Additionally, also the symbols processed between a predecessor of s' and s' itself also count into this share. The last number is indicated by $lc_{s'}$. The last part of the share is given by $\text{forward}_2(q_{lc_s})$. This number gives the length of a shortest suffix that completes a prefix p to a word on $L(\mathcal{A}_2)$. This prefix p stops being processed in predecessing state of s' that inserted the triple in position lc_s of hist_s.

Summarising this information leads to precisely the share given in the formula from Proposition 5.2.21 for non-productive s_2. ⛶

Having done all the computations and constructions from above, the sum

$$\sum_1 = d_1 + d_2 + d_{3a} + d_{3b} + d_{3c} + d_{3d}$$

represents the sum of the prefix distances of each word of $L(\mathcal{A}_1)$ to $L(\mathcal{A}_2)$. Like mentioned before, the prefix distance for all words of $L(\mathcal{A}_2)$ to $L(\mathcal{A}_1)$ is received by quite similar computations and constructions. The only difference is that the roles of $L(\mathcal{A}_1)$ and $L(\mathcal{A}_2)$ are interchanged. Let this prefix distance be denoted by $\sum_2$. When adding up the two sums $\sum_1$ and $\sum_2$, the whole parameterized prefix distance pref-$D(n, L(\mathcal{A}_1), L(\mathcal{A}_2))$ for a certain length $n \geq 0$ is retrieved.

Theorem 5.2.7. *The parameterized prefix distance of the two regular languages accepted by the minimal DFA $\mathcal{A}_1$ and $\mathcal{A}_2$ is computed by*

$$\begin{aligned}
&\sum_{s \in S_1} 0 + \\
&\sum_{i=0}^{n} \varrho_{H_c}(i) \cdot (i + |m|) + \\
&\sum_{s \in S_{3a}} \sum_{i=0}^{n} \varrho_{H_s}(i) \cdot (i + |m|) + \\
&\sum_{\substack{s \in S_{3b} \\ s_2 \textit{ non-productive}}} \sum_{s_{\text{pred}} \in \text{Pred}_s} \sum_{i=0}^{n} \sum_{j=0}^{n-i} \varrho_{H_{s_{\text{pred}},\text{pref}}}(i) \cdot \varrho_{H_{s_{\text{pred}},\text{suff}}}(j) \cdot \\
&\qquad\qquad (j + la_s - 1) + \\
&\sum_{\substack{s \in S_{3b} \\ s_2 \textit{ productive}}} \sum_{s_{\text{pred}} \in \text{Pred}_s} \sum_{i=0}^{n} \sum_{j=0}^{n-i} \varrho_{H_{s_{\text{pred}},\text{pref}}}(i) \cdot \varrho_{H_{s_{\text{pred}},\text{suff}}}(j) \cdot (j + la_s) + \\
&\sum_{s \in S_{3c}} \text{cens}_{H_s}(n) \cdot \text{forward}_2(s_2) + \\
&\sum_{\substack{s \in S_{3d} \\ s_2 \textit{ non-productive}}} \sum_{s_{\text{pred}} \in \text{Pred}_s} \sum_{i=0}^{n} \sum_{j=0}^{n-i} \varrho_{H_{s_{\text{pred}},\text{pref}}}(i) \cdot \varrho_{H_{s_{\text{pred}},\text{suff}}}(j) \cdot \\
&\qquad\qquad (j + lc_s - 1 + \text{forward}_2(q_{lc_s})) + \\
&\sum_{\substack{s \in S_{3d} \\ s_2 \textit{ productive}}} \sum_{s_{\text{pred}} \in \text{Pred}_s} \sum_{i=0}^{n} \sum_{j=0}^{n-i} \varrho_{H_{s_{\text{pred}},\text{pref}}}(i) \cdot \varrho_{H_{s_{\text{pred}},\text{suff}}}(j) \cdot \\
&\qquad\qquad (j + lc_s + \text{forward}_2(q_{lc_s})),
\end{aligned} \tag{5.11}$$

where $s = (s_1, s_2, \text{read}_s, \text{hist}_s, \text{accept}_2)$.

Proof. The automaton H and all the necessary automata obtained from it are constructed in the way it is described in the former definitions. For this automaton the following sets are defined:

$$S_j = \{q \in Q_H \mid \text{case } j \text{ holds true for state } q\},$$

where $j \in \{1, 2, 3a, 3b, 3c, 3d\}$.

Having these sets we can compute the parameterized prefix distance of all words of $L(\mathcal{A}_1)$ to $L(\mathcal{A}_2)$ for a length $n \geq 0$ by the following formula:

$$
\begin{aligned}
&\sum_{s \in S_1} 0 \; + \\
&\sum_{i=0}^{n} \varrho_{H_c}(i) \cdot (i + |m|) \; + \\
&\sum_{s \in S_{3a}} \sum_{i=0}^{n} \varrho_{H_s}(i) \cdot (i + |m|) \; + \\
&\sum_{\substack{s \in S_{3b} \\ s_2 \text{ non-productive}}} \sum_{s_{\text{pred}} \in \text{Pred}_s} \sum_{i=0}^{n} \sum_{j=0}^{n-i} \varrho_{H_{s_{\text{pred}},\text{pref}}}(i) \cdot \varrho_{H_{s_{\text{pred}},\text{suff}}}(j) \cdot \\
&\qquad\qquad (j + la_s - 1) \; + \\
&\sum_{\substack{s \in S_{3b} \\ s_2 \text{ productive}}} \sum_{s_{\text{pred}} \in \text{Pred}_s} \sum_{i=0}^{n} \sum_{j=0}^{n-i} \varrho_{H_{s_{\text{pred}},\text{pref}}}(i) \cdot \varrho_{H_{s_{\text{pred}},\text{suff}}}(j) \cdot (j + la_s) \; + \\
&\sum_{s \in S_{3c}} \text{cens}_{H_s}(n) \cdot \text{forward}_2(s_2) \; + \\
&\sum_{\substack{s \in S_{3d} \\ s_2 \text{ non-productive}}} \sum_{s_{\text{pred}} \in \text{Pred}_s} \sum_{i=0}^{n} \sum_{j=0}^{n-i} \varrho_{H_{s_{\text{pred}},\text{pref}}}(i) \cdot \varrho_{H_{s_{\text{pred}},\text{suff}}}(j) \cdot \\
&\qquad\qquad (j + lc_s - 1 + \text{forward}_2(q_{lc_s})) \; + \\
&\sum_{\substack{s \in S_{3d} \\ s_2 \text{ productive}}} \sum_{s_{\text{pred}} \in \text{Pred}_s} \sum_{i=0}^{n} \sum_{j=0}^{n-i} \varrho_{H_{s_{\text{pred}},\text{pref}}}(i) \cdot \varrho_{H_{s_{\text{pred}},\text{suff}}}(j) \cdot \\
&\qquad\qquad (j + lc_s + \text{forward}_2(q_{lc_s})),
\end{aligned}
$$

where always $s = (s_1, s_2, \text{read}_s, \text{hist}_s, \text{accept}_2)$, m is one of the shortest words of $L(\mathcal{A}_2)$, and la_s and lc_s are the indices from Definitions 5.2.5 and 5.2.4. □

For the second sum in formula 5.11 of the parameterized prefix distance, the same definitions and computations with interchanged roles of the languages $\mathcal{A}_1$ and $\mathcal{A}_2$ need to be done.

5.2.10 Analysis of the Sum for the Prefix Distance on Base of H

Before analysing the sum from Formula 5.11 for one direction of the parameterized prefix distance pref-$D(n, L_1, L_2)$, we want to recall that we already know that the automaton H defined in Section 5.2.7 is finite. Its number of states is bounded by a constant depending on the numbers of states of the underlying minimal DFA.

With this information, the analysis of the sum given in Formula 5.11 is possible. The analysis of the sums for the individual cases in this formula can be restricted to the analysis of only parts of these sums.

Proposition 5.2.22. *The analysis of the sum given in Formula 5.11 can be restricted to the analysis of the sums*

$$\sum_{i=0}^{n} \varrho_{H_s}(i) \cdot i, \tag{5.12}$$

$$\sum_{i=0}^{n} \varrho_{H_{s_{\text{pred}},\text{pref}}}(i) \cdot \sum_{j=0}^{n-i} \varrho_{s_{\text{pred}},\text{suff}}(j) \cdot j, \tag{5.13}$$

$$\text{cens}_{H_s}(n). \tag{5.14}$$

Proof. The sums over the subsets of the set of accepting states of automaton H are all finite. This means only finitely many amounts of distances are summed up. It is well known, that the biggest amount belonging to such a sum gives the order of the whole sum. Therefore, the following investigations can be concentrated on the inner parts. This explicitly means the investigation of the following sums for states s and s_{pred}, that are specified in Formula 5.11:

$$\sum_{i=0}^{n} \varrho_{H_c}(i) \cdot (i + |m|)\,, \tag{5.15}$$

$$\sum_{i=0}^{n} \varrho_{H_s}(i) \cdot (i + |m|), \tag{5.16}$$

$$\sum_{i=0}^{n} \varrho_{H_{s_{\text{pred}},\text{pref}}}(i) \cdot \sum_{j=0}^{n-i} \varrho_{H_{s_{\text{pred}},\text{suff}}}(j) \cdot (j + la_s - 1)\,, \tag{5.17}$$

$$\sum_{i=0}^{n} \varrho_{H_{s_{\text{pred}},\text{pref}}}(i) \cdot \sum_{j=0}^{n-i} \varrho_{H_{s_{\text{pred}},\text{suff}}}(j) \cdot (j + la_s)\,, \tag{5.18}$$

$$\text{cens}_{H_s}(n) \cdot \text{forward}_2(s_2), \tag{5.19}$$

$$\sum_{i=0}^{n} \varrho_{H_{s_{\text{pred}},\text{pref}}}(i) \cdot \sum_{j=0}^{n-i} \varrho_{s_{\text{pred}},\text{suff}}(j) \cdot (j + lc_s - 1 + \text{forward}_2(q_{lc_s})), \tag{5.20}$$

$$\sum_{i=0}^{n} \varrho_{H_{s_{\text{pred}},\text{pref}}}(i) \cdot \sum_{j=0}^{n-i} \varrho_{s_{\text{pred}},\text{suff}}(j) \cdot (j + lc_s + \text{forward}_2(q_{lc_s})). \tag{5.21}$$

If all of these sums for all states s and s_{pred} are finite and independent of n, this gives a constant distance that is provided by the sum given in Formula 5.11. This can only hold true if all densities and census functions are constant which is the case if the language accepted by automaton H (which is $L(\mathcal{A}_1)$) is finite.

In case that at least one of the sums in Formulas 5.15, 5.16, 5.17, 5.18, 5.19, 5.20, or 5.21 is not constant for at least one state s, the whole sum in Formula 5.11 cannot be constant anymore. This is the case if at least a polynomial number of words contribute an amount of at least one to the parameterized prefix distance computed by the sum given in Formula 5.11.

In the following, the circumstances where the distance given by Formula 5.11 is polynomial or superpolynomial are identified. It is polynomial if all the inner parts cited in Formulas 5.15 to 5.21 are polynomials in n. In this case, only finitely many polynomials of certain degrees are summed up which is known to be a polynomial again. Its degree is determined by the maximal degree of the polynomials that are summed up.

If at least one of the sums in the formulas from above provides a superpolynomial amount to the prefix distance, the whole distance must also be superpolynomial. Thus, for every combination of states defined by the outer sums in Formula 5.11 it needs to be checked, if all amounts are at most polynomial or if at least one is superpolynomial.

Lets start with the analysis of the sums. Each of the automata $H_c, H_s, H_{s_{\text{pred}},\text{pref}}$, and $H_{s_{\text{pred}},\text{suff}}$ can be constructed since automaton H can effectively be constructed for given minimal DFA $\mathcal{A}_1$ and $\mathcal{A}_2$. For each of these automata based on H, the density can be determined for all lengths between 0 and n, and also the census function for n. The factors la_s and lc_s are constants for a fixed state s. The same holds true for the number given by forward$_2$ applied to a fixed state that is stored either in state s directly, or in hist$_s$ of this state. This is due to the fact that this number does not depend on n. The length $|m|$ of a shortest word of $L(\mathcal{A}_2)$ is a constant, too.

This leaves in all Formulas 5.15 to 5.21 only the densities, the census, and the lengths i and j to be non-constant. Due to this, these formulas can be simplified. Instead of the formulas stated before, the investigations can be restricted to the following ones:

$$\sum_{i=0}^{n} \varrho_{H_c}(i) \cdot (i + c_s)\,, \tag{5.22}$$

$$\sum_{i=0}^{n} \varrho_{H_s}(i) \cdot (i + c_s), \tag{5.23}$$

$$\sum_{i=0}^{n} \varrho_{H_{s_{\text{pred}},\text{pref}}}(i) \cdot \sum_{j=0}^{n-i} \varrho_{s_{\text{pred}},\text{suff}}(j) \cdot (j + c_s), \tag{5.24}$$

$$\text{cens}_{H_s}(n) \cdot c_s, \tag{5.25}$$

where c_s in each formula is a constant that only depends on the chosen state s for which the sum is computed.

The first and the second formula have the same structure, so the investigations can be restricted to the second one. When restricting the constant parameters in Formulas 5.17, 5.18, 5.20 and 5.21, this leads to the common Formula 5.24.

Having a closer look at Formula 5.22, the investigations can further be restricted to those parts of the sums that do not contain any of the constants. This is because if the sum is split into the two sums

$$\sum_{i=0}^{n} \varrho_{H_c}(i) \cdot i,$$

and

$$\sum_{i=0}^{n} \varrho_{H_c}(i) \cdot c_s,$$

the second sum is always smaller than the first one. So the second sum does not affect the highest degree of the polynomial if the whole sum is a polynomial in n, and it also does not affect the property for being superpolynomial. The same holds true for the sum stated in Formula 5.23. In Formula 5.25, the constant that multiplies the census function can also be omitted. Finally, in Formula 5.24 the constant c_s can also be omitted. This leaves the following formulas to be investigated, that are stated in this proposition:

$$\sum_{i=0}^{n} \varrho_{H_s}(i) \cdot i$$

$$\sum_{i=0}^{n} \varrho_{H_{s_{\text{pred}}},\text{pref}}(i) \cdot \sum_{j=0}^{n-i} \varrho_{s_{\text{pred}},\text{suff}}(j) \cdot j$$

$$\text{cens}_{H_s}(n)$$

□

For the analysis of these three formulas the estimations given in the two following Lemmas are useful.

Lemma 5.2.9. *Let $n \geq 0$ and $k, l \geq 1$. Then the following estimations are true:*

$$\sum_{i=0}^{n} \Omega(i^k) \subseteq \Omega(n^{k+1}) \tag{5.26}$$

and

$$\sum_{i=0}^{n} \Theta(i^k) \cdot \Omega((n-i)^l) \subseteq \Omega(n^{k+l+1}) \tag{5.27}$$

Proof.

$$\begin{aligned}
\sum_{i=0}^{n} \Theta(i^k) &\geq \sum_{i=0}^{n} c_{\min} i^k = c_{\min} \sum_{i=0}^{n} i^k \\
&\geq c_{\min} \left(\underbrace{\left(\frac{1}{4}n\right)^k + \left(\left(\frac{1}{4}n\right)^k + 1\right) + \cdots + \left(\frac{3}{4}n\right)^k}_{\frac{n}{2}} \right) \\
&\geq c_{\min} \left(\underbrace{\left(\frac{1}{4}n\right)^k + \cdots + \left(\frac{1}{4}n\right)^k}_{\frac{n}{2}} \right) \\
&= c_{\min} \left(\left(\frac{1}{4}n\right)^k \cdot \frac{n}{2} \right) \\
&= c_{\min} \cdot \left(\frac{1}{4}\right)^k \cdot \frac{1}{2} \cdot n^{k+1} \subseteq \Theta(n^{k+1}),
\end{aligned}$$

where $c_{\min}$ is the minimal constant from $\{c_0, c_1, \ldots, c_n\}$ such that for each length $0 \leq i \leq n$ the polynomial in the sum from above is bounded from below by $c_i \cdot i^k$.

$$\begin{aligned}
\sum_{i=0}^{n} \Theta(i^k) &\leq \sum_{i=0}^{n} c_{\max} i^k = c_{\max} \sum_{i=0}^{n} i^k \\
&\leq c_{\max} \sum_{i=0}^{n} n^k = c_{\max}(n+1)n^k \\
&= c_{\max} n^{k+1} + c_{\max} n^k \subseteq \Theta(n^{k+1}),
\end{aligned}$$

where $c_{\max}$ is the maximal constant from $\{c_0, c_1, \ldots, c_n\}$ such that for each length $0 \leq i \leq n$ the polynomial in the sum from above is bounded from above by $c_i \cdot i^k$.

From these estimation it follows that

$$\sum_{i=0}^{n} \Omega(i^k) \subseteq \Omega(n^{k+1})$$

and

$$\sum_{i=0}^{n} \Theta(i^k) \cdot \Omega((n-i)^l) \subseteq \Omega(n^{k+l+1}).$$

$\square$

Lemma 5.2.10. *Let $n \geq 0$ and $k, l \geq l$. The estimation*

$$\sum_{i=0}^{n} \Theta(i^k) \cdot \Theta((n-i)^l) \in \Theta(n^{k+l+1})$$

holds true.

Proof.

$$\begin{aligned}\sum_{i=0}^{n} \Theta(i^k) \cdot \Theta((n-i)^l) &= \sum_{i=0}^{n} \Theta(i^k) \cdot \Theta(i^l) \\ &= \sum_{i=0}^{n} \Theta(i^{k+l}) \subseteq \Theta(n^{k+l+1}),\end{aligned}$$

where $k, l \geq 0$ are the degrees of the considered polynomials. $\square$

Lets start investigating the first sum given in Formula 5.12.

Proposition 5.2.23. *For the sum stated in Formula 5.12, the following facts hold true:*

- *If and only if the language $L(\mathcal{A}_2)$ is finite, the sum is constant.*
- *If and only if at least one density ϱ_{H_s} for one state s is a polynomial but none is superpolynomial, the sum is polynomial. If the highest degree of all of the polynomial densities is l the polynomial of the sum has degree $l+2$.*
- *If and only if at least one density ϱ_{H_s} for one state s is superpolynomial, the sum is superpolynomial.*

The state s is some state of automaton H and is specified in Formula 5.11. Here, s is an accepting state of H for which Case 2 or Case 3a is applied.

Proof. If the density is constant, that means $\varrho_{H_s}(i) = c$ for all $i > n_0 \geq 0$ for a fixed size n_0, this leads to

$$\sum_{i=0}^{n} \varrho_{H_s}(i) \cdot i = \sum_{i=0}^{n} \Theta(1) \cdot i = \Theta(1) \cdot \sum_{i=0}^{n} i = \Theta(1) \cdot \frac{n(n+1)}{2},$$

which is a quadratic polynomial in n.

In case the density $\varrho_{H_s}(i)$ is a polynomial $p(i) = \sum_{k=0}^{l} c_k \cdot i^k$ for constants c_k of degree $l \geq 1$, this gives

$$\sum_{i=0}^{n} \varrho_{H_s}(i) \cdot i = \sum_{i=0}^{n} p(i) \cdot i = \sum_{i=0}^{n} \Theta(i^l) \cdot i = \sum_{i=0}^{n} \Theta(i^{l+1}) = \Theta(n^{l+2}),$$

which is a polynomial in n of degree $l+2$.

For a superpolynomial density, the sum is also superpolynomial. This gives

$$\sum_{i=0}^{n} \varrho_{H_s}(i) \cdot i = \sum_{i=0}^{n} \Omega(i^l) \cdot i = \sum_{i=0}^{n} \Omega(i^{l+1}) = \Omega(n^{l+2}),$$

for all $l \geq 0$.

This shows that for all possible densities, if the language $L(\mathcal{A}_2)$ is not finite, the sum is not constant. Due to this, the Formula 5.12 can only be constant, if $L(\mathcal{A}_2)$ is finite. Then there exist only finitely many summands for all word lengths n. For all n greater than or equal to the length of the longest words in $L(\mathcal{A}_2)$, the sum will not change anymore and is fixed to some constant. $\square$

The next sum to investigate is the sum stated in Formula 5.13. Since in this formula there exist two different densities, all possibilities need to be combined.

Proposition 5.2.24. *For the sum given in Formula 5.13, the following facts can be summarised:*

- *If and only if the language $L(\mathcal{A}_2)$ is finite, the sum is constant.*
- *If and only if at least one density $\varrho_{H_{s_{\text{pred}},\text{pref}}}$ or $\varrho_{H_{s_{\text{pred}},\text{suff}}}$ for one state s and a corresponding predecessor s_{pred} is a polynomial but none is superpolynomial, the whole sum is polynomial. If the highest degree of all of the polynomial densities $\varrho_{H_{s_{\text{pred}},\text{pref}}}$ is r and the highest degree of all the polynomial densities $\varrho_{H_{s_{\text{pred}},\text{suff}}}$ is l the polynomial of the sum has degree $r + l + 3$.*
- *If and only if at least one density $\varrho_{H_{s_{\text{pred}},\text{pref}}}$ or $\varrho_{H_{s_{\text{pred}},\text{suff}}}$ for one state s and a corresponding predecessor s_{pred} is superpolynomial, the whole sum is superpolynomial.*

The state s is some state of automaton H and is specified in Formula 5.11. Here, s is an accepting state of H for which Case 3b or Case 3d is applied. The state s_{pred} belongs to the set Pred_s.

Proof. At first, $\varrho_{H_{s_{\text{pred}},\text{pref}}}(i)$ to be constant is considered, which means that we have $\varrho_{H_{s_{\text{pred}},\text{pref}}}(i) = c_1$ for all $i > n_0 \geq 0$ for a fixed n_0. If also the density $\varrho_{H_{s_{\text{pred}},\text{suff}}}(i)$ is constant, that is $\varrho_{H_{s_{\text{pred}},\text{suff}}}(j) = c_2$ for all $j > n_1 \geq 0$ for a fixed n_1, this leads to

$$\begin{aligned}\sum_{i=0}^{n} \varrho_{H_{s_{\text{pred}},\text{pref}}}(i) \cdot \sum_{j=0}^{n-i} \varrho_{s_{\text{pred}},\text{suff}}(j) \cdot j &= \sum_{i=0}^{n} c_1 \cdot \sum_{j=0}^{n-i} c_2 \cdot j \\ =& c_1 \cdot c_2 \sum_{i=0}^{n} \sum_{j=0}^{n-i} j = c_1 \cdot c_2 \sum_{i=0}^{n} \Theta(i^2) = \Theta(n^3),\end{aligned}$$

which is a cubic polynomial in n.

In case that the density $\varrho_{H_{s_{\text{pred}},\text{suff}}}(j)$ is a polynomial $p(j) = \sum_{k=0}^{l} c_k \cdot j^k$ for constants c_k of degree $l \geq 1$, this gives

$$\begin{aligned}&\sum_{i=0}^{n} \varrho_{H_{s_{\text{pred}},\text{pref}}}(i) \cdot \sum_{j=0}^{n-i} \varrho_{s_{\text{pred}},\text{suff}}(j) \cdot j = \sum_{i=0}^{n} c_1 \cdot \sum_{j=0}^{n-i} \Theta(j^l) \cdot j \\ =&\sum_{i=0}^{n} c_1 \cdot \sum_{j=0}^{n-i} \Theta(j^{l+1}) = \sum_{i=0}^{n} c_1 \cdot \Theta(i^{l+2}) = \Theta(n^{l+3})\end{aligned}$$

which is a polynomial in n of degree $l+3$.

For a superpolynomial density $\varrho_{H_{s_{\text{pred}},\text{suff}}}(j)$, the sum is also superpolynomial. This combination leads to

$$\begin{aligned}&\sum_{i=0}^{n} \varrho_{H_{s_{\text{pred}},\text{pref}}}(i) \cdot \sum_{j=0}^{n-i} \varrho_{s_{\text{pred}},\text{suff}}(j) \cdot j = \sum_{i=0}^{n} c_1 \cdot \sum_{j=0}^{n-i} \Omega(j^l) \cdot j \\ =&\sum_{i=0}^{n} c_1 \cdot \sum_{j=0}^{n-i} \Omega(j^{l+1}) = \sum_{i=0}^{n} c_1 \cdot \Omega(j^{l+2}) = \Omega(n^{l+3})\end{aligned}$$

for all $l \geq 0$.

Considering $\varrho_{H_{s_{\text{pred}},\text{pref}}}(i)$ to be a polynomial $p(i) = \sum_{k=0}^{l} c_k \cdot i^k$ for constants c_k of degree $r \geq 1$, first let $\varrho_{H_{s_{\text{pred}},\text{suff}}}(j)$ be constant, that means $\varrho_{H_{s_{\text{pred}},\text{suff}}}(j) = c_1$ for all $i > n_0 \geq 0$ for a fixed n_0. Then

$$\begin{aligned}&\sum_{i=0}^{n} \varrho_{H_{s_{\text{pred}},\text{pref}}}(i) \cdot \sum_{j=0}^{n-i} \varrho_{s_{\text{pred}},\text{suff}}(j) \cdot j = \sum_{i=0}^{n} \Theta(i^r) \cdot \sum_{j=0}^{n-i} c_1 \cdot j \\ =&c_2 \cdot \sum_{i=0}^{n} \Theta(i^r) \sum_{j=0}^{n-i} j = c_2 \cdot \sum_{i=0}^{n} \Theta(i^r) \cdot \Theta(i^2) = c_2 \cdot \sum_{i=0}^{n} \Theta(i^{r+2}) \\ &= \Theta(n^{r+3})\end{aligned}$$

is obtained, which is a polynomial in n of degree $r+3$.

Having that $\varrho_{H_{s_{\text{pred}},\text{suff}}}(j)$ is also a polynomial $p(j) = \sum_{k=0}^{l} c_k \cdot j^k$ for constants c_k of degree $l \geq 1$, this provides

$$\begin{aligned}&\sum_{i=0}^{n} \varrho_{H_{s_{\text{pred}},\text{pref}}}(i) \cdot \sum_{j=0}^{n-i} \varrho_{s_{\text{pred}},\text{suff}}(j) \cdot j = \sum_{i=0}^{n} \Theta(i^r) \cdot \sum_{j=0}^{n-i} \Theta(j^l) \cdot j \\ =&\sum_{i=0}^{n} \Theta(i^r) \cdot \sum_{j=0}^{n-i} \Theta(j^{l+1}) = \sum_{i=0}^{n} \Theta(i^r) \cdot \Theta(i^{l+2}) = \sum_{i=0}^{n} \Theta(i^{r+l+2}) \\ =&\Theta(n^{r+l+3}),\end{aligned}$$

which is a polynomial in n of degree $r+l+3$.

The last case for polynomial density $\varrho_{H_{s_{\text{pred}},\text{pref}}}(i)$ is that $\varrho_{H_{s_{\text{pred}},\text{suff}}}(j)$ is superpolynomial. The following then holds true:

$$\begin{aligned}
&\sum_{i=0}^{n} \varrho_{H_{s_{\text{pred}},\text{pref}}}(i) \cdot \sum_{j=0}^{n-i} \varrho_{s_{\text{pred}},\text{suff}}(j) \cdot j = \sum_{i=0}^{n} \Theta(i^r) \cdot \sum_{j=0}^{n-i} \Omega(j^l) \cdot j \\
=&\sum_{i=0}^{n} \Theta(i^r) \cdot \sum_{j=0}^{n-i} \Omega(j^{l+1}) = \sum_{i=0}^{n} \Theta(i^r) \cdot \Omega(i^{l+2}) = \sum_{i=0}^{n} \Omega(i^{r+l+2}) \\
=&\Omega(n^{r+l+3})
\end{aligned}$$

which is superpolynomial in n for all $l \geq 1$.

The last case for $\varrho_{H_{s_{\text{pred}},\text{pref}}}(i)$ is the superpolynomial one. This does not need any further investigations since even if the inner sum is a constant c, the outer sum gives a superpolynomial:

$$\sum_{i=0}^{n} \Omega(i^l) \cdot c = \Omega(n^{l+1}),$$

which is superpolynomial in n for all $l \geq 1$.

This also shows that for all possible densities, if the language $L(\mathcal{A}_2)$ is not finite, the sum is not constant. Like in Theorem 5.2.23, the Formula 5.13 can only be constant, if $L(\mathcal{A}_2)$ is finite. Then there exist only finitely many summands for all word lengths n. For all n greater than or equal to the length of the longest words in $L(\mathcal{A}_2)$, the sum will not change anymore and is fixed to some constant. □

Investigating Formula 5.14, the census function directly gives the class for the sum, which means if $\text{cens}_{H_s}(n)$ is constant, the sum is constant. If it is polynomial, so is the sum, and if it is superpolynomial, the sum is also superpolynomial.

Proposition 5.2.25. *The sum given in Formula 5.14 is constant if* $\text{cens}_{H_s}(n)$ *is constant, it is polynomial, if* $\text{cens}_{H_s}(n)$ *is polynomial, and it is superpolynomial, if* $\text{cens}_{H_s}(n)$ *is superpolynomial. The state s is some state of automaton H and is specified in Formula 5.11. Here, s is an accepting state of H for which Case 3c is applied.*

All the results of this section can be summarised to the following theorem.

Theorem 5.2.8. *The sum provided in Formula 5.11 is polynomial, if at least one of the investigated sums is polynomial but none is superpolynomial. Its degree is the one of the polynomial of the highest degree that occurs in one of the summands. If at least one of the summands is superpolynomial, also the whole sum is superpolynomial. The sum can only be constant if the language accepted by automaton H is finite, that is the language accepted by the DFA $\mathcal{A}_1$ is finite.*

The next theorem follows directly from Theorem 5.2.8.

Theorem 5.2.9. *The parameterized prefix distance can only be constant, polynomial, or superpolynomial.*

5.3 The Parameterized Suffix Distance

The parameterized suffix distance suff-D_n is defined analogously to the parameterized prefix distance. Only the underlying distance for words is changed from the prefix to the suffix distance d_{suff}. Of course, this also changes the distance of a word to a language to the distance suff-d:

$$\text{suff-}D(n, L_1, L_2) := \sum_{\substack{w \in L_1, \\ 0 \leq |w| \leq n}} \text{suff-}d(w, L_2) + \sum_{\substack{w \in L_2, \\ 0 \leq |w| \leq n}} \text{suff-}d(w, L_1).$$

The following lemma is the base of all the results for the parameterized suffix language.

Lemma 5.3.1. *Let L_1 and L_2 be two languages over a common alphabet Σ. For two words $v \in L_1$ and $w \in L_2$ it holds true that*

$$d_{\text{suff}}(v, w) = d_{\text{pref}}(v^R, w^R).$$

Proof.

$$\begin{aligned} d_{\text{suff}}(v, w) &= |v| + |w| - 2\max\{|u| \mid v, w \in \Sigma^* u\} \\ &= \left|v^R\right| + \left|w^R\right| - 2\max\{|u| \mid v, w \in \Sigma^* u\}^R \\ &= \left|v^R\right| + \left|w^R\right| - 2\max\{\left|u^R\right| \mid v^R, w^R \in (\Sigma^* u)^R\} \\ &= \left|v^R\right| + \left|w^R\right| - 2\max\{\left|u^R\right| \mid v^R, w^R \in u^R \Sigma^*\} \\ &= d_{\text{pref}}(v^R, w^R). \end{aligned}$$

□

Therefore, all of the results for the parameterized suffix distance in this section are directly inherited from the parameterized prefix distance.

Proposition 5.3.1. *Let $L_1, L_2 \subseteq \Sigma^*$ be two languages so that $L_1 \subseteq L_2$.*

1. *For a word $v \in L_2 \setminus L_1$, let $L_1' = L_1 \cup \{v\}$ and $L_2' = L_2 \setminus \{v\}$. Then*

$$\begin{aligned} \text{suff-}D(n, L_1, L_2) &> \text{suff-}D(n, L_1', L_2) \text{ and} \\ \text{suff-}D(n, L_1, L_2) &> \text{suff-}D(n, L_1, L_2'). \end{aligned}$$

2. *For a word $v \in \Sigma^* \setminus L_2$, let $L_2' = L_2 \cup \{v\}$. Then*

$$\text{suff-}D(n, L_1, L_2) < \text{suff-}D(n, L_1, L_2').$$

3. *For a word $v \in L_1$, let $L_1' = L_1 \setminus \{v\}$. Then*

$$\text{suff-}D(n, L_1, L_2) < \text{suff-}D(n, L_1', L_2).$$

5.3.1 Upper and Lower Bounds

Proposition 5.3.2. *Let $L_1, L_2 \subseteq \Sigma^*$ be two non-empty languages,*

$$m = \min\{\min\{\,|w| \mid w \in L_1\,\}, \min\{\,|w| \mid w \in L_2\,\}\}, \ \textit{and}$$

$$M = \max\{\min\{\,|w| \mid w \in L_1\,\}, \min\{\,|w| \mid w \in L_2\,\}\}.$$

Then $\text{suff-}D(n, L_1, L_2) \leq \sum_{i=m}^{n} |\Sigma|^i \cdot (i + M)$.

Lemma 5.3.2. *Let L_1, L_2 be languages with*

$$m = \min\{\min\{\,|w| \mid w \in L_1\,\}, \min\{\,|w| \mid w \in L_2\,\}\} \ \textit{and}$$

$$M = \max\{\min\{\,|w| \mid w \in L_1\,\}, \min\{\,|w| \mid w \in L_2\,\}\}.$$

Then the upper bound of Proposition 5.3.2 is met only if (i) each word $w \in L_1 \cup L_2$ contributes $|w|+M$ to the prefix distance, (ii) $L_1 \cap L_2 = \emptyset$ if $m \geq 1$, and $L_1 \cap L_2 \subseteq \{\lambda\}$ if $m = 0$, and (iii) $L_1 \cup L_2 = \{\, w \in \Sigma^* \mid |w| \geq m\,\}$.

Proposition 5.3.3. *For any $M = m \geq 0$, there are binary regular languages $L_1, L_2 \subseteq \{a, b\}^*$ so that* $\text{suff-}D(n, L_1, L_2) = \sum_{i=m}^{n} |\Sigma|^i \cdot (i + M)$*, where m is the minimum and M is the maximum of the lengths of the shortest words in L_1 and L_2.*

Proposition 5.3.4. *Let $L_1 \subseteq \{a\}^*$ and $L_2 \subseteq \{a\}^*$ be two non-empty unary languages. Then* $\text{suff-}D(n, L_1, L_2) \leq \frac{n(n+1)}{2} + 1$.

Proposition 5.3.5. *There are unary regular languages $L_1, L_2 \subseteq \{a\}^*$ so that their suffix distance is* $\text{suff-}D(n, L_1, L_2) = \frac{n(n+1)}{2} + 1$.

5.3.2 Distances Below the Upper Bound

Proposition 5.3.6. *There are regular languages L_1 and L_2 even over a binary alphabet so that* $\text{suff-}D(n, L_1, L_2) \in \Theta(n2^n)$.

Proposition 5.3.7. *Let $c \geq 1$ be an integer. Then there are unary regular languages L_1 and L_2 so that* $\text{suff-}D(n, L_1, L_2) = c$*, for all $n \geq c$.*

Theorem 5.3.1. *Let $p(n) = x_k \cdot n^k + x_{k-1} \cdot n^{k-1} + \cdots + x_0$ be a polynomial of degree $k \geq 0$ with integer coefficients x_i, $0 \leq i \leq k$, and $x_k \geq 1$. Then two regular languages L_1 and L_2 over the alphabet $\{a, b\}$ can effectively be constructed so that* $\text{suff-}D(n, L_1, L_2) = p(n)$*, for all $n \geq n_0$, where n_0 is some constant.*

5.3.3 Decidability of the Order of the Distances

Theorem 5.3.2. *Let L_1 and L_2 be two regular languages. Then it is decidable whether the parameterized suffix distance* suff-$D(n, L_1, L_2)$ *is ultimately constant.*

Theorem 5.3.3. *Let L_1 and L_2 be two regular languages. Then it is decidable whether the parameterized suffix distance* suff-$D(n, L_1, L_2)$ *is exponential.*

Theorem 5.3.4. *Let L_1 and L_2 be two regular languages and $k \geq 1$ be a constant. Then it is decidable whether the parameterized suffix distance* suff-$D(n, L_1, L_2)$ *belongs to $\Omega(n^k) \cap O(n^{k+1})$.*

5.3.4 Calculation of the Suffix Distance

If the automaton H from Definition 5.2.3 is constructed based on the minimal DFA accepting the languages $L(\mathcal{A}_1)^R$ and $L(\mathcal{A}_2)^R$, the sum in Theorem 5.2.7 gives the exact parameterized suffix distance of the languages $L(\mathcal{A}_1)$ and $L(\mathcal{A}_2)$. This provides the following result.

Theorem 5.3.5. *The parameterized suffix distance can only be constant, polynomial, or superpolynomial.*

6 Concluding Remarks

In the beginning of this chapter, we will analyse the relation between the distances defined in Chapter 5 and the defects considered in the previous chapters. Afterwards, we discuss a different definition of the languages L_+, L_-, and $L_?$ from Chapter 4. This definition may be considered to be more detailed than the one already researched in the previous chapter.

6.1 Parameterized Distances and Defects

In Chapters 3 and 4, we have analysed the effects on the number of states of minimal deterministic finite automata for defects and languages related to these defects. Chapter 5 deals with parameterized distances of two regular languages. The relation between these distances and the defects is still missing.

The languages L_+ and $L_?$ are used again, that were already introduced in Chapter 4. To shortly recall the definitions of these languages, L_+ is the set that collects all words that are accepted by a defective finite automaton, that do not use the defect while being processed by this automaton. The language $L_?$ collects all the words, that are accepted or rejected by the defective automaton, and that use the defect at least once while being processed.

Qualitative Results Stating qualitative conclusions for defects requires a possibility to somehow measure the degree of severity of the defects. This can be done for example by using one of the parameterized distances, that were analysed in Chapter 5. This requires not only the knowledge of the measure and of the defective automaton, but also the language $L(\mathcal{A})$ that is accepted by the original DFA from which the defective one is retrieved. Then it is possible to measure the distance between the languages L_+ and $L(\mathcal{A})$ respectively $L_?$ and $L(\mathcal{A})$.

Depending on the purpose of the DFA, it is possible to value the sizes of the distances. If the distance between L_+ and $L(\mathcal{A})$ is constant, then there only exist finitely many words that differ. In this case, the defective automaton may still be useful for some applications, if finitely many faults are tolerable. The same holds true for constant distance of $L_?$ and $L(\mathcal{A})$.

In case that the distance of L_+ and $L(\mathcal{A})$ is polynomial, there exist infinitely, but not superpolynomially many words that differ in these two languages. Then the defective automaton may also still be useful. But for a superpolynomial distance of L_+ and $L(\mathcal{A})$, in general, the defective automaton cannot be used anymore, since there exist to many differences between the languages of the original and the defective automaton. Even if the distance between $L_?$ and $L(\mathcal{A})$ is superpolynomial, the defective automaton may not be useful anymore, since then there may be superpolynomially many words accepted that do not belong to $L(\mathcal{A})$. This massively changes the language.

Quantitative Results Let $\mathcal{A}$ be a minimal DFA, and $\mathcal{A}'$ be the defective automaton retrieved from $\mathcal{A}$ by some defect. For any of the defects considered in the former chapters, it is possible to construct a DFA $\mathcal{B}$ that only accepts the words of $L(\mathcal{A}')$ that belong to L_+. To be able to do this, the defect needs to be known, and also which of the transitions or states the defect affects. Having this information, the automaton for L_+ can be constructed from automaton $\mathcal{A}'$ by simply deleting the defective transitions and states. This automaton is a DFA, but it does not need to be minimal and not even complete.

Definition 6.1.1. Let $\mathcal{A}$ denote a minimal DFA, and let $\mathcal{A}'$ be the automaton resulting from $\mathcal{A}$ by some defect. The automaton accepting L_+ of automaton $\mathcal{A}'$ is denoted by $\mathcal{B}$. For a fixed number $n \geq 0$, $\text{cens}_{n,\text{g}}$ is the number of words up to length n that belong to L_+, that is

$$\text{cens}_{n,+} = \text{cens}_{\mathcal{B}}(n),$$

and

$$\text{cens}_{n,\text{o}} = \text{cens}_{\mathcal{A}}(n)$$

denotes the number of words up to length n, that are accepted by $\mathcal{A}$. Then,

$$\text{cens}_{n,\text{diff}} := \text{cens}_{n,\text{o}} - \text{cens}_{n,+}$$

gives the number of words of language $L(\mathcal{A})$, that are not accepted by $\mathcal{B}$ up to the fixed length n. These words may still be accepted by $\mathcal{A}'$, but, in any case, the defect is used while processing such a word.

Fact 6.1.1. The smaller $\text{cens}_{\text{diff}}$ is, the more words of $L(\mathcal{A})$ are accepted by the defective automaton $\mathcal{A}'$ without using the defect.

This does not necessarily mean, that the languages are equal. This is not true, since it is possible that by $L_?$, a big number of words is inserted to the language $L(\mathcal{A}')$, that do not belong to $L(\mathcal{A})$. This means, the symmetric difference of the languages $L(\mathcal{A})$ and $L(\mathcal{A}')$ may consist of infinitely many words.

From all of this, we can deduce, that the use of only the census function as a measure for the difference of the languages $L(\mathcal{A})$ and $L(\mathcal{A}')$, is not sufficient to make valuable predications about the usability of the defective automaton.

Combined Results The combination of the two measures census and distance establishes the possibility to decide the degree of severity of the defect and the usefulness of the defective automaton. For example it may be possible that $\text{cens}_{n,\text{diff}}$ is big for some $n \geq 0$, which means that a lot of words of $L(\mathcal{A})$ are not accepted anymore in the defective automaton not using the defect, but pref-$D(n,L_+,L(\mathcal{A}))$ is small, perhaps only linear. Then the defect seems to affect only some short suffixes of some words. This may be tolerable. For example natural languages are fault tolerable up to a certain degree, since the suffix of a word is not that important for the comprehension of a word.

6.2 A Different Definition of L_+, L_- and $L_?$

In Chapter 4, the language L_+ was defined to be the collection of all words accepted by the defective automaton, that do not use the defect. The language L_- was defined quite similar, but concerned the rejected words. The language $L_?$ contained all of the words processed by the defective automaton using the defect.

To be able to construct these languages, it is necessary to know the defect. It does not suffice to know only the type of defect, but it is also needed to know which state is affected by this defect. This means, the special state needs to be known that is defective or for which a transition is defective. For the defects that concern the transitions, this state can always be determined. But for the defects that affect the acceptance property of a state, in general, it is impossible to determine the defective state.

Knowing only the type of defect responsible for the given defective automaton, we can try to correct the defect. We can construct all possible deterministic automata, from which the resulting defective one can be received by precisely the known defect. There only exist finitely many of these automata for each of the possible defects. In the following, we will consider this fact for all of the defects separately. Based on these corrected automata, the language L_+ is defined to be the set of words accepted by all of these automata. Language L_- collects all words that are rejected by all of these corrected automata, and, finally, $L_?$ is the language that is accepted or rejected by only some of these automata. Formally, let $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_k$ denote the different corrected automata, $k \geq 1$. Then L_+ can be denoted by $L(\mathcal{A}_1) \cap L(\mathcal{A}_2) \cap \cdots \cap L(\mathcal{A}_k)$, the language L_- is $\overline{L(\mathcal{A}_1)} \cap \overline{L(\mathcal{A}_2)} \cap \cdots \cap \overline{L(\mathcal{A}_k)}$, where $\overline{L(\mathcal{A}_i)} = \Sigma^* \setminus L(\mathcal{A}_i)$ is the complement of $L(\mathcal{A}_i)$, that means all words rejected by automaton $\mathcal{A}_i$, and $L_?$ can be considered to be $\Sigma^* \setminus (L_+ \cup L_-)$, where Σ is the underlying alphabet of the defective automaton. The language $L_?$ can also be expressed by

$$\begin{aligned}
&\overline{(L(\mathcal{A}_1) \cap L(\mathcal{A}_2) \cap \cdots \cap L(\mathcal{A}_k)) \cup \left(\overline{L(\mathcal{A}_1)} \cap \overline{L(\mathcal{A}_2)} \cap \cdots \cap \overline{L(\mathcal{A}_k)}\right)} \\
&= \left(\overline{L(\mathcal{A}_1)} \cup \overline{L(\mathcal{A}_2)} \cup \cdots \cup \overline{L(\mathcal{A}_k)}\right) \cap (L(\mathcal{A}_1) \cup L(\mathcal{A}_2) \cup \cdots \cup L(\mathcal{A}_k)) \,.
\end{aligned}$$

This means, $L_?$ collects all the words of the repaired automata, that are accepted or rejected by at least one but not by all of the automata at the same time.

In the following, we assume again that only one occurrence of one type of defect lead to the given defective automaton. Based on this assumption, upper bounds for the number of states for minimal DFA accepting the so defined languages L_+, L_- resprectively $L_?$ can be derived from upper bounds for the constructions of minimal

DFA accepting the union, intersection, and the complement of regular languages. These bounds are $m \cdot n$ for the union and the intersection, if the united or intersected minimal DFA consist of m respectively n states. For the complement of a DFA with n states, this bound is n.

All of the repaired automata will be DFA. Their equivalent minimal DFA have at most equally many states. Let $k \geq 1$ denote the number of repaired automata, n the number of states of the original automaton, F the set of accepting states of the original automaton, and Σ the alphabet of the automata. For all defect, the minimal DFA accepting L_+ has at most n^k states, the one accepting L_- also n^k states, and the minimal DFA accepting $L_?$ consists of at most n^{2k} states. In the following, we will determine the number k for each defect separately.

Exchange Symbol For this type of defect, we already know that there exists one state s with an undefined transition for precisely one symbol a, and two transitions on the same symbol b. The correction is done by replacing the symbol b of one of the transitions of state s by the symbol a. This leads to at most two different and perhaps minimal DFA.

Insert Transition There also exist at most two repaired automata for this defect. An automaton with this type of defect consists of one state with no undefined transition, but two transitions on the same symbol. To repair this defect, one of these two transitions needs to be deleted.

Flip Transition The flipped transition can be recognised and repaired by exchanging the source and target of this transition. Therefore, there only exists one repaired automaton.

Delete Transition For a deleted transition, there exists one state in the defective automaton with one undefined transition. So the source of the deleted transition can be recognised. The correction of this defect leads to n different automata, since there exist n possible targets for the transition that needs to be inserted.

Delete Accepting State The deletion of an accepting state leads to undefined transitions in the defective automaton. When inserting a new accepting state, the resulting automaton has n states again, just like the original automaton. This new state is the target for all of the missing transitions of the defective automaton. But the targets of the transitions starting in this new state are still unknown. This leads to a number of $\binom{n}{|\Sigma|}$ different repaired automata.

Delete Non-Accepting State For this defect, there also exist at most $\binom{n}{|\Sigma|}$ different repaired automata. The correction is done quite similar to the one for the defect delete accepting state.

	L_+	L_-	$L_?$
Delete Transition	n^n	n^n	n^{2n}
Insert Transition	n^2	n^2	n^4
Flip Transition	n	n	n^2
Exchange Symbol	n^2	n^2	n^4
Delete Non-Acc. State	$n^{\binom{n}{\lvert\Sigma\rvert}}$	$n^{\binom{n}{\lvert\Sigma\rvert}}$	$n^{2\binom{n}{\lvert\Sigma\rvert}}$
Delete Accepting State	$n^{\binom{n}{\lvert\Sigma\rvert}}$	$n^{\binom{n}{\lvert\Sigma\rvert}}$	$n^{2\binom{n}{\lvert\Sigma\rvert}}$
Remove Acceptance	$n^{n-(\lvert F\rvert-1)}$	$n^{n-(\lvert F\rvert-1)}$	$n^{2(n-(\lvert F\rvert-1))}$
Add Acceptance	$n^{\lvert F\rvert+1}$	$n^{\lvert F\rvert+1}$	$n^{2(\lvert F\rvert+1)}$

Table 6.1: Upper bounds for the number of states of minimal DFA accepting the languages L_+, L_-, and $L_?$ for the different defects, where n is the number of states of the original DFA, Σ is its alphabet, and F is the set of accepting states of the original DFA.

Remove Acceptance This defect is corrected by adding the property of acceptance to one of the non-accepting states of the defective automaton. This leads to a number of $n - (|F| - 1)$ repaired automata.

Add Acceptance The correction of this defect is done by removing the acceptance for one of the accepting states of the defective automaton. Therefore, there possibly exist $|F| + 1$ many repaired automata.

Tabular 6.1 summarises the results.

6.3 Future Research

In the future, the relation of the distances and the defects should be further researched. The question for the existence of distance classes for the different defects may be considered. This means, when fixing one type of defect, which distance classes can be reached (linear, polynomial, exponential). Perhaps in this direction, there exist better possiblities to decide if a defective automaton can still be used for some applications.

Of course, the parameterized distance can also be considered to be defined for other word distances like the edit distance or the Hamming distance. Maybe these distances lead to other distance classes besides linear, polynomial, and exponential.

For the defect of insertion of a transition, it is still an open problem if the upper bound of $2^n - 1$ is reachable for binary alphabet. In this thesis, the tightness of this bound was only shown for unary and at least ternary alphabets. It may also be possible to improve the bounds for finite languages. For the defects insert transition, flip transition, or exchang symbol, the upper bounds were only shown to be tight in the order of magnitude.

It would also be possible to consider defects different to the ones considered in this thesis. One source for such defects may be applications where DFA need to be exchanged in some way.

Bibliography

[1] Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. *Compilers: Principles, Techniques, and Tools.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1986.

[2] Alfred V. Aho and Jeffrey D. Ullman. *The Theory of Parsing, Translation, and Compiling.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1972.

[3] Henning Bordihn, Markus Holzer, and Martin Kutrib. Determinization of finite automata accepting subregular languages. *Theoretical Computer Science*, 410(35):3209 – 3222, 2009.

[4] Janusz A. Brzozowski. Canonical regular expressions and minimal state graphs for definite events. *Mathematical Theory of Automata*, 12:529–561, 1962.

[5] Janusz A. Brzozowski. Quotient complexity of regular languages. *Journal of Automata, Languages and Combinatorics*, 15(1/2):71–89, 2010.

[6] Janusz A. Brzozowski, Galina Jirásková, Baiyu Li, and Joshua Smith. Quotient complexity of bifix-, factor-, and subword-free regular languages. In *Automata and Formal Languages, 13th International Conference, AFL 2011, Debrecen, Hungary, August 17-22, 2011, Proceedings.*, pages 123–137, 2011.

[7] Janusz A. Brzozowski, Galina Jirásková, and Chenglong Zou. Quotient complexity of closed languages. *Theory Comput. Syst.*, 54(2):277–292, 2014.

[8] Janusz A. Brzozowski and Bo Liu. Quotient complexity of star-free languages. *International Journal of Foundations of Computer Science*, 23(6):1261–1276, 2012.

[9] Janusz A. Brzozowski and University of Waterloo. Department of Computer Science. *Open Problems about Regular Languages.* University of Waterloo Computer Science Department. Department of Computer Science, University of Waterloo, 1980.

[10] Janusz A. Brzozowski and M. Yoeli. *Digital networks.* Prentice-Hall series in automatic computation. Prentice-Hall, 1976.

[11] Cezar Câmpeanu, Karel Čulik, Kai Salomaa, and S. Yu. State complexity of basic operations on finite languages. In Oliver Boldt and Helmut Jürgensen, editors, *Workshop on Implementing Automata (WIA 1999)*, volume 2214 of *LNCS*, pages 60–70. Springer, 2001.

[12] Ashok K. Chandra, Dexter Kozen, and Larry J. Stockmeyer. Alternation. *J. ACM*, 28(1):114–133, 1981.

[13] Christian Choffrut and Giovanni Pighizzini. Distances between languages and reflexivity of relations. *Theor. Comput. Sci.*, 286:117–138, 2002.

[14] Noam Chomsky. Three models for the description of language. *IRE Transactions on Information Theory*, 2:113–124, 1956.

[15] Noam Chomsky. On certain formal properties of grammars. *Information and Control*, 2:137–167, 1959.

[16] Marek Chrobak. Finite automata and unary languages. *Theoretical Computer Science*, 47(3):149–158, 1986.

[17] Marek Chrobak. Errata to: &#x0022;finite automata and unary languages&#x0022;. *Theor. Comput. Sci.*, 302(1-3):497–498, 2003.

[18] Aldo de Luca and Stefano Varricchio. On noncounting regular classes. *Theoretical Computer Science*, 100(1):67 – 104, 1992.

[19] Yuan Gao, Nelma Moreira, Rogério Reis, and Sheng Yu. A survey on operational state complexity. *CoRR*, abs/1509.03254, 2015.

[20] Viliam Geffert. (non)determinism and the size of one-way finite automata. In Carlo Mereghetti, Beatrice Palano, Giovanni Pighizzini, and Detlef Wotschke, editors, *DCFS*, pages 23–37. Universit degli Studi di Milano, Milan, Italy, 2005.

[21] Viliam Geffert. Magic numbers in the state hierarchy of finite automata. *Information and Computation*, 205(11):1652–1670, 2007.

[22] Seymour Ginsburg and Sheila Greibach. Deterministic context free languages. *Information and Control*, 9(6):620 – 648, 1966.

[23] Leonard H. Haines. *Generation and Recognition of Formal Languages*. PhD Thesis, Massachusetts Institute of Technology, Deparment of Mathematics, Cambridge, Massachusetts, USA, 1965.

[24] Kosaburo Hashiguchi. Algorithms for determining relative star height and star height. *Information and Computation*, 78(2):124 – 169, 1988.

[25] Markus Holzer, Sebastian Jakobi, and Martin Kutrib. The magic number problem for subregular language families. *International Journal of Foundations of Computer Science*, 23(1):115–131, 2012.

[26] Markus Holzer and Martin Kutrib. State complexity of basic operations on nondeterministic finite automata. In *Implementation and Application of Automata, 7th International Conference, CIAA 2002, Tours, France, July 3-5, 2002, Revised Papers*, pages 148–157, 2002.

[27] Markus Holzer and Martin Kutrib. Descriptional and computational complexity of finite automataa survey. *Information and Computation*, 209(3):456 – 470, 2011. Special Issue: 3rd International Conference on Language and Automata Theory and Applications (LATA 2009).

[28] Markus Holzer and Martin Kutrib. *Descriptional Complexity An Introductory Survey*, chapter 1, pages 1–58. Imperial College Press, 2011.

[29] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to automata theory, languages, and computation - international edition (2. ed).* Addison-Wesley, 2003.

[30] David A. Huffman. The synthesis of sequential switching circuits. *Journal of the Franklin Institute*, 257:161 – 190, 1954.

[31] Kazuo Iwama, Yahiko Kambayashi, and Kazuya Takaki. Tight bounds on the number of states of {DFAs} that are equivalent to n-state {NFAs}. *Theoretical Computer Science*, 237:485 – 494, 2000.

[32] Kazuo Iwama, Akihiro Matsuura, and Mike Paterson. A family of nfas which need $2^n - \alpha$ deterministic states. *Theoretical Computer Science*, 301:451 – 462, 2003.

[33] Jozef Jirásek, Galina Jirásková, and Alexander Szabari. Deterministic blow-ups of minimal nondeterministic finite automata over a fixed alphabet. In *Developments in Language Theory, 11th International Conference, DLT 2007, Turku, Finland, July 3-6, 2007, Proceedings*, pages 254–265, 2007.

[34] Galina Jirásková. Note on minimal finite automata. In Ji Sgall, Ale Pultr, and Petr Kolman, editors, *Mathematical Foundations of Computer Science 2001*, volume 2136 of *Lecture Notes in Computer Science*, pages 421–431. Springer Berlin Heidelberg, 2001.

[35] Galina Jirásková. On the state complexity of complements, stars, and reversals of regular languages. In Masami Ito and Masafumi Toyama, editors, *Developments in Language Theory*, volume 5257 of *Lecture Notes in Computer Science*, pages 431–442. Springer Berlin Heidelberg, 2008.

[36] Galina Jirásková. Concatenation of regular languages and descriptional complexity. In Anna Frid, Andrey Morozov, Andrey Rybalchenko, and KlausW. Wagner, editors, *Computer Science - Theory and Applications*, volume 5675 of *Lecture Notes in Computer Science*, pages 203–214. Springer Berlin Heidelberg, 2009.

[37] Galina Jirásková. Magic numbers and ternary alphabet. In Volker Diekert and Dirk Nowotka, editors, *Developments in Language Theory*, volume 5583 of *Lecture Notes in Computer Science*, pages 300–311. Springer Berlin Heidelberg, 2009.

[38] Galina Jirásková and Juraj Sebej. Note on reversal of binary regular languages. In *DCFS*, volume 6808 of *Lecture Notes in Computer Science*, pages 212–221. Springer, 2011.

[39] Galina Jirskov. State complexity of some operations on binary regular languages. *Theoretical Computer Science*, 330(2):287 – 298, 2005.

[40] Christos Kapoutsis. Algorithms and lower bounds in finite automata size complexity. *Phd Thesis*, 2006.

[41] Stephen C. Kleene. Representation of events in nerve nets and finite automata. In Claude Shannon and John McCarthy, editors, *Automata Studies*, volume AM-34, pages 3–41. Princeton University Press, Princeton, USA, 1956.

[42] Donald E. Knuth, James H. Morris, and Vaughan R. Pratt. Fast Pattern Matching in Strings. *SIAM Journal on Computing*, 6(2):323–350, 1977.

[43] S.-Y. Kuroda. Classes of languages and linear-bounded automata. *Information and Control*, 7(2):207 – 223, 1964.

[44] Martin Kutrib. The phenomenon of non-recursive trade-offs. *International Journal of Foundations of Computer Science*, 16(5):957–973, 2005.

[45] Martin Kutrib, Katja Meckel, and Matthias Wendlandt. Parameterized prefix distance between regular languages. In *SOFSEM 2014: Theory and Practice of Computer Science - 40th International Conference on Current Trends in Theory and Practice of Computer Science, Nový Smokovec, Slovakia, January 26-29, 2014, Proceedings*, pages 419–430, 2014.

[46] Peter S. Landweber. Three theorems on phrase structure grammars of type 1. *Information and Control*, 6(2):131 – 136, 1963.

[47] Harry R. Lewis and Christos H. Papadimitriou. *Elements of the theory of computation (2. ed.)*. Prentice Hall, 1998.

[48] Oleg B. Lupanov. A comparison of two types of finite sources. *Problemy Kibernetiki*, 9:321–326, 1963. (in Russian).

[49] Akihiro Matsuura and Yusuke Saito. Equivalent transformation of minimal finite automata over a two-letter alphabet. In Cezar Câmpeanu and Giovanni Pighizzini, editors, *DCFS*, pages 224–232. University of Prince Edward Island, 2008.

[50] George H. Mealy. A method for synthesizing sequential circuits. *The Bell Systems Technical Journal*, 34(5):1045–1079, 1955.

[51] Albert R. Meyer and Michael J. Fischer. Economy of description by automata, grammars, and formal systems. In *12th Annual Symposium on Switching and*

Automata Theory, East Lansing, Michigan, USA, October 13-15, 1971, pages 188–191, 1971.

[52] Mehryar Mohri. On the use of sequential transducers in natural language processing. In Emmanuel Roche and Yves Shabes, editors, *Finite-State Language Processing*, chapter 12, pages 355–381. MIT Press, Cambridge, MA, USA, 1997.

[53] Edward F. Moore. Gedanken-experiments on sequential machines. In Claude Shannon and John McCarthy, editors, *Automata Studies*, volume AM-34, pages 129–153. Princeton University Press, Princeton, NJ, 1956.

[54] Frank R. Moore. On the bounds for state-set size in the proofs of equivalence between deterministic, nondeterministic, and two-way finite automata. *IEEE Transaction on Computing*, 20:1211–1219, 1971.

[55] John R. Myhill. Finite automata and the representation of events. Technical Report WADD TR-57-624, Wright Patterson Air Force Base, Ohio, USA, 1957.

[56] John R. Myhill. Linear bounded automata. Technical report, Wright Patterson Air Force Base, Ohio, USA, 1960.

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

[58] Michael O. Rabin and Dana S. Scott. Finite automata and their decision problems. *IBM J. Res. Dev.*, 3:114–125, 1959.

[59] Bala Ravikumar. Some applications of a technique of sakoda and sipser. *SIGACT News*, 21(4):73–77, 1990.

[60] Bala Ravikumar and Oscar H. Ibarra. Relating the type of ambiguity of finite automata to the succinctness of their representation. *SIAM J. Comput.*, 18(6):1263–1282, 1989.

[61] Kai Salomaa and Sheng Yu. NFA to DFA transformation for finite languages over arbitrary alphabets. *Journal of Automata, Languages and Combinatorics*, 2(3):177–186, 1997.

[62] Marcel P. Schützenberger. On context-free languages and push-down automata. *Information and Control*, 6(3):246 – 264, 1963.

[63] Peter W. Shor. A counterexample to the triangle conjecture. *Journal of Combinatorial Theory, Series A*, 38(1):110 – 112, 1985.

[64] Angelika Steger. *Diskrete Strukturen (Band 1)*. Springer-Verlag, 2001.

[65] Andrew Szilard, Sheng Yu, Kaizhong Zhang, and Jeffrey Shallit. Characterizing regular languages with polynomial densities. In *MFCS*, volume 629, pages 494–503, 1992.

[66] Alan M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, s2-42(1):230–265, 1936.

[67] Sheng Yu. Regular languages. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 1, chapter 2, pages 41–110. Springer, Berlin, 1997.

[68] 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.

Ich erkläre:

Ich habe die vorgelegte Dissertation selbständig und ohne unerlaubte fremde Hilfe und nur mit den Hilfen angefertigt, die ich in der Dissertation angegeben habe.
Alle Textstellen, die wörtlich oder sinngemäß aus veröffentlichten Schriften entnommen sind, und alle Angaben, die auf mündlichen Auskünften beruhen, sind als solche kenntlich gemacht.
Bei den von mir durchgeführten und in der Dissertation erwähnten Untersuchungen habe ich die Grundsätze guter wissenschaftlicher Praxis, wie sie in der "Satzung der Justus-Liebig-Universität Gießen zur Sicherung guter wissenschaftlicher Praxis" niedergelegt sind, eingehalten.

Wetzlar, Dezember 2015

____
(Katja Meckel)