Department of Computer Science and Engineering
Faculty of Electrical Engineering
Czech Technical University in Prague

TEXT SEARCHING ALGORITHMS

SEMINARS

Bořivoj Melichar, Jan Antoš, Jan Holub, Tomáš Polcar,
and Michal Voráček

December 21, 2005

Preface

"Practice makes perfect."

The aim of this tutorial text is to facilitate deeper understanding of principles and applications of text searching algorithms. It provides many exercises from different areas of this topic. The front part of this text is devoted to review of basic notions, principles and algorithms from the theory of finite automata. The reason for it is the reality that the next chapters devoted to different aspects of the area of text searching algorithms are based on the use of finite automata.

Chapter 1 contains collection of definitions of notions used in this tutorial. Chapter 2 is devoted to the basic algorithms from the area of finite automata and has been written by Jan Antoš. Chapter 3 contains basic algorithms and operations with finite automata and it has been partly written by Jan Antoš. Chapters 4 and 5 show construction of string matching automata for exact and approximate string matching and has been partly written by Tomáš Polcar. Chapter 6, devoted to the simulation of string matching automata has been written by Jan Holub. Chapter 7 and 8 concerning construction of finite automata accepting parts of strings has been partly written by Tomáš Polcar. Chapter 9 describing computation of border arrays has been written by Michal Voráček.

Collection of exercises in this tutorial text is based on notions, principles and algorithms described in main tutorial text *Text searching algorithms. Volume I, II.* Reference to this text has the form: [TSA, Chapter xx].

Special thanks of the authors go to Miroslav Balík, Martin Šimůnek and Jan Šupol for typesetting of the text using system LaTeX and to Olga Vrtišková for drawing all pictures using system Corel Draw and for her willingness to react on all changes, additions and corrections.

Authors

Prague, December 21, 2005

Contents

1 Definitions

This chapter provides the essential definitions needed for understanding this text. These definitions were incorporated here for more comfortable reading.

1.1 Basics

Definition 1.1 (Alphabet)
An *alphabet* A is a finite non-empty set of *symbols*.

Definition 1.2 (Complement of symbol)
A *complement* of symbol a over A, where $a \in A$, is a set $A \setminus \{a\}$ and is denoted $\overline{a}$.

Definition 1.3 (String)
A *string* over A is any sequence of symbols from A.

Definition 1.4 (Set of all strings)
The *set of all strings* over A is denoted A^*.

Definition 1.5 (Set of all non-empty strings)
The *set of all non-empty strings* over A is denoted A^+.

Definition 1.6 (Length of string)
The *length of string* x is the number of symbols in string $x \in A^*$ and is denoted $|x|$.

Definition 1.7 (Empty string)
An *empty string* is a string of length 0 and is denoted ε.

Remark 1.8
It holds $A^* = A^+ \cup \{\varepsilon\}$.

Notation 1.9
Exponents for string with repetitions will be used: $a^0 = \varepsilon$, $a^1 = a$, $a^2 = aa$, $a^3 = aaa$, ..., for $a \in A$ and $x^0 = \varepsilon, x^1 = x, x^2 = xx, x^3 = xxx, \ldots$, for $x \in A^*$.

Definition 1.10 (Concatenation)
The operation *concatenation* is defined over the set of strings A^* as follows: if x and y are strings over the alphabet A, then by appending the string y to the string x we obtain the string xy.

1.2 Finite automata

Definition 1.11 (Deterministic finite automaton)
A deterministic finite automaton (*DFA*) is a quintuple $M = (Q, A, \delta, q_0, F)$, where
Q is a finite set of states,
A is a finite input alphabet,
δ is a mapping from $Q \times A$ to Q, $(Q \times A \mapsto Q)$
$q_0 \in Q$ is an initial state,
$F \subset Q$ is the set of final states.

Definition 1.12 (Configuration of FA)
Let $M = (Q, A, \delta, q_0, F)$ be a finite automaton. A pair $(q, w) \in Q \times A^*$ is *a configuration of the*

finite automaton M. A configuration (q_0, w) is called *an initial configuration*, a configuration (q, ε), where $q \in F$, is called *a final (accepting) configuration* of the finite automaton M.

Definition 1.13 (Transition in DFA)
Let $M = (Q, A, \delta, q_0, F)$ be a deterministic finite automaton. A relation $\vdash_M \in (Q \times A^*) \times (Q \times A^*)$ is called *a transition* in the automaton M. If $\delta(q, a) = p$, then $(q, aw) \vdash_M (p, w)$ for each $w \in A^*$. The k-power of the relation $\vdash_M$ will be denoted by $\vdash_M^k$. The symbols $\vdash_M^+$ and $\vdash_M^*$ denote a transitive and a transitive reflexive closure of the relation $\vdash_M$, respectively.

Definition 1.14 (Language accepted by DFA)
We will say that an input string $w \in A^*$ *is accepted* by a finite deterministic automaton $M = (Q, A, \delta, q_0, F)$ if $(q_0, w) \vdash_M^* (q, \varepsilon)$ for some $q \in F$.
The language $L(M) = \{w : w \in T^*, (q_0, w) \vdash^* (q, \varepsilon), q \in F\}$ is *the language accepted* by a finite automaton M. A string $w \in L(M)$ if it consists only of symbols from the input alphabet and there is a sequence of transitions such that it leads from the initial configuration (q_0, w) to the final configuration (q, ε), $q \in F$.

Definition 1.15 (Complete DFA)
A finite deterministic automaton $M = (Q, A, \delta, q_0, F)$ is said to be *complete* if the mapping $\delta(q, a)$ is defined for each pair of states $q \in Q$ and input symbols $a \in A$.

Definition 1.16 (Nondeterministic finite automaton)
A nondeterministic finite automaton (*NFA*) is a quintuple $M = (Q, A, \delta, q_0, F)$, where
Q is a finite set of states,
A is a finite input alphabet,
δ is a mapping from $Q \times A$ into the set of subsets of Q,
$q_0 \in Q$ is an initial state,
$F \subset Q$ is the set of final states.

Definition 1.17 (Transition in NFA)
Let $M = (Q, A, \delta, q_0, F)$ be a nondeterministic finite automaton. A relation $\vdash_M \subset (Q \times A^*) \times (Q \times A^*)$ will be called *a transition* in the automaton M if $p \in \delta(q, a)$ then $(q, aw) \vdash_M (p, w)$, for each $w \in A^*$.

Definition 1.18 (Language accepted by NFA)
A string $w \in A^*$ is said to be *accepted by a nondeterministic finite automaton* $M = (Q, A, \delta, q_0, F)$, if there exists a sequence of transitions $(q_0, w) \vdash^* (q, \varepsilon)$ for some $q \in F$.
The language $L(M) = \{w : w \in A^*, (q_0, w) \vdash^* (q, \varepsilon)$ for some $q \in F\}$ is then *t*he language accepted by a nondeterministic finite automaton M.

Definition 1.19 (NFA with ε-transitions)
A nondeterministic finite automaton with ε-transitions is a quintuple $M = (Q, A, \delta, q_0, F)$, where
Q is a finite set of states,
A is a finite input alphabet,
δ is a mapping from $Q \times (A \cup \{\varepsilon\})$ into the set of subsets of Q,
$q_0 \in Q$ is an initial state,
$F \subset Q$ is the set of final states.

Definition 1.20 (Transition in NFA with ε-transitions)
Let $M = (Q, A, \delta, q_0, F)$ be a nondeterministic finite automaton with ε-transitions. A relation

$\vdash_M \subset (Q \times A^*) \times (Q \times A^*)$ will be called *a transition* in the automaton M if $p \in \delta(q,a)$, $a \in A \cup \{\varepsilon\}$, then $(q, aw) \vdash_M (p, w)$, for each $w \in A^*$.

Definition 1.21 (ε–*CLOSURE*)
Function ε–$CLOSURE$ for finite automaton $M = (Q, A, \delta, q_0, F)$ is defined as:

$$\varepsilon\text{–}CLOSURE(q) = \{p : (q, \varepsilon) \vdash^* (p, \varepsilon), p \in Q\}.$$

Definition 1.22 (NFA with set of initial states)
A nondeterministic finite automaton M with set of initial states I is a quintuple
$M = (Q, A, \delta, I, F)$, where:
Q is a finite set of states,
A is a finite input alphabet,
δ is a mapping from $Q \times A$ into the set of subsets of Q,
$I \subset Q$ is the non-empty set of initial states,
$F \subset Q$ is the set of final states.

Definition 1.23 (Accessible state)
Let $M = (Q, A, \delta, q_0, F)$ be a finite automaton. A state $q \in Q$ is called *accessible* if there exists a string $w \in A^*$ such that there exists a sequence of transitions from the initial state q_0 into the state q:

$$(q_0, w) \vdash_M (q, \varepsilon)$$

A state which is not accessible is called *inaccessible*.

Definition 1.24 (Useful state)
Let $M = (Q, A, \delta, q_0, F)$ be a finite automaton. A state $q \in Q$ is called *useful* if there exists a string $w \in A^*$ such that there exists a sequence of transitions from the state q into some final state:

$$(q, w) \vdash_M (p, \varepsilon),\ p \in F.$$

A state which is not useful is called *useless*.

Definition 1.25 (Finite automaton)
Finite automaton (*FA*) is DFA or NFA.

Definition 1.26 (Equivalence of finite automata)
Finite automata M_1 and M_2 are said to be *equivalent* if they accept the same language, it means $L(M_1) = L(M_2)$.

Definition 1.27 (Sets of states)
Let $M = (Q, A, \delta, q_0, F)$ be a finite automaton. Let us define for arbitrary $a \in A$ the set $Q(a) \subset Q$ as follows:

$$Q(a) = \{q : q \in \delta(p, a), a \in A, p, q \in Q\}.$$

Definition 1.28 (Homogenous automaton)
Let $M = (Q, A, \delta, q_0, F)$ be a finite automaton and $Q(a)$ be sets of states for all symbols $a \in T$. If for all pairs of symbols $a, b \in A$, $a \neq b$, it holds $Q(a) \cap Q(b) = \emptyset$, then the automaton M is called homogenous. The collection of sets $\{Q(a) : a \in A\}$ is for the homogenous finite automaton the decomposition on classes having one of these two forms:

1. $Q = \bigcup_{a \in A} Q(a) \cup \{q_0\}$ in case that $q_0 \notin \delta(q, a)$ for all $q \in Q$ and all $a \in A$,

2. $Q = \bigcup_{a \in A} Q(a)$ in case that $q_0 \in \delta(q, a)$ for some $q \in Q, a \in A$. In this case $q_0 \in Q(a)$.

1.3 Text searching

Definition 1.29 (Replace)
Edit operation *replace* is an operation which converts string wav to string wbv, where $w, v \in A^*$, $a, b \in A$ (one symbol is replaced by another).

Definition 1.30 (Insert)
Edit operation *insert* is an operation which converts string wv to string wav, where $w, v \in A^*$, $a \in A$ (a symbol is inserted into a string).

Definition 1.31 (Delete)
Edit operation *delete* is an operation which converts string wav to string wv, where $w, v \in A^*$, $a \in A$ (a symbol is removed from a string).

Definition 1.32 (Transpose)
Edit operation *transpose* is an operation which converts string $wabv$ to string $wbav$, where $w, v \in A^*$, $a, b \in A$ (two adjacent symbols are exchanged).

Definition 1.33 (Distances of strings)
Three variants of distances between two strings u and v, where $u, v \in A^*$, are defined as minimal number of editing operations:

1. *replace* (Hamming distance, R-distance),
2. *delete, insert* and *replace* (Levenshtein distance, *DIR*-distance),
3. *delete, insert, replace* and *transpose* (generalized Levenshtein distance, *DIR* T-distance),

needed to convert string u into string v.

Definition 1.34 ("Don't care" symbol)
"Don't care" symbol is a special universal symbol $\circ$ that matches any other symbol including itself.

Definition 1.35 (Set of all prefixes)
The set $Pref(x),\ x \in A^*$, is a set of all prefixes of string x:
$Pref(x) = \{y : x = yu,\ x, y, u \in A^*\}$.

Definition 1.36 (Set of all suffixes)
The set $Suff(x),\ x \in A^*$, is a set of all suffixes of string x:
$Suff(x) = \{y : x = uy,\ x, y, u \in A^*\}$.

Definition 1.37 (Set of all factors)
The set $Fact(x),\ x \in A^*$, is a set of all factors of the string x:
$Fact(x) = \{y : x = uyv,\ x, y, u, v \in A^*\}$.

Definition 1.38 (Set of all subsequences)
The set $Sub(x),\ x \in A^*$, is a set of all subsequences of the string x:

$$Sub(x) = \{\, a_1 a_2 \dots a_m : x = y_0 a_1 y_1 a_2 \dots a_m y_m, \\ y_i \in A^*, i = 0, 1, 2, \dots, m, a_j \in A, \\ j = 1, 2, \dots, m,\ m \geq 0\}$$

Definition 1.39 (Set of approximate prefixes)
The set of approximate prefixes *APref* of the string x is a set:
$APref(x) = \{u : v \in Pref(x),\ D(u, v) \leq k\}$.

Definition 1.40 (Set of approximate suffixes)
The set of approximate suffixes *ASuff* of the string x is a set:
$ASuff(x) = \{u : v \in Suff(x),\ D(u, v) \leq k\}$.

Definition 1.41 (Set of approximate factors)
The set of approximate factors $AFact$ of the string x is a set:
$AFact(x) = \{u : v \in Fact(x),\ D(u, v) \leq k\}$.

Definition 1.42 (Set of approximate subsequences)
The set of approximate subsequences $ASub$ of the string x is a set:
$ASub(x) = \{u : v \in Sub(x),\ D(u, v) \leq k\}$.

Definition 1.43 (Prefix automaton)
The *prefix automaton* for string u is a finite automaton accepting the language $Pref(u)$.

Definition 1.44 (Suffix automaton)
The *suffix automaton* for string u is a finite automaton accepting the language $Suff(u)$.

Definition 1.45 (Factor automaton)
The *factor automaton* for string u is a finite automaton accepting the language $Fact(u)$.

Definition 1.46 (Subsequence automaton)
The *subsequence automaton* for string u is a finite automaton accepting the language $Sub(u)$.

Definition 1.47 (Approximate prefix automaton)
The *approximate prefix automaton* for string u is a finite automaton accepting the language $APref(u)$.

Definition 1.48 (Approximate suffix automaton)
The *approximate suffix automaton* for string u is a finite automaton accepting the language $ASuff(u)$.

Definition 1.49 (Approximate factor automaton)
The *approximate factor automaton* for string u is a finite automaton accepting the language $AFact(u)$.

Definition 1.50 (Approximate subsequence automaton)
The *approximate subsequence automaton* for string u is a finite automaton accepting the language $ASub(u)$.

Definition 1.51 (Basic pattern matching problems)
Given a text string $T = t_1 t_2 \cdots t_n$ and a pattern $P = p_1 p_2 \cdots p_m$. Then we may define:

1. String matching: verify whether string P is a substring of text T.

2. Sequence matching: verify whether sequence P is a subsequence of text T.
3. Subpattern matching: verify whether a subpattern of P (substring or subsequence) occurs in text T.
4. Approximate pattern matching: verify whether pattern X occurs in text T so that the distance $D(P, X) \leq k$ for given $k < m$.
5. Pattern matching with "don't care" symbols: verify whether pattern P containing "don't care" symbols occurs in text T.

Definition 1.52 (Matching a sequence of patterns)
Given a text string $T = t_1 t_2 \cdots t_n$ and a sequence of patterns (strings or sequences) $P_1, P_2, \ldots$, P_s. Matching of sequence of patterns $P_1, P_2, \ldots, P_s$ is a verification whether an occurrence of pattern P_i in text T is followed by an occurrence of P_{i+1}, $1 \leq i < s$.

2 Mastering finite automata

In this Chapter we show some basic operations that can be applied to finite automata. We then explain algorithms for performing these operations and we show a number of examples for further clarification of these algorithms.

2.1 Elimination of ε-transitions

A nondeterministic finite automata can contain such transitions, for which no input symbol is read on an automaton transition. Such transitions are called ε-transitions.

These transitions complicate some automaton operations, they are unwelcome when implementing the automaton and as the most important thing, they have to be eliminated during transformation of nondeterministic automaton into deterministic one (see Section 2.3).

Any nondeterministic finite automaton with ε-transitions has an equivalent nondeterministic finite automaton without ε-transitions. We will present a simple algorithm for elimination of these ε-transitions.

Example 2.1
Let M be a nondeterministic finite automaton with ε-transitions:
$M = (\{q_0, q_1, q_2, q_3\}, \{a, b\}, \delta, q_0, \{q_1, q_2, q_3\})$, where mapping δ is defined as:

$$\begin{array}{ll} \delta(q_0, \varepsilon) = \{q_1, q_2\}, & \\ \delta(q_0, a) = \{q_1\}, & \\ \delta(q_1, b) = \{q_2\}, & \\ \delta(q_2, a) = \{q_3\}, & \\ \delta(q, x) = \emptyset & \text{in all other cases.} \end{array}$$

Transition diagram of this automaton is given in Figure 2.1. □

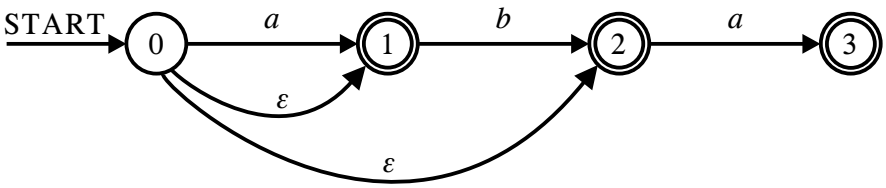

Figure 2.1: Transition diagram of the nondeterministic finite automaton with ε-transitions from Example 2.1

To be able to compute the equivalent nondeterministic finite automaton without ε-transitions we use function $\varepsilon{-}CLOSURE(q)$. (See Def. 1.21) Value of this function is the set of states that can be reached from the state q without reading any input symbol.

Example 2.2
For an automaton from Example 2.1 is:

$$\begin{array}{lcl} \varepsilon{-}CLOSURE(q_0) & = & \{q_0, q_1, q_2\}, \\ \varepsilon{-}CLOSURE(q_1) & = & \{q_1\}, \\ \varepsilon{-}CLOSURE(q_2) & = & \{q_2\}, \\ \varepsilon{-}CLOSURE(q_3) & = & \{q_3\}. \end{array}$$

□

We can now formulate the algorithm for elimination of ε-transitions.

Algorithm 2.3
Construction of a nondeterministic finite automaton without ε-transitions equivalent to nondeterministic finite automaton with ε-transitions.
Input: Finite automaton $M = (Q, A, \delta, q_0, F)$ with ε-transitions.
Output: Finite automaton $M' = (Q, A, \delta', q_0, F')$ without ε-transitions equivalent to M.
Method:

1. $\delta'(q, a) = \bigcup_{p \in \varepsilon - CLOSURE(q)} \delta(p, a)$.

2. $F' = \{q : \varepsilon - CLOSURE(q) \cap F \neq \emptyset, q \in Q\}$.

□

Example 2.4
Let us construct the equivalent nondeterministic finite automaton without ε-transitions for the finite automaton from Example 2.1. We use already computed function $\varepsilon - CLOSURE$ from Example 2.2. We are then able to construct automaton $M = (Q, A, \delta', q_0, F')$, where:

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

$F' = \{q_0, q_1, q_2, q_3\}$, because

$\varepsilon - CLOSURE(q_0) \cap F = \{q_1, q_2\}$,
$\varepsilon - CLOSURE(q_1) \cap F = \{q_1\}$,
$\varepsilon - CLOSURE(q_2) \cap F = \{q_2\}$,
$\varepsilon - CLOSURE(q_3) \cap F = \{q_3\}$.

The transition diagram of resulting automaton is given in Figure 2.2. □

Example 2.5
Construct a nondeterministic finite automaton without ε-transitions equivalent to the automaton having transition diagram depicted in Figure 2.3.

$\varepsilon - CLOSURE$ function for nontrivial cases is:

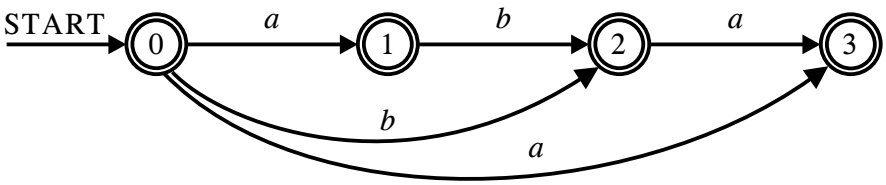

Figure 2.2: Transition diagram of the resulting nondeterministic finite automaton from Example 2.4

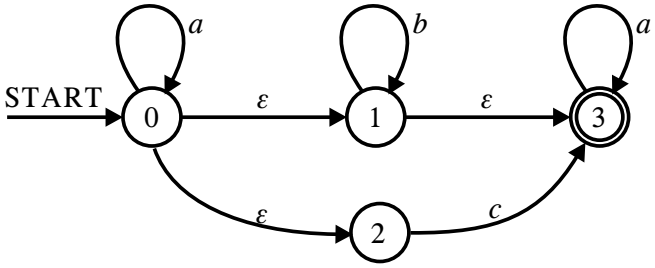

Figure 2.3: Transition diagram of the nondeterministic finite automaton with ε-transitions from Example 2.5

$\varepsilon\text{–}CLOSURE(q_0) = \{q_0, q_1, q_2, q_3\}$,
$\varepsilon\text{–}CLOSURE(q_1) = \{q_1, q_3\}$.
Now we are able to construct resulting automaton $M = (Q, A, \delta', q_0, F')$, where:

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

Transition diagram of the resulting automaton is given in Figure 2.4. Notice, that state q_2 is not accessible and can be removed from the automaton. □

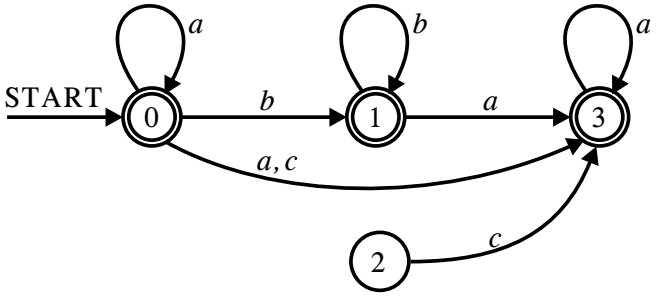

Figure 2.4: Transition diagram of the nondeterministic finite automaton without ε-transitions from Example 2.5

2.2 Elimination of multiple initial states

A nondeterministic finite automaton can be defined with a single initial state or with a finite set of initial states. For every automaton with the set of initial states there exists an equivalent automaton with a single initial state. This Section shows how to construct such an equivalent automaton.

It is shown in Figure 2.5 an example of an automaton with three initial states. As you can see, states q_0, q_1 and q_2 are all initial states. Next example (Figure 2.6) shows, that automaton with more initial states does not have to be necessarily connected.

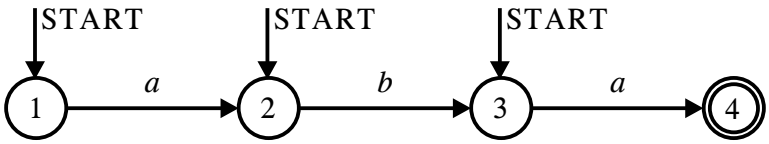

Figure 2.5: Transition diagram of the nondeterministic finite automaton with more initial states

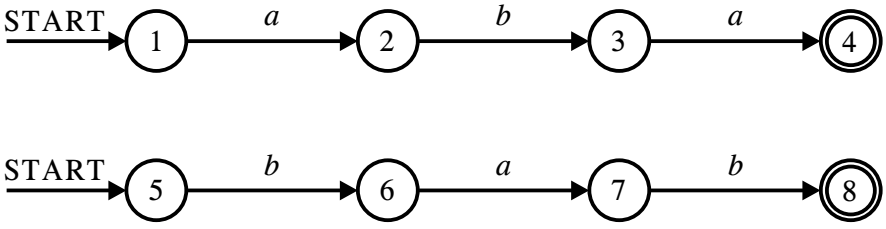

Figure 2.6: Transition diagram of the nondeterministic finite automaton with more initial states

Algorithm 2.6
Construction of a nondeterministic finite automaton with single initial state equivalent to nondeterministic finite automaton with several initial states.
Input: Finite automaton $M = (Q, A, \delta, I, F)$ with nonempty set I.
Output: Finite automaton $M' = (Q', A, \delta', q_0, F)$ with a single initial state q_0.
Method: Automaton M' will be constructed using following two steps:

1. $Q' = Q \cup \{q_0\},\ q_0 \notin Q$,
2. $\delta'(q_0, \varepsilon) = I$,
   $\delta'(q, a) = \delta(q, a)$ for all $q \in Q$ and all $a \in A$. □

Algorithm is very straightforward. It just adds a new state and makes ε-transitions to all former initial states. If we do not want to get an automaton with ε-transitions, we have to eliminate these using Algorithm 2.3.

Example 2.7
Construct an equivalent nondeterministic automaton with a single initial state to the automaton having transition diagram depicted in Figure 2.5. Transition diagram of the resulting automaton is depicted in Figure 2.7.

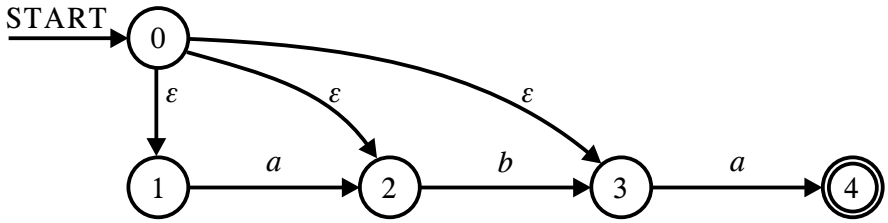

Figure 2.7: Transition diagram of the nondeterministic finite automaton with a single initial state from Example 2.7

Example 2.8
Construct an equivalent nondeterministic automaton with a single initial state to the automaton having transition diagram depicted in Figure 2.6. Transition diagram of the resulting automaton is depicted in Figure 2.8.

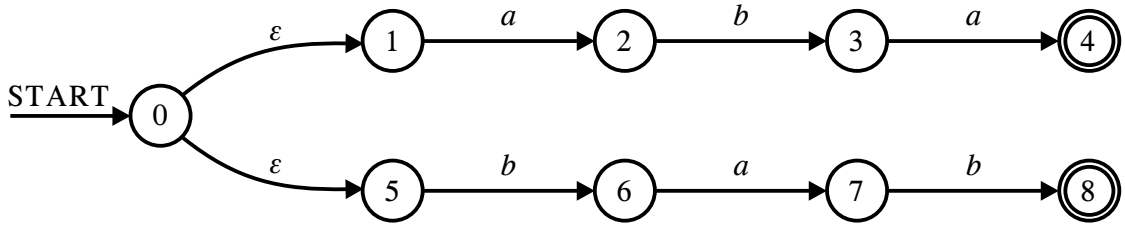

Figure 2.8: Transition diagram of the nondeterministic finite automaton with a single initial state from Example 2.8

2.3 Transformation of nondeterministic finite automaton to deterministic finite automaton

A deterministic automaton can be understood as a special case of a nondeterministic finite automaton. From this follows that each language accepted by a deterministic finite automaton is also accepted by a nondeterministic finite automaton. The fact, that for each language accepted by a nondeterministic finite automaton a deterministic finite automaton accepting the same language can be found, is one of the most important results from the theory of finite automata.

Algorithm 2.9
Transformation of a nondeterministic finite automaton to a deterministic finite automaton.
Input: Nondeterministic finite automaton $M = (Q, A, \delta, q_0, F)$.
Output: Deterministic finite automaton $M' = (Q', A, \delta', q_0', F')$ such that
$L(M) = L(M')$.
Method:

1. The set $Q' = \{\{q_0\}\}$ will be defined, the state $q = \{q_0\}$ will be treated as unmarked. (Please notice, that each state of a deterministic automaton consists of a set of states of a nondeterministic automaton.)
2. If each state in Q' is marked then continue with step 4.
3. An unmarked state q' will be chosen from Q' and the following operations will be executed:
   (a) $\delta'(q', a) = \bigcup \delta(p, a)$ for $p \in q'$ and for all $a \in A$,
   (b) $Q' = Q' \cup \delta'(q', a)$ for all $a \in A$,
   (c) the state $q' \in Q'$ will be marked,
   (d) continue with step 2.
4. $q_0' = \{q_0\}$.
5. $F' = \{q' : q' \in Q', q' \cap F \neq \emptyset\}$. □

Note: Let us mention, that all states of the resulting deterministic finite automaton M' are accessible states. (See Def. 1.23)

A deterministic finite automaton resulting from this algorithm can be seen as a "parallel" simulator of all possible sequences of moves of the nondeterministic finite automaton. States of the deterministic automaton are sets of states of the nondeterministic automaton which can be reached for at least one input string.

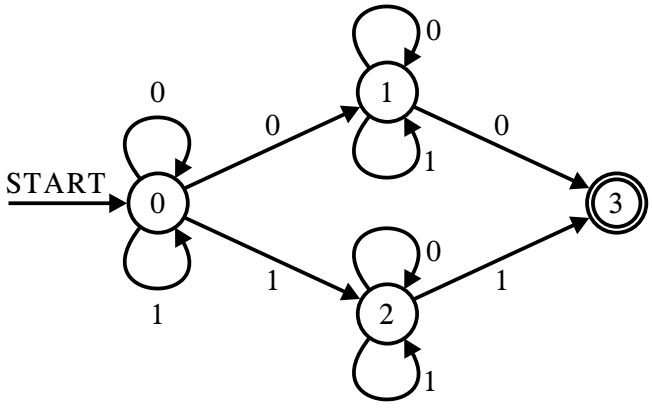

Figure 2.9: Transition diagram of the nondeterministic finite automaton from Example 2.10

Example 2.10
Construct a deterministic finite automaton equivalent to the nondeterministic finite automaton having transition diagram given in Figure 2.9. The transition table of this automaton is of the form:

δ	0	1
q_0	$\{q_0, q_1\}$	$\{q_0, q_2\}$
q_1	$\{q_1, q_3\}$	$\{q_1\}$
q_2	$\{q_2\}$	$\{q_2, q_3\}$
q_3	$\emptyset$	$\emptyset$

We start with the initial state q_0 and create a adequate line in the transition table of resultant automaton. As we see, we have defined two new states named $q_{0,1}$ and $q_{0,2}$. For these states we add new lines to the table and follow computing the line number two. It states the transitions for the state $q_{0,1}$. Because of:

$$\delta'(q_{0,1}, 0) = \delta(q_0, 0) \ \cup \ \delta(q_1, 0) = \{q_0, q_1, q_3\},$$
$$\delta'(q_{0,1}, 1) = \delta(q_0, 1) \ \cup \ \delta(q_1, 1) = \{q_0, q_1, q_2\}$$

we create new two states named $q_{0,1,3}$ and $q_{0,1,2}$. We follow this approach until we receive the resulting transition table:

δ	0	1
q_0	$q_{0,1}$	$q_{0,2}$
$q_{0,1}$	$q_{0,1,3}$	$q_{0,1,2}$
$q_{0,2}$	$q_{0,1,2}$	$q_{0,2,3}$
$q_{0,1,3}$	$q_{0,1,3}$	$q_{0,1,2}$
$q_{0,2,3}$	$q_{0,1,2}$	$q_{0,2,3}$
$q_{0,1,2}$	$q_{0,1,2,3}$	$q_{0,1,2,3}$
$q_{0,1,2,3}$	$q_{0,1,2,3}$	$q_{0,1,2,3}$

States accessible from the initial state are listed in this table only. An accessible state is such a state, into which an automaton can get from the initial state by performing some sequence of moves. (See Def. 1.23)

The states of resulting deterministic automaton are the sets of the original nondeterministic automaton. Final states of DFA are those states, that contain at least one final state of NFA. In our case state q_3 is a final state in NFA, so states $q_{0,1,3}$, $q_{0,2,3}$, $q_{0,1,2,3}$ are final states in the resulting DFA.

Transition diagram of the automaton for this transition table is depicted in Figure 2.10.

Example 2.11
Construct a deterministic finite automaton for the nondeterministic finite automaton having transition diagram given in Figure 2.11.

We use the same approach as in the previous example. We start with the initial state q_0. Transition function in NFA for state q_0 is $\delta(q_0, 0) = \{q_0, q_1\}$ so transition function for state q_0 in DFA is $\delta'(q_0, 0) = \{q_{0,1}\}$. The same applies for $\delta(q_0, 1) = \emptyset$ which gives us $\delta'(q_0, 1) = \emptyset$. We then compute the transition function for the newly created state $q_{0,1}$:

$$\delta'(q_{0,1}, 0) = \delta(q_0, 0) \ \cup \ \delta(q_1, 0) = \{q_0, q_1\},$$
$$\delta'(q_{0,1}, 1) = \delta(q_0, 1) \ \cup \ \delta(q_1, 1) = \{q_1, q_2\}.$$

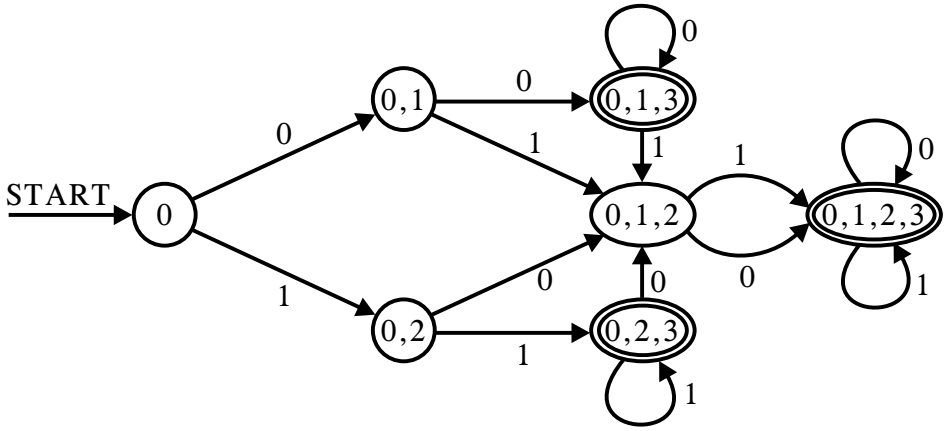

Figure 2.10: Transition diagram of the deterministic finite automaton from Example 2.10

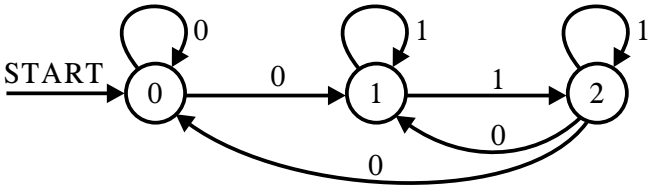

Figure 2.11: Transition diagram of the nondeterministic finite automaton from Example 2.11

A new state $q_{1,2}$ was created during this process so we also have to compute transition function for this state:

$$\delta'(q_{1,2}, 0) = \delta(q_1, 0) \ \cup \ \delta(q_2, 0) = \{q_0, q_1\},$$
$$\delta'(q_{1,2}, 1) = \delta(q_1, 1) \ \cup \ \delta(q_2, 1) = \{q_0, q_1\}.$$

Final state in nondeterministic finite automaton was state q_2, so all states of deterministic finite automaton containing this state are final. In this example it is the only state $q_{1,2}$.

The final transition function is shown in the following table and the transition diagram of the resulting automaton is depicted in Figure 2.12.

δ	0	1
q_0	$q_{0,1}$	$\emptyset$
$q_{0,1}$	$q_{0,1}$	$q_{1,2}$
$q_{1,2}$	$q_{0,1}$	$q_{0,1}$

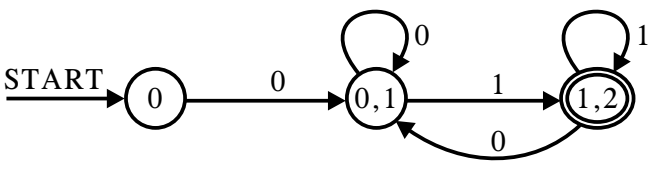

Figure 2.12: Transition diagram of the deterministic finite automaton from Example 2.11

2.4 Construction of complete finite automaton equivalent to given finite automaton

Algorithm 2.12
Making a finite automaton complete.
Input: Finite automaton $M = (Q, T, \delta, q_0, F)$.
Output: Complete finite automaton $M' = (Q', T, \delta', q_0, F)$ such that $L(M') = L(M)$.
Method:

1. Create a new state $q_\emptyset \notin Q$ which will be called sink state. $Q' = Q \cup \{q_\emptyset\}$.
2. If $\delta(q, a)$ in not defined for any pair $q \in Q'$, $a \in T$ then we define $\delta'(q, a) = q_\emptyset$ for such pair.
3. For all other cases $\delta'(q, a) = \delta(q, a)$. □

Example 2.13
Let a finite automaton be $M = (\{q_0, q_1, q_2\}, \{a, b, c\}, \delta, q_0, \{q_0, q_1, q_2\})$, where mapping δ is given by the table:

δ	a	b	c
q_0	q_0	q_1	q_2
q_1		q_1	q_2
q_2			q_2

The mapping δ is not defined for some cases. The complete finite automaton has the following transition table:

	a	b	c
q_0	q_0	q_1	q_2
q_1	$q_\emptyset$	q_1	q_2
q_2	$q_\emptyset$	$q_\emptyset$	q_2
$q_\emptyset$	$q_\emptyset$	$q_\emptyset$	$q_\emptyset$

□

2.5 Minimisation of set of states of a finite automaton

The construction of a finite automaton may lead to the situation that the resulting automaton can contain equivalent states. In this section, an approach is shown which leads to the construction of finite automaton where equivalent states are replaced so that the resulting automaton will be equivalent to the original automaton.

Example 2.14
Let finite automaton be $M = (\{A, B, C, D, E, F\}, \{a, b\}, \delta, A, \{A\})$, where mapping δ is shown in Fig. 2.13 in the form of transition diagram and transition table. It is matter of fact that states C and E can be replaced by single state CE with appropriate reconstruction of mapping δ.

δ	a	b
A	B	C
B	D	A
C	E	F
D	A	C
E	C	F
F	A	D

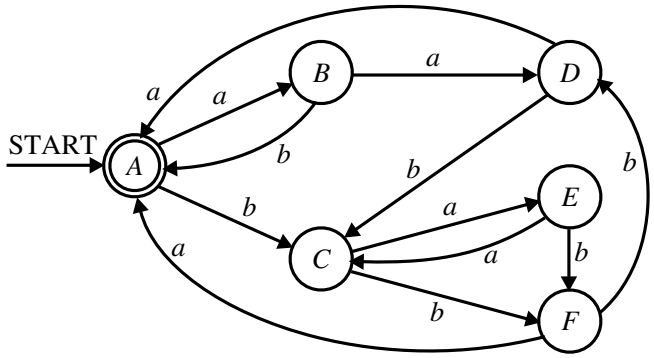

Figure 2.13: Transition table and transition diagram of the finite automaton from Example 2.14

Definition 2.15
Let a finite automaton be $M = (Q, T, \delta, q_0, F)$. States $q_i, q_j \in Q$ of automaton M are *equivalent* when all possible sequences of transitions starting in states q_i and q_j for every string of input symbols are leading either to some final state or to some nonfinal state. More formally: It holds for every string $x \in T^*$ which can be read by finite automaton M:

$$(q_i, x) \vdash \ldots \vdash (q_1, \varepsilon),$$
$$(q_j, x) \vdash \ldots \vdash (q_2, \varepsilon)$$

and either $q_1, q_2 \in F$ or $q_1, q_2 \notin F$.
If states q_i and q_j are not equivalent then they are *distinguishable*.

A method of minimisation of a finite automaton consists in finding of classes of equivalent states and replacing of each such class by a single state. The method shown here is applicable for complete automata. The method proceeds backwards. At first, it distinguishes states into two classes: class of final states and class of nonfinal states. In the next steps, it distinguishes states inside of every class. This is done according to the existence of transition for some symbol to different classes of states.

Algorithm 2.16
Minimisation of deterministic complete finite automaton.
Input: Deterministic complete finite automaton $M = (Q, T, \delta, q_0, F)$.
Output: Finite automaton M_{min} equivalent to the automaton M which have minimum number of states.
Method:

1. Create two classes of states: $Q_{1,1} = F$, $Q_{1,2} = Q \setminus F$. Set $k = 2$.

2. Create a transition table for all states $q \in Q$. The classes of states will be used instead of states.
3. Divide every class $Q_{k,x}$ into classes $Q_{k+1,y1}, Q_{k+1,y2}, \ldots, Q_{k+1,yn}$ so that states in one class have equal rows in transition table.
4. Set $k := k + 1$.
5. Repeat steps 2, 3 and 4 as long as the number of classes is growing. Stop this process in the moment when $Q_{k,x} = Q_{k-1,x}$ for all x.
6. Resulting automaton will contain a single state for every class.

Example 2.17
Complete finite automaton M is given in Example 2.14. Construct equivalent deterministic finite automaton having minimum number of states. We proceed in this way:
Step 1: Divide states of automaton M into two classes: $Q_{1,1} = \{A\}$, $Q_{1,2} = \{B, C, D, E, F\}$. State A is the final state, other states are nonfinal ones. This division is shown in the last column of Table 2.1.

δ	a	b	$Q_{1,x}$
A	B	C	1
B	D	A	2
C	E	F	2
D	A	C	2
E	C	F	2
F	A	D	2

Table 2.1: Transition table after the first step of minimisation of finite automaton from Example 2.17

Step 2: The transition table of automaton M we change in such way that we replace states inside the table by the number of class to which the state belongs. The result is shown in the Table 2.2 a).

$Q_{1,x}$	δ	a	b	$Q_{2,x}$
1	A	2	2	1
2	B	2	1	2
2	C	2	2	3
2	D	1	2	4
2	E	2	2	3
2	F	1	2	4

a)

$Q_{2,x}$	δ	a	b	$Q_{3,x}$
1	A	2	3	1
2	B	4	1	2
3	C	3	4	3
3	E	3	4	3
4	D	1	3	4
4	F	1	4	5

b)

Table 2.2: Transition tables during minimisation of finite automaton from Example 2.17

Step 3: $k = 2$. The new set of classes is created according to transitions for symbols a and b. The class 1 cannot be divided and therefore: $Q_{2,1} = \{A\}$. The class 2 is divided into three new classes: $Q_{2,2} = \{B\}, Q_{2,3} = \{C, E\}, Q_{2,4} = \{D, F\}$. The resulting division is shown in the last column of Table 2.2a).

Step 4: Repeat steps 2 and 3 for classes $Q_{2,x}$. The resulting transition table is shown in Table 2.2b). The iteration of classes $Q_{3,x}$ is performed. It is not necessary to divide classes $Q_{2,1}, Q_{2,2}$ and $Q_{2,3}$. Therefore $Q_{3,1} = Q_{2,1}$, $Q_{3,2} = Q_{2,2}$ and $Q_{3,3} = Q_{2,3}$. The class $Q_{2,4}$ must be divided.The classes $Q_{3,4} = \{D\}$ and $Q_{3,5} = \{F\}$ are result of this division. The result is shown again in the last column of Table 2.2b). Next iteration is performed for classes $Q_{3,x}$. The result is shown in Table 2.3a). The number of classes is 5.

$Q_{3,x}$	δ	a	b
1	A	2	3
2	B	4	1
3	C	3	5
3	E	3	5
4	D	1	3
5	F	1	4

$a)$

δ	a	b
1	2	3
2	4	1
3	3	5
4	1	3
5	1	4

$b)$

Table 2.3: Transition tables during minimisation of finite automaton from Example 2.17, steps 4 and 5

Step 5: The resulting finite automaton M_{min} is constructed in this way: A single state of finite automaton is created for every class $Q_{3,x}$. The transition table of the minimised finite automaton is shown in Table 2.3b).

3 Operations with finite automata

3.1 Union of finite automata

The languages accepted by finite automata are regular languages. Set operations union, intersection defined for these regular languages. So it is possible to compute the union of languages, intersection of languages and so on. Therefore we can also construct finite automata for languages, which are the results of the operations of union, and intersection.

This section presents a construction of finite automata for union of two languages. The algorithm is based on insertion of new initial state with ε-transitions to all former initial states.

Algorithm 3.1
Construction of finite automaton for union of languages.
Input: Two finite automata M_1 and M_2.
Output: Finite automaton M accepting the language $L(M) = L(M_1) \cup L(M_2)$.
Method:

1. Let $M_1 = (Q_1, A, \delta_1, q_{01}, F_1),\ M_2 = (Q_2, A, \delta_2, q_{02}, F_2)$.
2. Resulting automaton $M = (Q, A, \delta, q_0, F)$ is constructed using following steps:
   (a) $Q = Q_1 \cup Q_2 \cup \{q_0\},\ q_0 \notin Q_1 \cup Q_2$,
   (b) $\delta(q_0, \varepsilon) = \{q_{01}, q_{02}\}$,
   $\delta(q, a) = \delta_1(q, a)$ for all $q \in Q_1$ and all $a \in A$,
   $\delta(q, a) = \delta_2(q, a)$ for all $q \in Q_2$ and all $a \in A$.
3. $F = F_1 \cup F_2$. □

As you can notice, this algorithm is very similar to the algorithm for elimination of multiple initial states 2.6. Especially Example 2.8 can be seen as union of two automata (if we consider each connected part of the mentioned automaton to be a separated automaton).

Example 3.2
For finite automata M_1 and M_2 having transition diagrams depicted in Fig. 3.1 we construct automaton M using Algorithm 3.1. Its transition diagram is depicted in Fig. 3.2 □

If we need to obtain a deterministic finite automaton accepting the union of languages, we have to eliminate the ε-transitions and construct an equivalent deterministic automaton by using Algorithms 2.3 and 2.9.

3.2 Intersection of finite automata

This section presents a construction of finite automata for intersection of two languages. The algorithm presented here generates only the accessible states of resulting automaton.

Algorithm 3.3
Construction of finite automaton for intersection of two languages.
Input: Two finite automata $M_1 = (Q_1, A, \delta_1, q_{01}, F_1), M_2 = (Q_2, A, \delta_2, q_{02}, F_2)$.
Output: Finite automaton $M = (Q, A, \delta, q_0, F)$, accepting language $L(M) = L(M_1) \cap L(M_2)$.
Method:

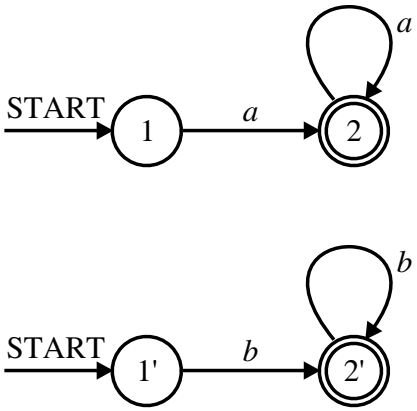

Figure 3.1: Transition diagrams of finite automata M_1 and M_2 accepting languages a^+ and b^+, respectively, from Example 3.2

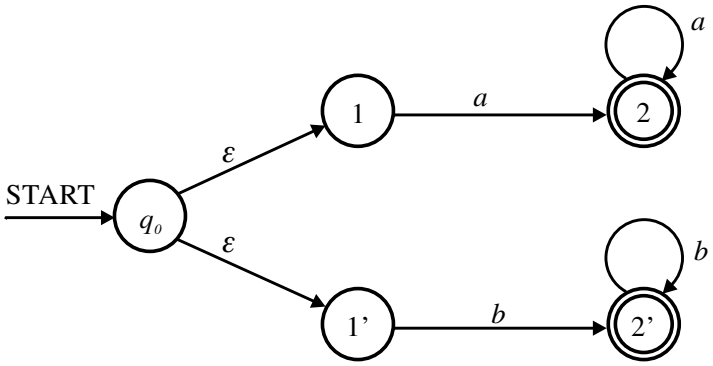

Figure 3.2: Transition diagram of finite automaton M accepting language $a^+ \cup b^+$ from Example 3.2

1. Let $Q = \{(q_{01}, q_{02})\}$. A state (q_{01}, q_{02}) will be treated as unmarked.
2. If all states in Q are marked go to step 4.
3. Take any unmarked state $q = (q_{n1}, q_{m2})$ from Q and perform these operations:
   (a) determine $\delta((q_{n1}, q_{m2}), a) = (\delta_1(q_{n1}, a), \delta_2(q_{m2}, a))$ for all $a \in A$,
   (b) if both transitions $\delta_1(q_{n1}, a)$ and $\delta_2(q_{m2}, a)$ are defined then $Q = Q \cup (\delta_1(q_{n1}, a), \delta_2(a_{m2}, a))$ and state $(\delta_1(q_{n1}, a), \delta_2(q_{m2}, a))$ will be treated as unmarked only if it is a new state in Q,
   (c) state (q_{n1}, q_{m2}) in Q will be treated as marked,
   (d) go to step 2.
4. $q_0 = (q_{01}, q_{02})$.
5. $F = \{q : q \in Q,\ q = (q_{n1}, q_{m2}),\ q_{n1} \in F,\ q_{m2} \in F\}$. □

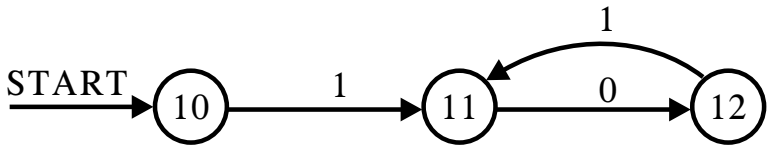

Figure 3.3: Transition diagram of the finite automaton M_1 from Example 3.4

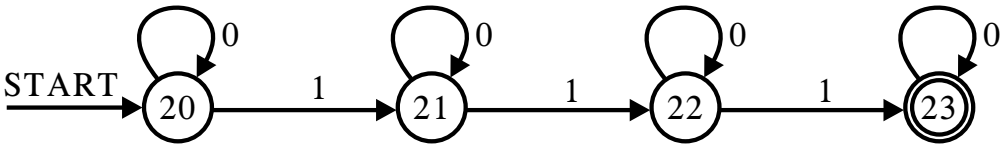

Figure 3.4: Transition diagram of the finite automaton M_2 from Example 3.4

Example 3.4
Construct a finite automaton M for intersection of two languages $L(M) = L(M_1) \cap L(M_2)$. Transition diagram of the automata M_1 and M_2 are depicted in Figures 3.3 and 3.4.

We start with the state (q_{10}, q_{20}). There are transitions from states q_{10} and q_{20} for symbol 1 defined in both of these automata: $\delta(q_{10}, 1) = \{q_{11}\}$ and $\delta(q_{20}, 1) = \{q_{21}\}$ so the new transition in resulting automaton is $\delta'(q_{10,20}) = \{q_{11,21}\}$. Transition for symbol 0 is not defined in both of the automata, so it will not be defined in the resulting automaton either. Then we make transitions from state $q_{11,21}$. Transitions for symbol are 1 are not defined, but transitions for symbol 0 are: $\delta(q_{11}, 0) = \{q_{12}\}$ and $\delta(q_{21}, 0) = \{q_{21}\}$. Therefore new transition is $\delta'(q_{11,21}, 0) = \{q_{12,21}\}$. We make transitions from the state $q_{12,21}$ and so on. The resulting automaton has the transition diagram depicted in Figure 3.5. As you can notice, the resulting automaton accepts language of only one string $\{101010\}$. This string is the only string accepted by both automata M_1 and M_2. □

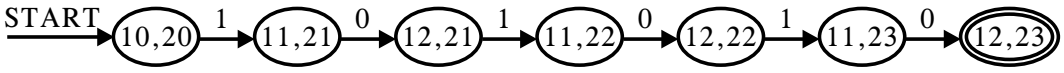

Figure 3.5: Transition diagram of the finite automaton M from Example 3.4

Example 3.5
Construct a finite automaton M for intersection of two languages $L(M) = L(M_1) \cap L(M_2)$. Automata M_1 (accepting the language of binary numbers divisible by 2) and M_2 (accepting the language of binary numbers divisible by 3) have transition diagrams depicted in Figures 3.6 and 3.7. The resulting automaton accepting the language of binary numbers divisible by 6 is depicted in Figure 3.8. □

3.3 Regular expressions and finite automata

There are several ways of construction of an equivalent finite automata for a given regular expressions. In this section we show the method called 'method of neighbours'.

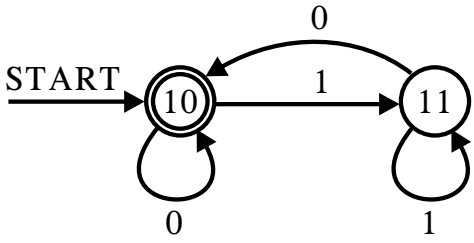

Figure 3.6: Transition diagram of the finite automaton M_1 from Example 3.5

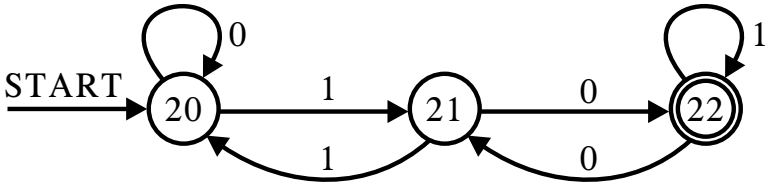

Figure 3.7: Transition diagram of the finite automaton M_2 from Example 3.5

Algorithm 3.6
Construction of an equivalent finite automaton for a given regular expression.
Input: Regular expression V.
Output: Finite automaton $M = (Q, A, \delta, q_0, F)$ such, that $h(V) = L(M)$.
Method: Let us have a regular expression V over alphabet A.

1. Let us number $1, 2, ..., n$ all occurrences of symbols from A in expression V so that every two occurrences of the same symbol are numbered with different numbers. Let us name the resulting regular expression V'.
2. Let us construct the set of initial symbols $Z = \{x_i : x \in A$, symbol x_i is the initial symbol of some string from $h(V')\}$.
3. Let us construct set of neighbors $P = \{x_i y_j$: symbols x_i and y_j occur next to each other in some string from $h(V')\}$.
4. Let us construct the set of final states $F = \{x_i$: symbol x_i is the last symbol of some

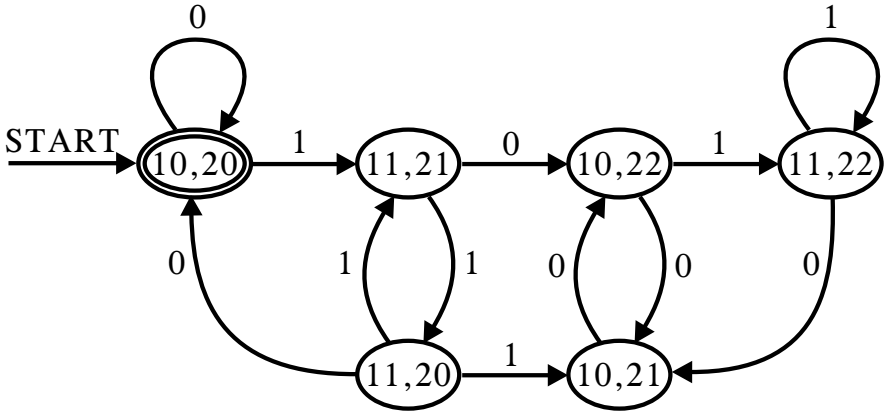

Figure 3.8: Transition diagram of the finite automaton M from Example 3.5

string from $h(V)\} \cup \{q_0 : \varepsilon \in h(V)\}$.

5. Let us construct the set of states of finite automaton $Q = \{q_0\} \cup \{x_i : x \in A, i \in <1, n>\}$.
6. Function δ is defined as follows:
   (a) $\delta(q_0, x)$ contains x_i for all $x_i \in Z$ such, that x_i was created by numbering x.
   (b) $\delta(x_i, y)$ contains y_j for all couples $x_i y_j \in P$ such that y_j was created by numbering y.
7. Set F is the set of final states. □

Example 3.7
Let us have a regular expression $V = (a+b)^*ab(a+b)^*$. Let us construct the equivalent finite automaton M:
$V' = (a_1 + b_2)^* a_3 b_4 (a_5 + b_6)^*$,
$Z = \{a_1, b_2, a_3\}$,
$P = \{a_1a_1, a_1b_2, a_1a_3, b_2a_1, b_2b_2, b_2a_3, a_3b_4, b_4a_5, b_4b_6, a_5a_5, a_5b_6, b_6a_5, b_6b_6\}$,
$F = \{b_4, a_5, b_6\}$.
The initial state of automaton is state q_0, the set of final states is F and the transition function is defined as follows:
$\delta(q_0, x)$ contains x_i for all $x_i \in Z$ such, that x_i was created by numbering x,
$\delta(x_i, y)$ contains y_j for all couples $x_i y_j \in P$ such, that y_j was created by numbering y.

For given regular expression we then construct the finite automaton
$M_1 = (\{q_0, a_1, b_2, a_3, b_4, a_5, b_6\}, \{a, b\}, \delta, q_0, \{b_4, a_5, b_6\})$, where function δ is defined by the following transition table:

δ	a	b
q_0	$\{a_1, a_3\}$	$\{b_2\}$
a_1	$\{a_1, a_3\}$	$\{b_2\}$
b_2	$\{a_1, a_3\}$	$\{b_2\}$
a_3	$\emptyset$	$\{b_4\}$
b_4	$\{a_5\}$	$\{b_6\}$
a_5	$\{a_5\}$	$\{b_6\}$
b_6	$\{a_5\}$	$\{b_6\}$

□

4 Construction of string matching automata for exact matching

Exercise 4.1
Create an exact string matching automaton for string *abab* over an alphabet $A = \{a, b\}$.

Solution of Exercise 4.1 is a deterministic automaton accepting language A^*abab, which contains all strings over the alphabet A closed by the suffix *abab*. This automaton can be created by following two steps:

1. Create a nondeterministic version of this automaton as it is shown in Figure 4.1. It is a combination of an automaton that accepts the language A^* and an automaton accepting only one string *abab*.

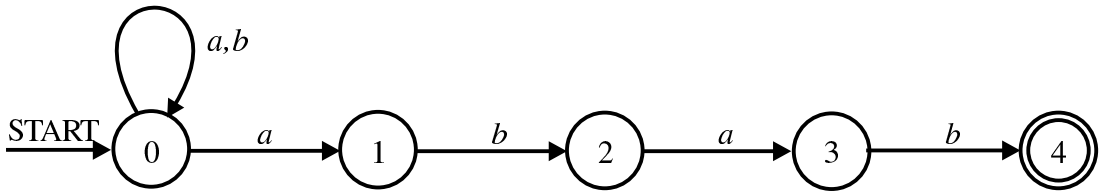

Figure 4.1: Transition diagram of the nondeterministic finite automaton for searching the string *abab* from Exercise 4.1 step 1

2. Transform this automaton to a deterministic one using subset construction. Transition tables of both (nondeterministic and deterministic) automata are described in the following tables:

	a	b
0	0, 1	0
1		2
2	3	
3		4
4		

	a	b
0	01	0
01	01	02
02	013	0
013	01	024
024	013	0

.

Transition diagram of the deterministic automaton is shown in Figure 4.2. □

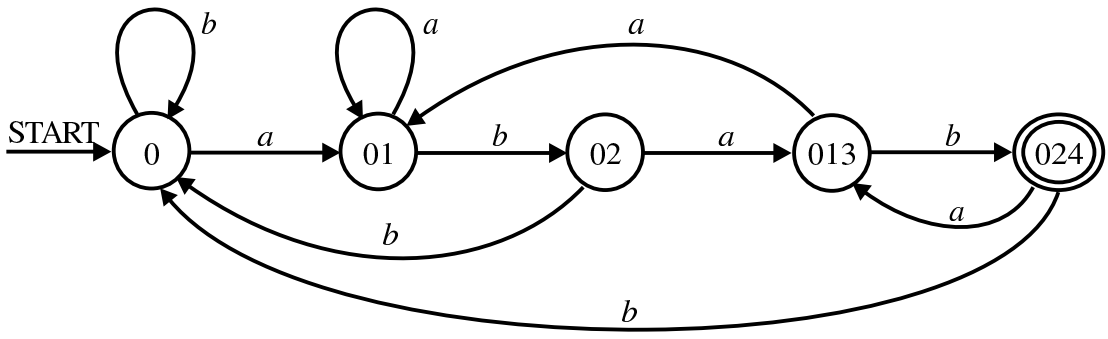

Figure 4.2: Transition diagram of the deterministic finite automaton for searching the string *abab* from Exercise 4.1 step 2

This exercise has shown a general method of the pattern matching automata construction. It is created a nondeterministic automaton that is transformed to a deterministic one using subset construction.

Exercise 4.1 has also shown that the number of states of the deterministic finite automaton is the same as the number of states of an equivalent nondeterministic finite automaton. This proposition holds for exact matching of either one string or finite number of strings.

Exercise 4.2
Create a deterministic finite automaton for searching all nonempty factors of a string *abab*, over an alphabet $A = \{a, b\}$. The method of construction is performed in three steps:

1. Construction of nondeterministic finite automaton. The transition diagram of the nondeterministic finite automaton is shown in Figure 4.3.

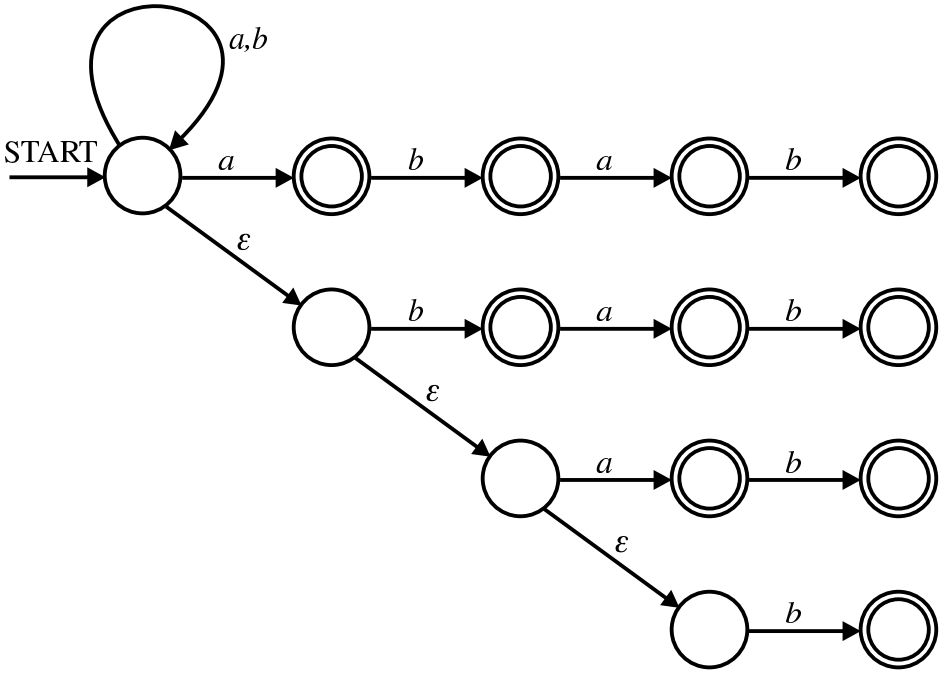

Figure 4.3: Transition diagram of the nondeterministic finite automaton for searching all nonempty factors of the string *abab* from Exercise 4.2 step 1

2. Removal of ε-transitions. Transition diagram of the automaton after removing ε-transitions is shown in Figure 4.4.
3. Determinisation. Transition tables of both (nondeterministic and deterministic) automata are shown in the Table 3. Transition diagram of the deterministic automaton is shown in Figure 4.5.

Exercise 4.3
Create a deterministic finite automaton for searching all strings from the set $P = \{kajak, jak, kaj, aja, ja\}$, over an alphabet $A = \{a, j, k, x\}$.

1. The transition diagram of the nondeterministic automaton is shown in Figure 4.6.

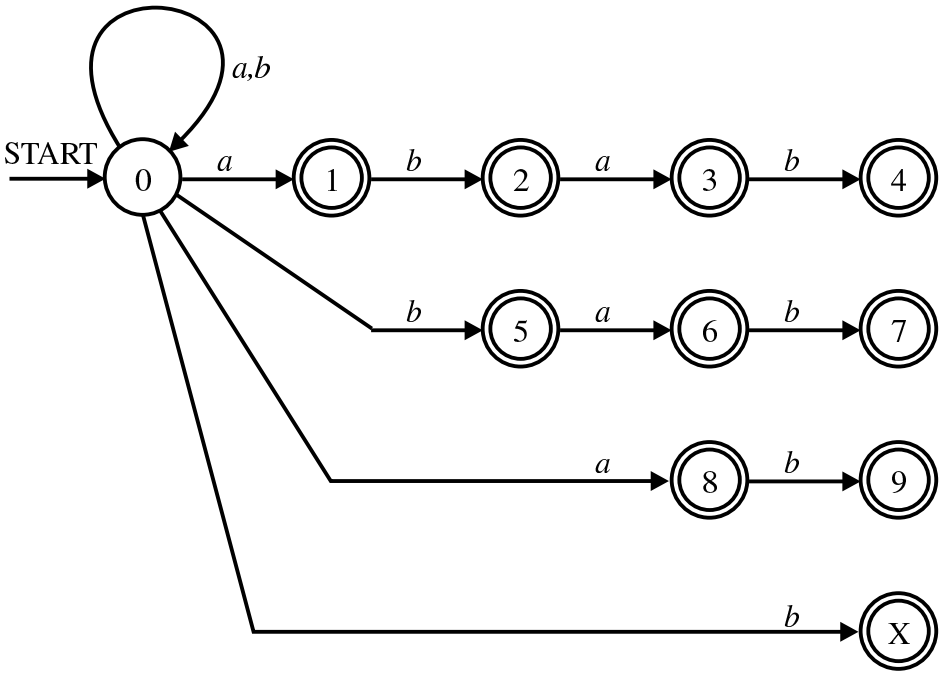

Figure 4.4: Transition diagram of the nondeterministic finite automaton for searching all nonempty factors of the string *abab* after removing ε-transitions from Exercise 4.2 step 2

	a	b
0	$0, 1, 8$	$0, 5, X$
1		2
2	3	
3		4
4		
5	6	
6		7
7		
8		9
9		
X		

	a	b
0	018	05X
018	018	0259X
0259X	01368	05X
01368	018	024579X
024579X	01368	05X
05X	0168	05X
0168	018	02579X
02579X	01368	05X

Table 4.1: Transition tables of both nondeterministic and deterministic finite automata from Exercise 4.2

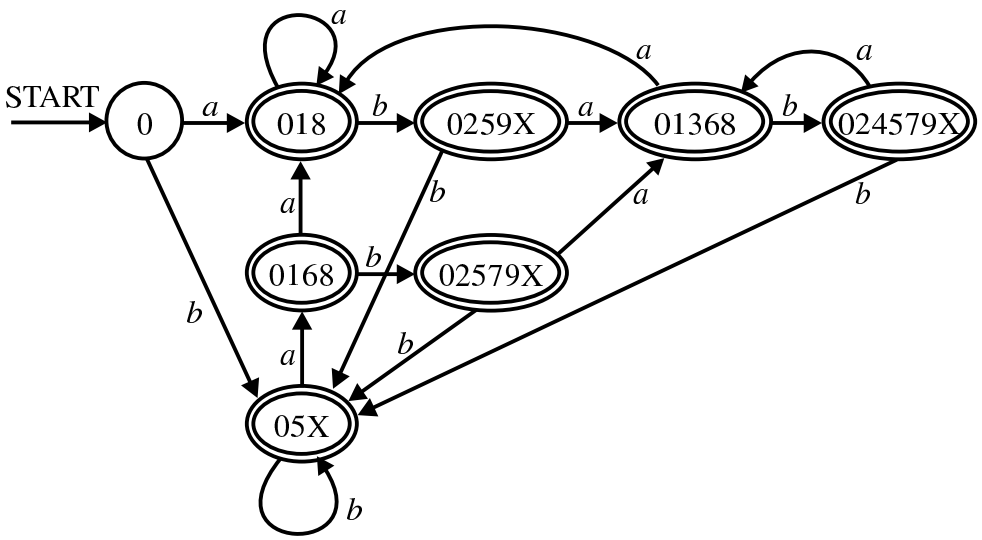

Figure 4.5: Transition diagram of the deterministic finite automaton for searching all non-empty factors of the string *abab* from Exercise 4.2 step 3

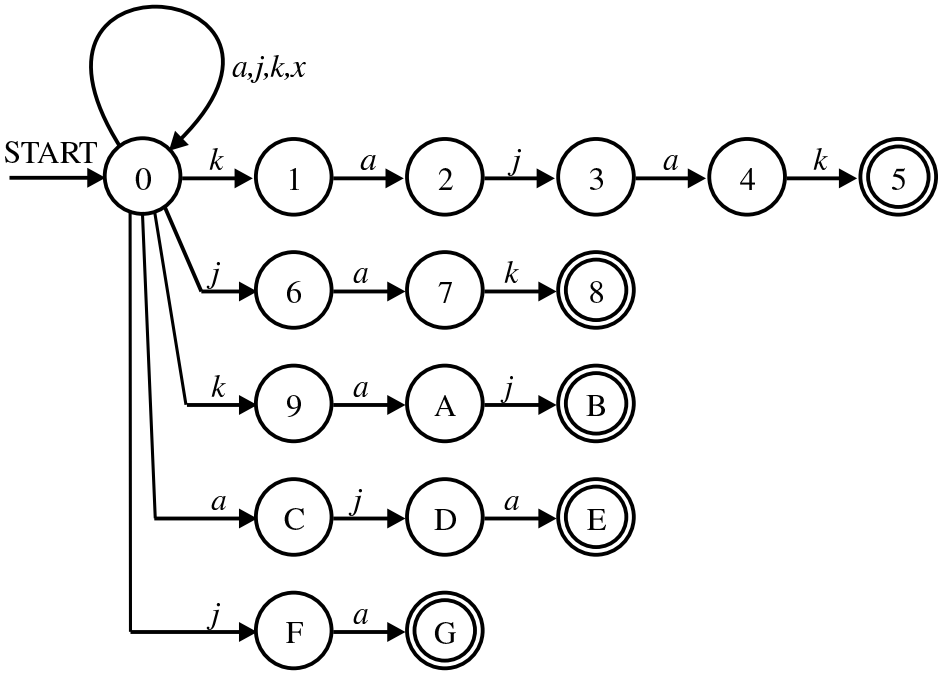

Figure 4.6: Transition diagram of the nondeterministic finite automaton for searching all strings from the set $P = \{kajak, jak, kaj, aja, ja\}$ from Exercise 4.3 step 1

2. Transition tables of both the nondeterministic and the deterministic finite automata are shown in Table 4.2. Transition diagram of this automaton is shown in Figure 4.7.

	a	j	k	x
0	$0, C$	$0, 6, F$	$0, 1, 9$	0
1	2			
2		3		
3	4			
4			5	
5				
6	7			
7			8	
8				
9	A			
A		B		
B				
C		D		
D	E			
E				
F	G			
G				

	a	j	k	x
0	$0C$	$06F$	019	0
019	$02AC$	$06F$	019	0
$02AC$	$0C$	$036BDF$	019	0
$036BDF$	$047CEG$	$06F$	019	0
$047CEG$	$0C$	$06DF$	01589	0
01589	$02AC$	$06F$	019	0
$06F$	$07CG$	$06F$	019	0
$06DF$	$07CEG$	$06F$	019	0
$07CG$	$0C$	$06DF$	0189	0
$07ECG$	$0C$	$06DF$	0189	0
$0C$	$0C$	$06DF$	019	0
0189	$02AC$	$06F$	019	0

Table 4.2: Transition tables of both the nondeterministic and the deterministic finite automata from Exercise 4.3

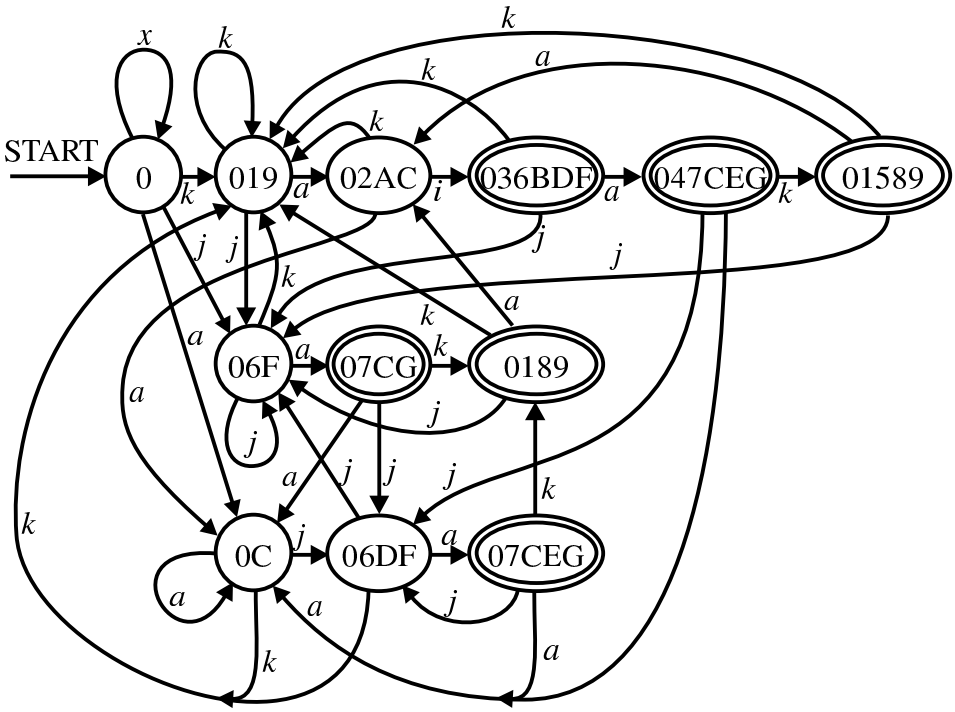

Figure 4.7: Transition diagram of the deterministic finite automaton for searching all strings from the set $P = \{kajak, jak, kaj, aja, ja\}$ from Exercise 4.3 step 2

5 Construction of string matching automata for approximate string matching

Exercise 5.1
Create a deterministic finite automaton for searching a string *abc* using Hamming distance with at most two errors allowed, $k = 2$, over an alphabet $A = \{a, b, c, d\}$.

1. The transition diagram of the nondeterministic automaton is shown in Figure 5.1.
2. Transition tables of both (nondeterministic and deterministic) automata are described in the following tables:

	a	b	c	d
0	$0, 1$	$0, 4$	$0, 4$	$0, 4$
1	5	2	5	5
2	6	6	3	6
3				
4	7	5	7	7
5	8	8	6	8
6				
7			8	
8				

	a	b	c	d
0	01	04	04	04
01	015	024	045	045
015	0158	0248	0456	0458
0158	0158	0248	0456	0458
024	0167	0456	0347	0467
0167	015	024	0458	045
0248	0167	0456	0347	0467
0347	017	045	0478	047
017	015	024	0458	045
045	0178	0458	0467	0478
0178	015	024	0458	045
0456	0178	0458	0467	0478
0458	0178	0458	0467	0478
0467	017	045	0478	047
047	017	045	0478	047
0478	017	045	0478	047
04	017	045	047	047

□

Exercise 5.2
Create a deterministic finite automaton for searching a string *aba* using Levenshtein distance with at most one error allowed, $k = 1$, over an alphabet $A = \{a, b\}$. Method of this automaton construction is nearly the same as the automata construction in previous cases. The only difference is one more step, removing ε-transitions.

1. Create a nondeterministic version of this automaton as it is shown in Figure 5.2.
2. Remove ε-transitions. The automaton after removing ε-transitions is shown in Figure 5.3.
3. Transform this automaton to a deterministic one using subset construction. Transition tables of both (nondeterministic and deterministic) automata are shown in Table 5.1. Transition diagram of the deterministic automaton is shown in Figure 5.4. □

Exercise 5.3
Create a deterministic finite automaton for searching string *aba* using generalized Levenshtein distance with at most one error allowed, $k = 1$, over an alphabet $A = \{a, b\}$.

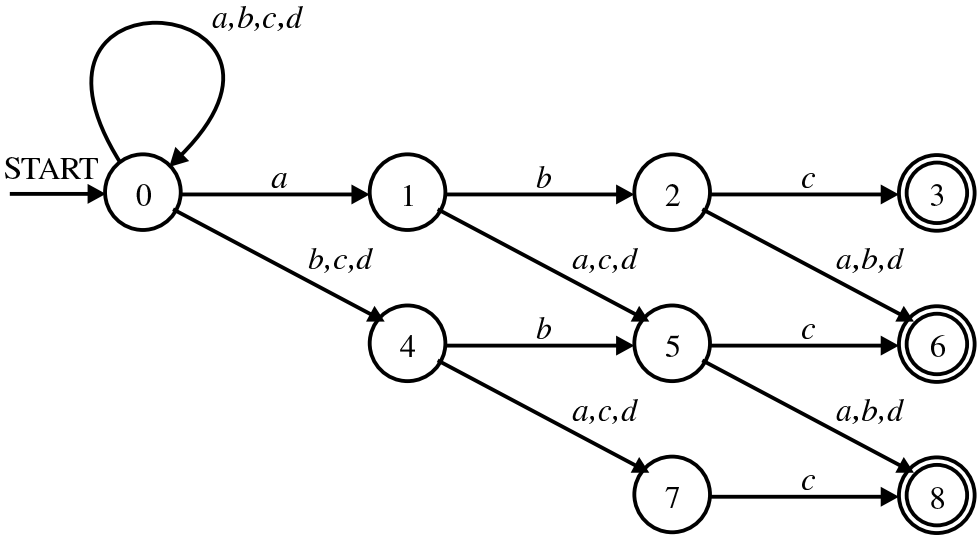

Figure 5.1: Transition diagram of the nondeterministic finite automaton for searching the string *abc* using Hamming distance with at most 2 errors allowed from Exercise 5.1 step 1

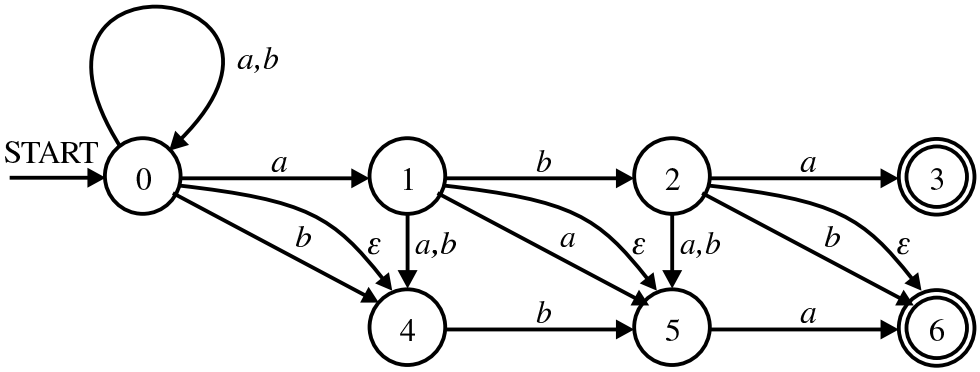

Figure 5.2: Transition diagram of the nondeterministic finite automaton for searching the string *aba* using Levenshtein distance with at most 1 error allowed from Exercise 5.2 step 2

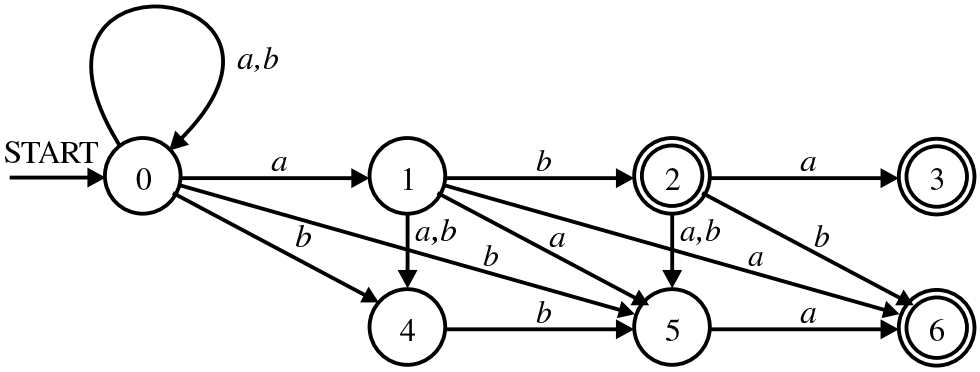

Figure 5.3: Transition diagram of the nondeterministic finite automaton for searching the string *aba* using Levenshtein distance with at most 1 error allowed after removing ε-transitions from Exercise 5.2 step 2

	a	b
0	$0, 1$	$0, 4, 5$
1	$4, 5, 6$	$4, 2$
2	$5, 3$	$5, 6$
3	$\emptyset$	$\emptyset$
4	$\emptyset$	5
5	6	$\emptyset$
6	$\emptyset$	$\emptyset$

	a	b
0	01	045
01	01456	0245
0245	01356	0456
01356	01456	02456
045	0156	045
0456	0156	045
0156	01456	02456
01456	01456	02456

Table 5.1: Transition tables of both nondeterministic and deterministic finite automata from Exercise 5.2

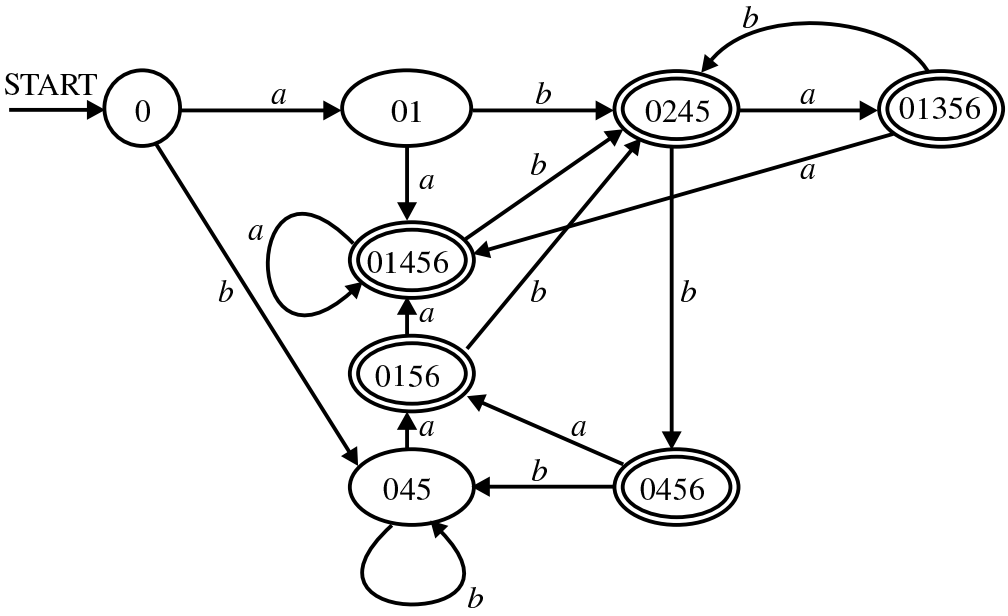

Figure 5.4: Transition diagram of the deterministic finite automaton for searching the string *aba* using Levenshtein distance with at most 1 error allowed from Exercise 5.2 step 3

1. The transition diagram of the nondeterministic automaton is shown in Figure 5.5.

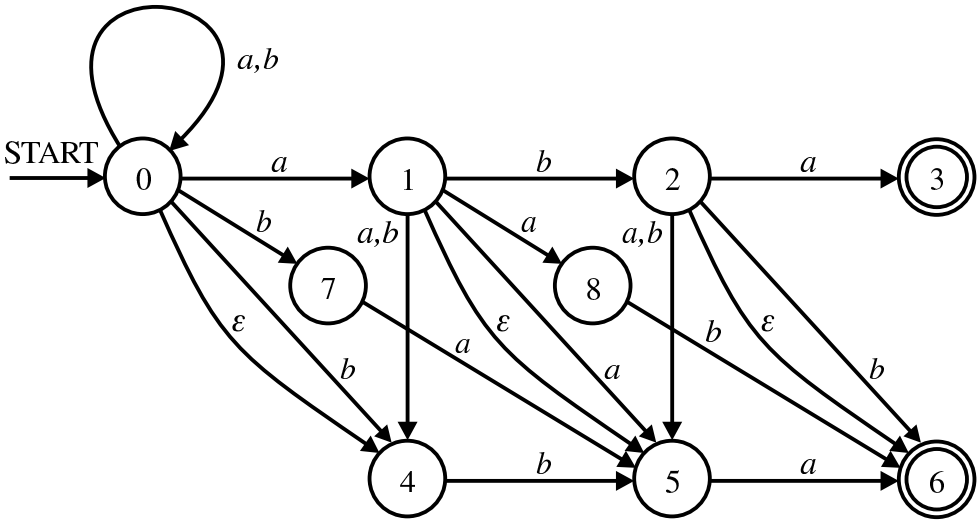

Figure 5.5: Transition diagram of the nondeterministic finite automaton for searching the string *aba* using generalized Levenshtein distance with at most 1 error allowed from Exercise 5.3 step 1

2. Transition diagram of the automaton after removing ε-transitions is shown in Figure 5.6.
3. Transition tables of both (nondeterministic and deterministic) automata are shown in the Table 5.2. Transition diagram of the deterministic automaton is shown in Figure 5.7. □

	a	b
0	$0, 1$	$0, 4, 5, 7$
1	$4, 5, 6, 8$	$2, 4$
2	3	6
3	$\emptyset$	$\emptyset$
4	$\emptyset$	5
5	6	$\emptyset$
6	$\emptyset$	$\emptyset$
7	5	$\emptyset$
8	$\emptyset$	6

	a	b
0	01	0457
01	014568	02457
02457	01356	04567
01356	014568	02457
04567	0156	0457
0457	0156	0457
014568	014568	024567
0156	014568	02457
024567	01356	04567

Table 5.2: Transition tables of both nondeterministic and deterministic finite automata from Exercise 5.3

Exercise 5.4
Create a deterministic finite automaton for searching all strings from the set $P = \{aba, aab\}$ using Hamming distance with at most one error allowed, over an alphabet $A = \{a, b\}$. This automaton can be created by following three steps:

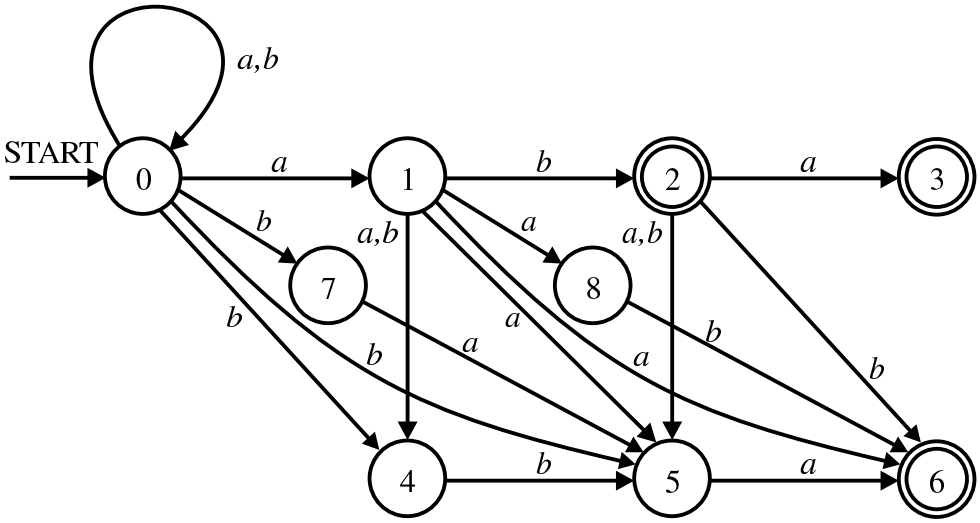

Figure 5.6: Transition diagram of the nondeterministic finite automaton for searching the string *aba* using generalized Levenshtein distance with at most 1 error allowed after removing ε-transitions from Exercise 5.3 step 2

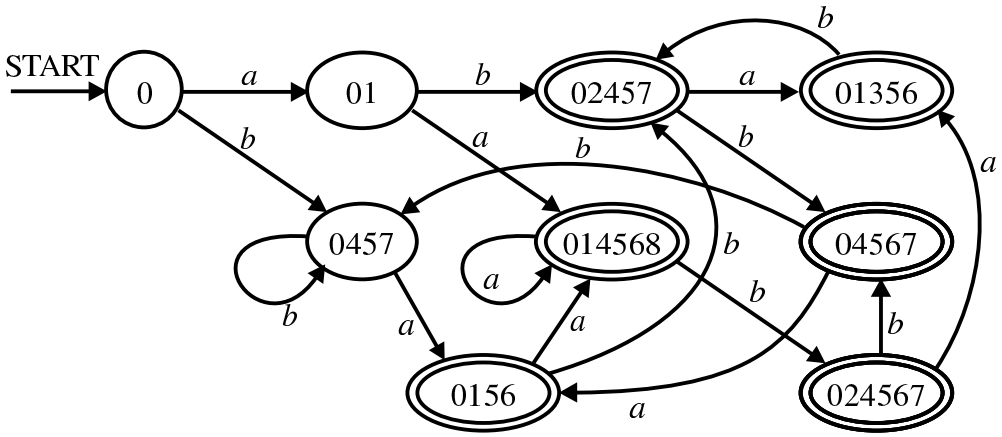

Figure 5.7: Transition diagram of the deterministic finite automaton for searching the string *aba* using generalized Levenshtein distance with at most 1 error allowed from Exercise 5.3 step 3

1. Create a string searching automaton using Hamming distance for each string from the set P. After that, construct an union of this automata. This way created automaton is depicted in Figure 5.8.

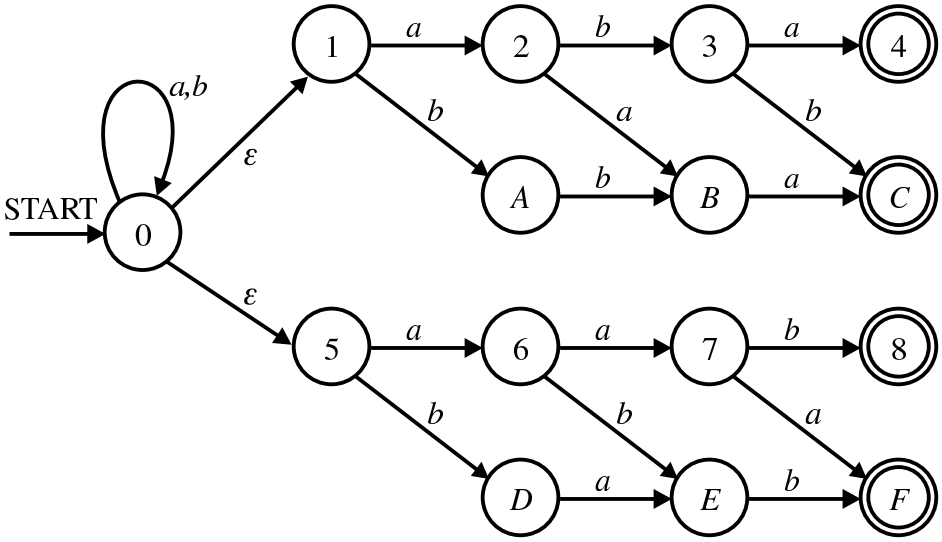

Figure 5.8: Transition diagram of the nondeterministic finite automaton for searching all strings from the set $P = \{aba, aab\}$ using Hamming distance with at most one error allowed from Exercise 5.4 step 1

2. Remove ε-transitions. Transition diagram of the automaton after removing ε-transitions is shown in Figure 5.9.
3. Transform this automaton to a deterministic one using subset construction. Transition tables of both (nondeterministic and deterministic) automata are shown in the Table 5.3. Transition diagram of the deterministic automaton is shown in Figure 5.10. □

Exercise 5.5
Construction of the *SFO*Δ*CO* nondeterministic finite automaton is based on the composition of two copies of the *SFOECO* automaton. The composition is done by insertion of "diagonal" transitions starting in states of the first copy. The inserted transitions represent replace operations. Algorithm 5.6 describes the construction of the *SFO*Δ*CO* automaton in detail.

Algorithm 5.6
Construction of the *SFO*Δ*CO* automaton.
Input: Pattern $P = p_1 p_2 \dots p_m$, k, *SFOECO* automaton $M' = (Q', A, \delta', q_0', F')$ for P.
Output: *SFO*Δ*CO* automaton M.
Method:

1. Create a pair of instances of *SFOECO* automata:
   $M_1 = (Q_1, A, \delta_1, q_{01}, F_1)$,
   $M_2 = (Q_2, A, \delta_2, q_{02}, F_2)$,
   where $Q_1 = \{q_{01}, q_{11}, \dots, q_{m1}\}$,
   $Q_2 = \{q_{02}, q_{12}, \dots, q_{m2}\}$.

	a	b
0	$0, 2, 6$	$0, A, D$
2	B	3
3	4	C
4		
A		B
B	C	
C		
6	7	E
7	F	8
8		
D	E	
E		F
F		

	a	b
0	026	$0AD$
026	$0267B$	$03ADE$
$0267B$	$0267BCF$	$038ADE$
$03ADE$	$0246E$	$0ABCDF$
$0267BCF$	$0267BCF$	$038ADE$
$0246E$	$0267B$	$03ADEF$
$038ADE$	$0246E$	$0ABCDF$
$0AD$	$026E$	$0ABD$
$0ABD$	$026CE$	$0ABD$
$026E$	$0267B$	$03ADEF$
$03ADEF$	$0246E$	$0ABCDF$
$0ABCDF$	$026CE$	$0ABD$
$026CE$	$0267B$	$03ADEF$

Table 5.3: Transition tables of both nondeterministic and deterministic finite automata from Exercise 5.4

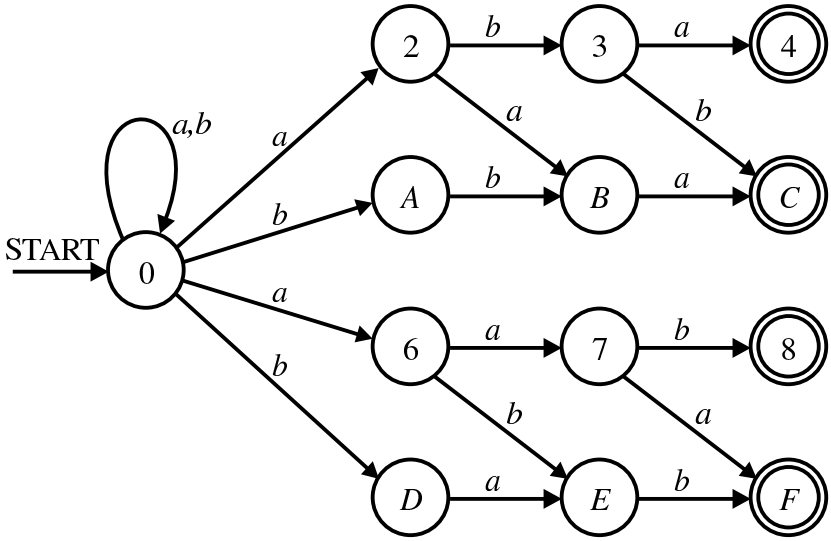

Figure 5.9: Transition diagram of the nondeterministic finite automaton without ε-transitions for searching all strings from the set $P = \{aba, aab\}$ using Hamming distance with at most one error allowed from Exercise 5.4 step 2

2. Construct $SFO\Delta CO$ automaton $M = (Q, A, \delta, q_0, F)$ as follows:
   $Q = Q_1 \cup Q_2$,
   $\delta(q, a) = \delta'(q, a)$ for all $q \in Q, a \in A$,
   $\delta(q_{i1}, a) = \{q_{i+1,2}\}, a \in p_i^{k+}$, for all $i = 0, 1, \ldots, m-1$,
   $q_0 = q_{01}$,
   $F = F' \cup \{q_{m2}\}$.
3. Remove state q_{02}, which is inaccessible from state q_{01}. □

The $SFO\Delta CO$ automaton has $2m + 1$ states.

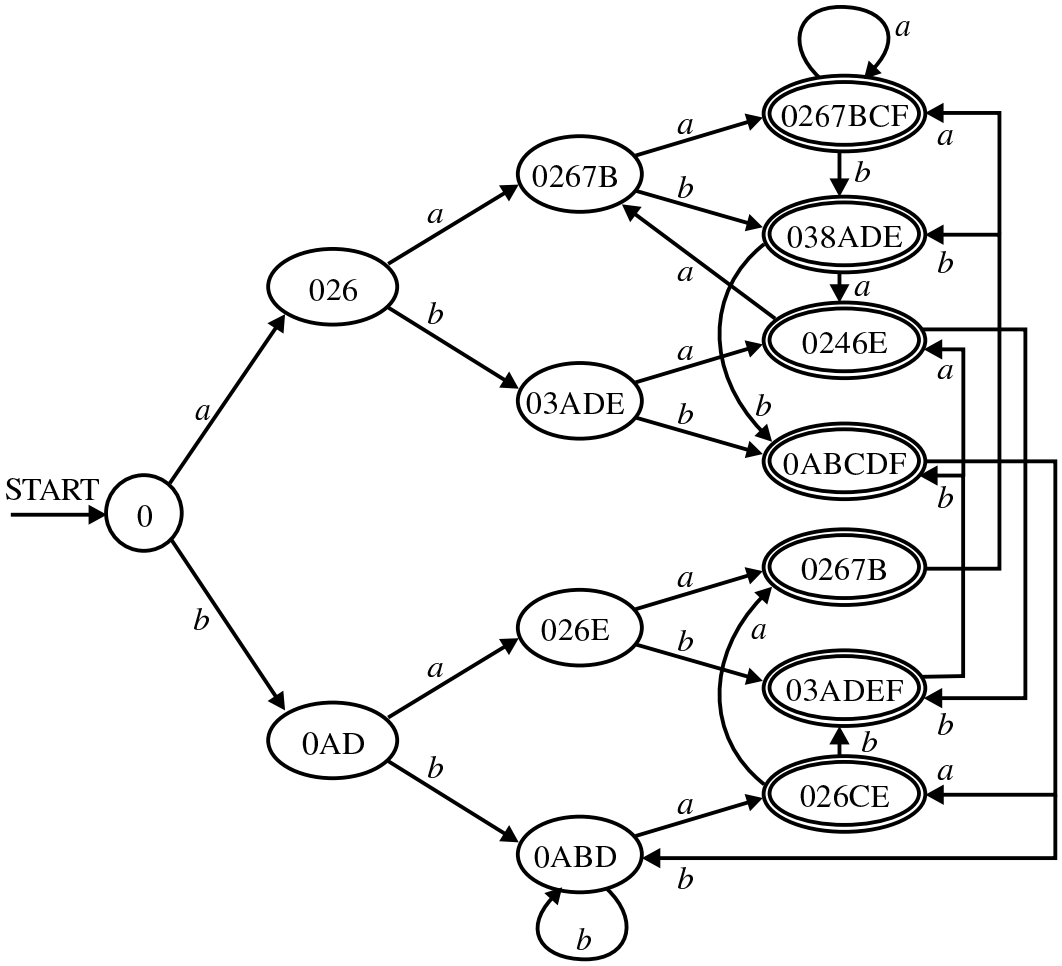

Figure 5.10: Transition diagram of the deterministic finite automaton for searching all strings from the set $P = \{aba, aab\}$ using Hamming distance with at most one error allowed from Exercise 5.4 step 3

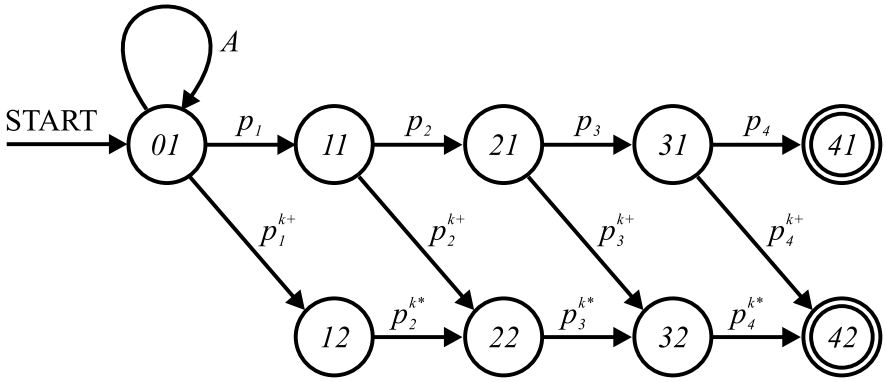

Figure 5.11: Transition diagram of *NFA* for string Δ-matching ($SFO\,\Delta\,CO$ problem) for pattern $P = p_1p_2p_3p_4$

6 Simulation of string matching automata

6.1 Construction of nondeterministic finite automata for string matching

Exercise 6.1
Create a nondeterministic finite automaton for exact string matching for string *abab* over an alphabet $A = \{a, b\}$.

We construct a finite automaton for accepting string *abab*. Then we add self-loop to the initial state labeled by A in order to skip a string preceding each occurrence of the string *abab*. The resulting automaton is in Figure 6.1.

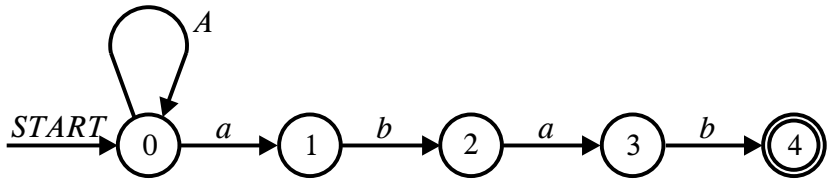

Figure 6.1: Transition diagram of the nondeterministic finite automaton for exact string matching for the string *abab* from Exercise 6.1

We can look at the resulting nondeterministic finite automaton also as to the concatenation of automaton accepting A^* and automaton accepting *abab*.

Exercise 6.2
Create a nondeterministic finite automaton for approximate string matching using Hamming distance for string *abab* over an alphabet $A = \{a, b\}$. Let maximum number k of errors allowed be 1.

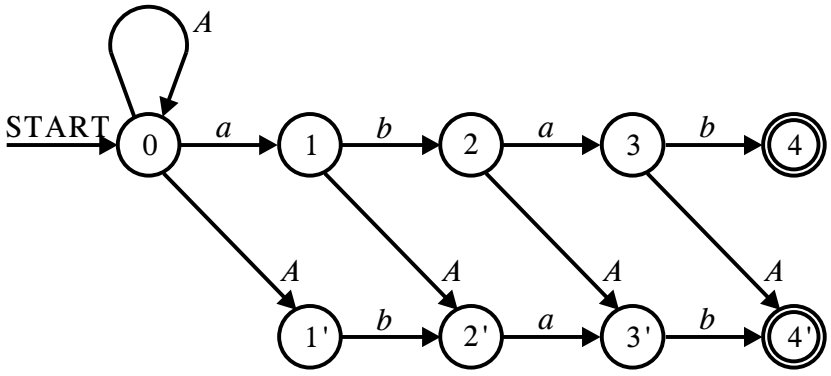

Figure 6.2: Transition diagram of the nondeterministic finite automaton for approximate string matching using Hamming distance for the string *abab* and $k = 1$ from Exercise 6.2

We take two copies $M = (Q, A, \delta, q_0, F)$ and $M' = (Q', A, \delta', q'_0, F')$ of nondeterministic finite automaton for exact string matching from Exercise 6.1. M forms level 0 and M' forms level 1 of the resulting automaton. Level j represents j errors in the found string.

We remove the initial state q'_0 and insert transitions representing edit operation replace. Such transition leads from state q_j to q'_{j+1} and is labeled by A. It increases number of errors (level is inreased) and it is labeled by any symbol of alphabet. The resulting automaton is shown in Figure 6.2.

Exercise 6.3
Create a nondeterministic finite automaton for approximate string matching using Levenshtein distance for string *abab* over an alphabet $A = \{a, b\}$. Let maximum number k of errors allowed be 1.

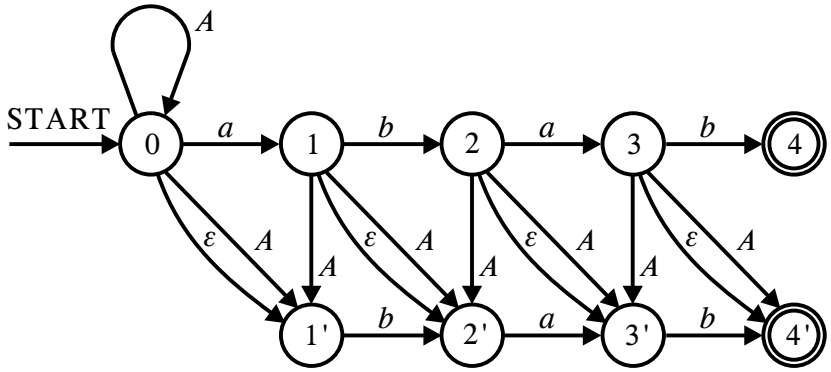

Figure 6.3: Transition diagram of the nondeterministic finite automaton for approximate string matching using Levenshtein distance for the string *abab* and $k = 1$ from Exercise 6.3

We take nondeterministic finite automaton from the previous exercise and add transition for edit operations insert and delete. Transition representing edit operation insert leads from state q_j to q'_j and is labeled by A. Transition representing edit operation delete leads from state q_j to q'_{j+1} and is labeled by ε. Both transitions increase number of errors by one. The resulting automaton is shown in Figure 6.3.

Exercise 6.4
Create a nondeterministic finite automaton for approximate string matching using generalized Levenshtein distance for string *abab* over an alphabet $A = \{a, b\}$. Let maximum number k of errors allowed be 1.

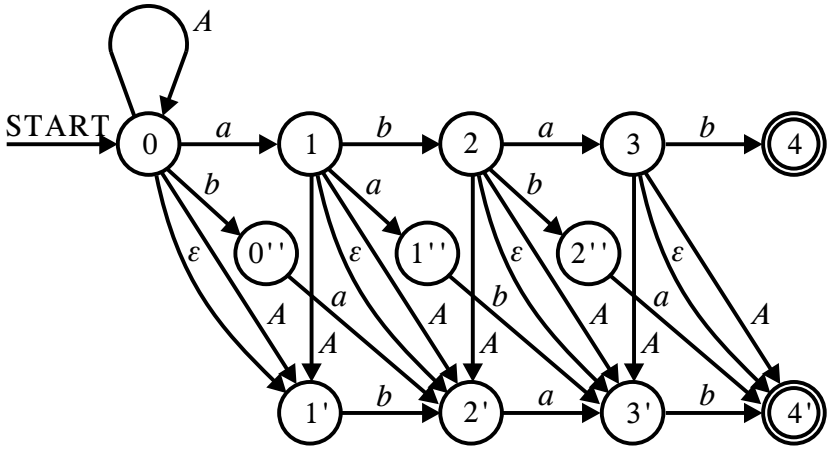

Figure 6.4: Transition diagram of the nondeterministic finite automaton for approximate string matching using generalized Levenshtein distance for the string *abab* and $k = 1$ from Exercise 6.4

We take nondeterministic finite automaton from the previous exercise and add transition

for edit operations transpose. This transition leads from state q_j to new state q''_j and is labeled by p_{j+2}. The transition leading from q''_j is then labeled by p_{j+1} and leads to state q'_{j+2}. It represents that symbols $p_{j+1}p_{j+2}$ are read in inverted order and this increses the number of errors by one. The resulting automaton is shown in Figure 6.4.

6.2 Simulation of nondeterministic finite automaton by dynamic programming

For simulation of nondeterministic finite automaton using the dynamic programming algorithm we evaluate matrix D of size $(m+1) \times (n+1)$, where m is length of pattern and n lenght of text. In each step of simulation we compute one column of the matrix. Value of the j-th row in i-th step of simulation represents the highest active state in depth j of the nondeterministic finite automaton after i-th symbol is read from the input.

Exercise 6.5
Let us have pattern $P = abab$ and text $T = aababaabab$. Construct nondeterministic finite automaton for exact string matching for pattern P and simulate the run of this nondeterministic finite automaton on text T using dynamic programming.

Nondeterministic finite automaton for exact string matching for pattern P has been constructed in Exercise 6.1. For the simulation we will evaluate matrix D of size $(|P|+1)\times(|T|+1)$ according to the following formula.

$$\begin{array}{ll} d_{j,0} = 1, & 0 < j \leq m, \\ d_{0,i} = 0, & 0 < i \leq n, \\ d_{j,i} = \textbf{if } t_i = p_j \textbf{ then } d_{j-1,i-1} \textbf{ else } 1 & \\ & 0 < i \leq n,\ 0 < j \leq m. \end{array} \tag{1}$$

At the beginning only the initial state is active. In each step i of simulation (i.e., evaluation of each column of the matrix) the only active states are those represented by value 0 in the i-th column. For example the column with values $0, 0, 1, 0, 1$ says that only states $0, 1, 3$ are active. They hold the information that prefixes a and aba of the pattern were found on the text position corresponding to the column. If the last value of vector is 0, the final state is active which means that the pattern was found in the corresponing position.

The resulting matrix is in Table 6.1. The pattern was found at positions 5 ($d_{4,5} = 0$) and 10 ($d_{4,10} = 0$).

D		a	a	b	a	b	a	a	b	a	b
	0	0	0	0	0	0	0	0	0	0	0
a	1	0	0	1	0	1	0	0	1	0	1
b	1	1	1	0	1	0	1	1	0	1	0
a	1	1	1	1	0	1	0	1	1	0	1
b	1	1	1	1	1	0	1	1	1	1	0

Table 6.1: Simulation of nondeterministic finite automaton for exact string matching for string $P = abab$ in text $T = aababaabab$ using dynamic programming

Exercise 6.6
Let us have pattern $P = adbbca$ and text $T = adcabcaabadbbca$. Construct nondeterministic

finite automaton for approximate string matching using Hamming distance for pattern P with maximum number of errors $k = 3$. Simulate then the run of the nondeterministic finite automaton on the text T using dynamic programming.

We construct the nondeterministic finite automaton which has transition diagram shown in Figure 6.5. Then we simulate its run. For the simulation we will evaluate matrix D using Formula 2.

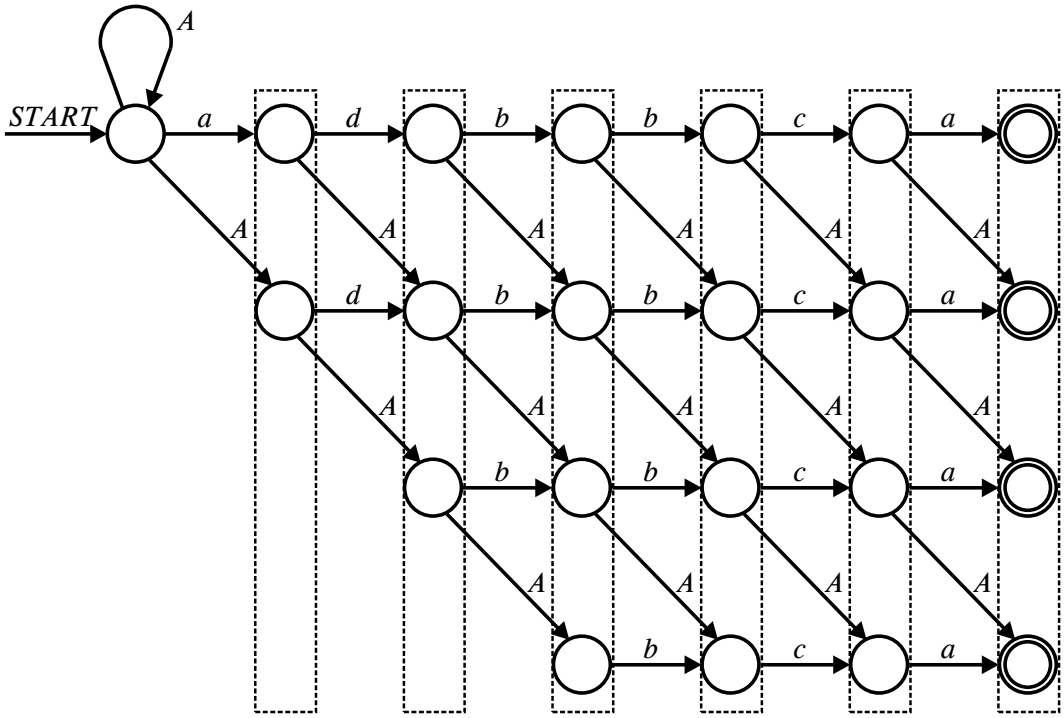

Figure 6.5: Transition diagram of the nondeterministic finite automaton for approximate string matching using Hamming distance for the string *adbbca* and $k = 3$ from Exercise 6.6

$$\begin{array}{lcll} d_{j,0} & \leftarrow & k+1, & 0 < j \leq m \\ d_{0,i} & \leftarrow & 0, & 0 \leq i \leq n \\ d_{j,i} & \leftarrow & \textbf{if } t_i = p_j \textbf{ then } d_{j-1,i-1} \textbf{ else } d_{j-1,i-1}+1, & 0 < i \leq n, 0 < j \leq m \end{array} \quad (2)$$

At the beginning only the initial state is active. Entries of vector $d_{*,0}$ (i.e., column of matrix D for step 0) for other depths is set to $k + 1$ which represents no active state in the corresponding depth of the nondeterministic finite automaton.

Term $d_{j-1,i-1}$ represents transition for *match*, when the input symbol t_i matches symbol p_j in the pattern. In case they do not match, the transition for edit operation *replace* is used. It is represented by term $d_{j-1,i-1}+1$ which increases level by one (and thus increases number of errors by one as well).

The simulation process is displayed in Table 6.2. Each entry greater than $k = 3$ represents no active state in the corresponding depth of nondeterministic finite automaton. We can replace all such values by value $k + 1$ and thus we may decrease space requirements for represenations of entries in the matrix since then we need only $\lceil \log_2(k+1) \rceil$ bits per matrix entry.

D	-	a	d	c	a	b	c	a	a	b	a	d	b	b	c	a
-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
a	4	0	1	1	0	1	1	0	0	1	0	1	1	1	1	0
d	4	5	0	2	2	1	2	2	1	1	2	0	2	2	2	2
b	4	5	6	1	3	2	2	3	3	1	2	3	0	2	3	3
b	4	5	6	7	2	3	3	3	4	3	2	3	3	0	3	4
c	4	5	6	6	8	3	3	4	4	5	4	3	4	4	0	4
a	4	4	6	7	6	9	4	3	4	5	5	5	4	5	5	0

Table 6.2: Simulation of nondeterministic finite automaton for approximate string matching using Hamming distace with at most $k = 3$ errors for pattern $P = adbbca$ and text $T = adcabcaabadbbca$

Exercise 6.7
Modify Formula 2 so that the greatest value in the matrix would be $k + 1$.

Exercise 6.8
Let us have pattern $P = adbbca$ and text $T = adcabcaabadbbca$. Construct nondeterministic finite automaton for approximate string matching using Levenshtein distance for pattern P with maximum number of errors $k = 3$. Simulate then the run of the nondeterministic finite automaton on the text T using dynamic programming.

We construct the nondeterministic finite automaton which has transition diagram shown in Figure 6.6. Then we simulate its run. For the simulation we will evaluate matrix D using Formula 3.

$$\begin{array}{lcll} d_{j,0} & \leftarrow & j, & 0 \leq j \leq m \\ d_{0,i} & \leftarrow & 0, & 0 \leq i \leq n \\ d_{j,i} & \leftarrow & \textbf{min}(\textbf{if } t_i = p_j \textbf{ then } d_{j-1,i-1} & \\ & & \quad \textbf{else } d_{j-1,i-1} + 1, & \\ & & \quad \textbf{if } j < m \textbf{ then } d_{j,i-1} + 1, & \\ & & \quad d_{j-1,i} + 1), & 0 < i \leq n, 0 < j \leq m \end{array} \tag{3}$$

At the beginning not only the initial state q_0 is active but also the state saccessible from the initial state using ε-trantitions (i.e., all states of $\varepsilon{-}CLOSURE(q_0)$). That is why entry $d_{j,0}$ of vector is set to j.

Term $d_{j-1,i-1}$ represents transition for *match*, term $d_{j-1,i-1} + 1$ represents *replace*. Term $d_{j,i-1} + 1$ represents *insert*—it increases level but preserves depth of the nondeterministic finite automaton. Term $d_{j-1,i} + 1$ represents *delete*—it increases both level and depth of the nondeterministic finite automaton while not reading any input symbol (i.e., index i does not change).

The simulation process is displayed in Table 6.3.

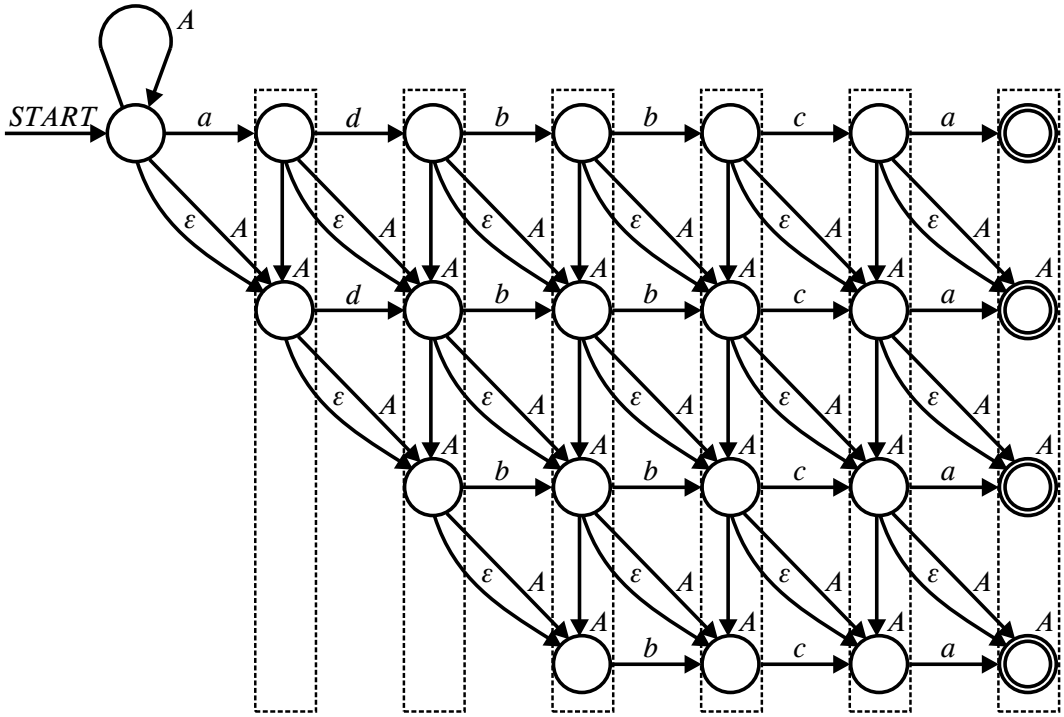

Figure 6.6: Transition diagram of the nondeterministic finite automaton for approximate string matching using Levenshtein distance for the string *adbbca* and $k = 3$ from Exercise 6.8

D	-	a	d	c	a	b	c	a	a	b	a	d	b	b	c	a
-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
a	1	0	1	1	0	1	1	0	0	1	0	1	1	1	1	0
d	2	1	0	1	1	1	2	1	1	1	1	0	1	2	2	1
b	3	2	1	1	2	1	2	2	2	1	2	1	0	1	2	2
b	4	3	2	2	2	2	2	3	3	2	2	2	1	0	1	2
c	5	4	3	2	3	3	2	3	4	3	3	3	2	1	0	1
a	6	5	4	3	2	4	3	2	3	4	3	4	3	2	1	0

Table 6.3: Simulation of the nondeterministic finite automaton for approximate string matching using Levenshtein distace with at most $k = 3$ errors for pattern $P = adbbca$ and text $T = adcabcaabadbbca$

Exercise 6.9
Let us have pattern $P = adbbca$ and text $T = adcabcaabadbbca$. Construct nondeterministic finite automaton for approximate string matching using generalized Levenshtein distance for pattern P with maximum number of errors $k = 3$. Simulate then the run of the nondeterministic finite automaton on the text T using dynamic programming.

We construct the nondeterministic finite automaton which has transition diagram shown in Figure 6.7. Then we simulate its run. For the simulation we will evaluate matrix D using Formula 4.

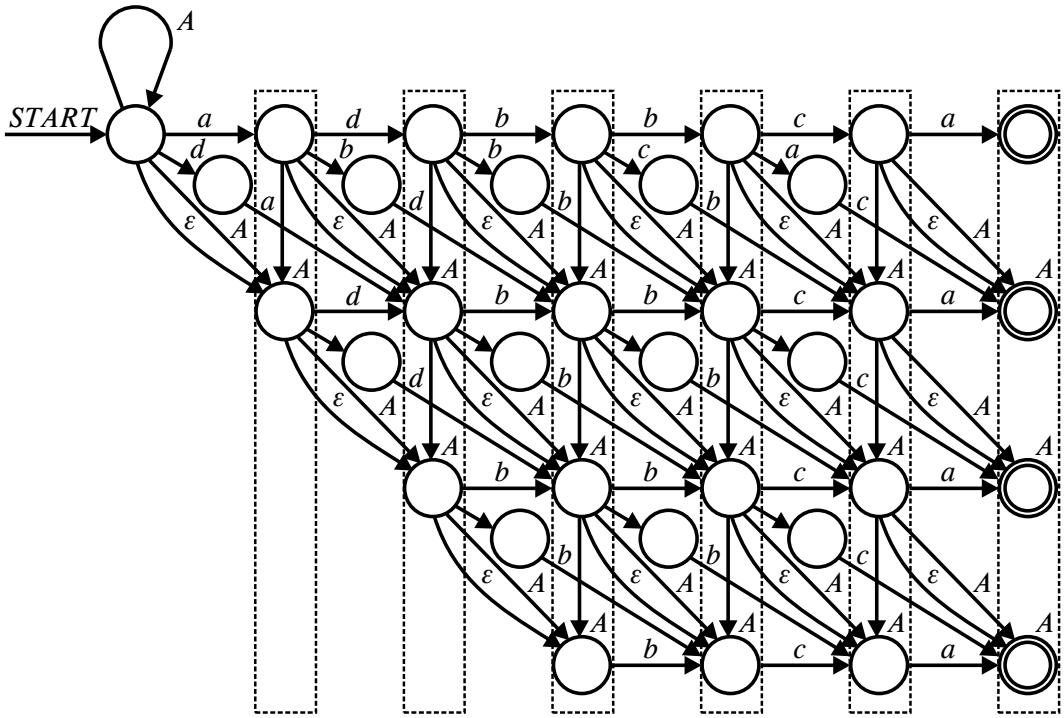

Figure 6.7: Transition diagram of the nondeterministic finite automaton for approximate string matching using generalized Levenshtein distance for the string $adbbca$ and $k = 3$ from Exercise 6.9

$$
\begin{array}{lcll}
d_{j,0} & \leftarrow & j, & 0 \le j \le m \\
d_{0,i} & \leftarrow & 0, & 0 \le i \le n \\
d_{j,i} & \leftarrow & \textbf{min}(\textbf{if } t_i = p_j \textbf{ then } d_{j-1,i-1} \textbf{ else } d_{j-1,i-1} + 1, & \\
 & & \quad \textbf{if } j < m \textbf{ then } d_{j,i-1} + 1, & \\
 & & \quad d_{j-1,i} + 1, & \\
 & & \quad \textbf{if } i > 1 \textbf{ and } j > 1 & \\
 & & \quad\ \textbf{and } t_{i-1} = p_j \textbf{ and } t_i = p_{j-1} & \\
 & & \quad \textbf{then } d_{j-2,i-2} + 1), & 0 < i \le n, 0 < j \le m
\end{array}
\tag{4}
$$

The initial configuration of vector $d_{*,0}$ is the same as in the previous exercise.

In addition to the term for transitions *match*, *replace*, *delete*, and *insert* we have to add term $d_{j-2,i-2} + 1)$ for transition *transpose*. In this transition an active state goes through

two depths of automaton while increasing level by one and reading two consequent symbols of the pattern in inverted order. The simulation process is displayed in Table 6.4.

D	-	a	d	b	c	b	a	a	b	a	d	b	b	c	a
-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
a	1	0	1	1	1	1	0	0	1	0	1	1	1	1	0
d	2	1	0	1	2	2	1	1	1	1	0	1	2	2	1
b	3	2	1	0	1	2	2	2	1	2	1	0	1	2	2
b	4	3	2	1	1	1	2	3	2	2	2	1	0	1	2
c	5	4	3	2	1	1	2	3	3	3	3	2	1	0	1
a	6	5	4	3	2	2	1	2	4	3	4	3	2	1	0

Table 6.4: Simulation of the nondeterministic finite automaton for approximate string matching using generalized Levenshtein distace with at most $k = 3$ errors for pattern $P = adbbca$ and text $T = adcabcaabadbbca$

6.3 Simulation of nondeterministic finite automaton by bit parallelism

Bit parallelism method is covered by several algorithms: Shift-Or, Shift-And, and Shift-Add. We will focus on Shift-Or. Algorithm Shift-And differs only in such a way that 0 and one is exchanged as well as bitwise operations OR and AND.

Shift-Or algorithm uses bitwise matrices R^l, $0 \leq l \leq k$ of size $m \times (n+1)$ and mask bitwise matrix D of size $m \times |A|$. Every entry $r^l_{j,i}$, $0 < j \leq m$, $0 \leq i \leq n$, contains 0 only if the state in depth j and level l is active. Every entry $d_{j,x}$, $0 < j \leq m$, $x \in A$, contains 0 when $p_j = x$, or 1 otherwise. The matrices are represented as tables of bit vectors as shown below.

$$R^l_i = \begin{bmatrix} r^l_{1,i} \\ r^l_{2,i} \\ \vdots \\ r^l_{m,i} \end{bmatrix} \text{ a } D[x] = \begin{bmatrix} d_{1,x} \\ d_{2,x} \\ \vdots \\ d_{m,x} \end{bmatrix}, 0 \leq i \leq n, 0 \leq l \leq k, x \in A. \tag{5}$$

Exercise 6.10
Let us have pattern $P = adbbca$ and text $T = adcabcaabadbbca$. Construct nondeterministic finite automaton for exact string matching for pattern P and simulate the run of this nondeterministic finite automaton on text T using Shift-Or algorithm.

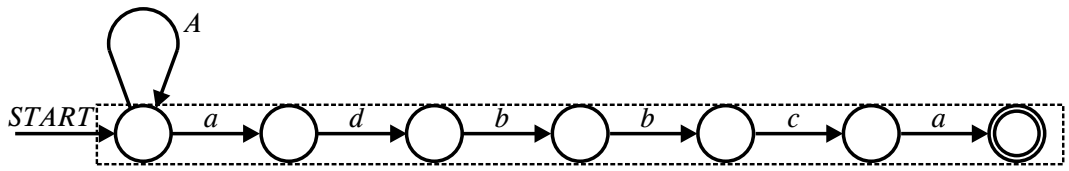

Figure 6.8: Transition diagram of the nondeterministic finite automaton for exact string matching for the string $adbbca$ from Exercise 6.10

Nondeterministic finite automaton for exact string matching for pattern P is shown in Figure 6.8. In Shift-Or algorithm we have one bit vector R^l of length m for each level l

of nondeterministic finite automaton. For the simulation we will evaluate R_i^0, $0 \leq i \leq n$ according to Formula 6.

$$\begin{array}{lcll} r_{j,0}^0 & \leftarrow & 1, & 0 < j \leq m \\ R_i^0 & \leftarrow & \mathbf{shr}(R_{i-1}^0) \text{ OR } D[t_i], & 0 < i \leq n \end{array} \quad (6)$$

Mask table for pattern P is shown in Table 6.5.

D	a	b	c	d	$\Sigma \setminus \{a,b,c,d\}$
a	0	1	1	1	1
d	1	1	1	0	1
b	1	0	1	1	1
b	1	0	1	1	1
c	1	1	0	1	1
a	0	1	1	1	1

Table 6.5: Matrix D for pattern $P = adbbca$

At the beginning only the initial state is active. That is why all bits of the vector R_0^l contain only ones. The initial state is not represented in the vector since it is always active. It is implemented by inserting 0 at the beginning of the vector during bitwise operation right shift (**shr**()).

Term $\mathbf{shr}(R_{i-1}^0)$ OR $D[t_i]$ simulates transition *match*, when all states move to the right and only those matching input symbol t_i are selected. The simulation process is displayed in Table 6.6.

R^0	-	a	d	c	a	b	c	a	a	b	a	d	b	b	c	a
a	1	**0**	1	1	**0**	1	1	**0**	**0**	1	**0**	1	1	1	1	**0**
d	1	1	**0**	1	1	1	1	1	1	1	1	**0**	1	1	1	1
b	1	1	1	1	1	1	1	1	1	1	1	1	**0**	1	1	1
b	1	1	1	1	1	1	1	1	1	1	1	1	1	**0**	1	1
c	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**0**	1
a	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**0**

Table 6.6: Matrix R^0 for exact string matching for pattern $P = adbbca$ and text $T = adcabcaabadbbca$

Exercise 6.11
Let us have pattern $P = adbbca$ and text $T = adcabcaabadbbca$. Construct nondeterministic finite automaton for approximate string matching using Hamming distance for pattern P with maximum number of errors $k = 3$. Simulate the run of this nondeterministic finite automaton on text T using Shift-Or algorithm.

For the simulation we need $k+1$ vectors R_i^l, $0 \leq i \leq n$, $0 \leq l \leq k$. We will evaluate them according to Formula 7.

$$
\begin{array}{lcll}
r^l_{j,0} & \leftarrow & 1, & 0 < j \leq m, 0 \leq l \leq k \\
R^0_i & \leftarrow & \mathbf{shr}(R^0_{i-1}) \text{ OR } D[t_i], & 0 < i \leq n \\
R^l_i & \leftarrow & (\mathbf{shr}(R^l_{i-1}) \text{ OR } D[t_i]) \text{ AND } \mathbf{shr}(R^{l-1}_{i-1}), & 0 < i \leq n, 0 < l \leq k
\end{array} \tag{7}
$$

Term $\mathbf{shr}(R^0_{i-1})$ OR $D[t_i]$ simulates transition *match* and term $\mathbf{shr}(R^{l-1}_{i-1})$ simulates transition *replace*, where all active states move to the next depth but also to the next level of the nondeterministic finite automaton. The simulation process is displayed in Table 6.7.

R^0	-	a	d	c	a	b	c	a	a	b	a	d	b	b	c	a
a	1	**0**	1	1	**0**	1	1	**0**	**0**	1	**0**	1	1	1	1	**0**
d	1	1	**0**	1	1	1	1	1	1	1	1	**0**	1	1	1	1
b	1	1	1	1	1	1	1	1	1	1	1	1	**0**	1	1	1
b	1	1	1	1	1	1	1	1	1	1	1	1	1	**0**	1	1
c	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**0**	1
a	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**0**
R^1	-	a	d	c	a	b	c	a	a	b	a	d	b	b	c	a
a	1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
d	1	1	**0**	1	1	**0**	1	1	**0**	**0**	1	**0**	1	1	1	1
b	1	1	1	**0**	1	1	1	1	1	**0**	1	1	**0**	1	1	1
b	1	1	1	1	1	1	1	1	1	1	1	1	1	**0**	1	1
c	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**0**	1
a	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**0**
R^2	-	a	d	c	a	b	c	a	a	b	a	d	b	b	c	a
a	1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
d	1	1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
b	1	1	1	**0**	1	**0**	**0**	1	1	**0**	**0**	1	**0**	**0**	1	1
b	1	1	1	1	**0**	1	1	1	1	1	**0**	1	1	**0**	1	1
c	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**0**	1
a	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**0**
R^3	-	a	d	c	a	b	c	a	a	b	a	d	b	b	c	a
a	1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
d	1	1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
b	1	1	1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
b	1	1	1	1	**0**	**0**	**0**	**0**	1	**0**	**0**	**0**	**0**	**0**	**0**	1
c	1	1	1	1	1	**0**	**0**	1	1	1	1	**0**	1	1	**0**	1
a	1	1	1	1	1	1	1	**0**	1	1	1	1	1	1	1	**0**

Table 6.7: Simulation of nondeterministic finite automaton for approximate string matching using Hamming distace with at most $k = 3$ errors for pattern $P = adbbca$ and text $T = adcabcaabadbbca$

Exercise 6.12
Let us have pattern $P = adbbca$ and text $T = adcabcaabadbbca$. Construct nondeterministic finite automaton for approximate string matching using Levenshtein distance for pattern P with maximum number of errors $k = 3$. Simulate the run of this nondeterministic finite automaton on text T using Shift-Or algorithm.

During the simulation we evaluate vectors R_i^l, $0 \leq i \leq n$, $0 \leq l \leq k$ according to Formulae 8 and 9.

$$\begin{array}{lcll} r_{j,0}^l & \leftarrow & 0, & 0 < j \leq l, 0 < l \leq k \\ r_{j,0}^l & \leftarrow & 1, & l < j \leq m, 0 \leq l \leq k \\ R_i^0 & \leftarrow & \mathbf{shr}(R_{i-1}^0) \text{ OR } D[t_i], & 0 < i \leq n \\ R_i^l & \leftarrow & (\mathbf{shr}(R_{i-1}^l) \text{ OR } D[t_i]) & \\ & & \text{AND } \mathbf{shr}(R_{i-1}^{l-1} \text{ AND } R_i^{l-1}) \text{ AND } (R_{i-1}^{l-1} \text{ OR } V), & 0 < i \leq n, 0 < l \leq k \end{array} \tag{8}$$

$$V = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{bmatrix}, \text{ kde } v_m = 1 \text{ a } v_j = 0, \forall j, 1 \leq j < m. \tag{9}$$

Term $\mathbf{shr}(R_{i-1}^0)$ OR $D[t_i]$ simulates transition *match* and term $\mathbf{shr}(R_{i-1}^{l-1})$ simulates transition *replace*. Term R_{i-1}^{l-1} OR V simulates transition *insert* where all active states move to the next level within the same depth. Vector V preserves *insert* transition in the last depth whe no transition *insert* is. Term $\mathbf{shr}(R_i^{l-1})$ represents transition *delete*, where all active states move to the next level in the next depth of the nondeterministic finite automaton while not input symbol is read. Note, that we changed the initial setting of the vector because of ε–$CLOSURE(q_0)$.

The simulation process is displayed in Table 6.8.

Exercise 6.13
Let us have pattern $P = adbbca$ and text $T = adcabcaabadbbca$. Construct nondeterministic finite automaton for approximate string matching using generalized Levenshtein distance for pattern P with maximum number of errors $k = 3$. Simulate the run of this nondeterministic finite automaton on text T using Shift-Or algorithm.

During the simulation we use not only vectors R_i^l, $0 \leq i \leq n$, $0 \leq l \leq k$ but also vectors S_i^l, $0 \leq i \leq n$, $0 \leq l < k$. We will evaluate them according to Formulae 11 and 10.

$$S_i^l = \begin{bmatrix} s_{1,i}^l \\ s_{2,i}^l \\ \vdots \\ s_{m,i}^l \end{bmatrix}, 0 \leq l < k, 0 \leq i < n. \tag{10}$$

$$
\begin{array}{lcll}
r^l_{j,0} & \leftarrow & 0, & 0 < j \leq l, 0 < l \leq k \\
r^l_{j,0} & \leftarrow & 1, & l < j \leq m, 0 \leq l \leq k \\
R^0_i & \leftarrow & \mathbf{shr}(R^0_{i-1}) \text{ OR } D[t_i], & 0 < i \leq n \\
R^l_i & \leftarrow & (\mathbf{shr}(R^l_{i-1}) \text{ OR } D[t_i]) & \\
 & & \text{AND } \mathbf{shr}(R^{l-1}_{i-1} \text{ AND } R^{l-1}_i \text{ AND } (S^{l-1}_{i-1} \text{ OR } D[t_i])) & \\
 & & \text{AND } (R^{l-1}_{i-1} \text{ OR } V), & 0 < i \leq n, 0 < l \leq k \\
s^l_{j,0} & \leftarrow & 1, & 0 < j \leq m, 0 \leq l < k \\
S^l_i & \leftarrow & \mathbf{shr}(R^l_{i-1}) \text{ OR } \mathbf{shl}(D[t_i]), & 0 < i < n, 0 \leq l < k
\end{array}
\tag{11}
$$

Term $\mathbf{shr}(R^0_{i-1})$ OR $D[t_i]$ simulates transition *match*, term $\mathbf{shr}(R^{l-1}_{i-1})$ simulates transition *replace*, term R^{l-1}_{i-1} OR V simulates transition *insert* and term $\mathbf{shr}(R^{l-1}_i)$ represents transition *delete*.

In addition to Formula 8 we had to add simulation of transition *replace*. Term S^{l-1}_{i-1} OR $D[t_i]$ for evaluation of R^l_i together with term $\mathbf{shr}(R^l_{i-1})$ OR $\mathbf{shl}(D[t_i])$ for evaluation of S^l_i provides that all active states move to the next level while depth of nondeterministic finite automaton is increased by two and two current consecutive pattern symbols are read in inverted order.

The simulation process is displayed in Table 6.9.

R^0	-	a	d	c	a	b	c	a	a	b	a	d	b	b	c	a
a	1	**0**	1	1	**0**	1	1	**0**	**0**	1	**0**	1	1	1	1	**0**
d	1	1	**0**	1	1	1	1	1	1	1	1	**0**	1	1	1	1
b	1	1	1	1	1	1	1	1	1	1	1	1	**0**	1	1	1
b	1	1	1	1	1	1	1	1	1	1	1	1	1	**0**	1	1
c	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**0**	1
a	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**0**

R^1	-	a	d	c	a	b	c	a	a	b	a	d	b	b	c	a
a	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
d	1	**0**	**0**	**0**	**0**	**0**	1	**0**	**0**	**0**	**0**	**0**	**0**	1	1	**0**
b	1	1	**0**	**0**	1	**0**	1	1	1	**0**	1	**0**	**0**	**0**	1	1
b	1	1	1	1	1	1	1	1	1	1	1	1	**0**	**0**	**0**	1
c	1	1	1	1	1	1	1	1	1	1	1	1	1	**0**	**0**	**0**
a	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**0**	**0**

R^2	-	a	d	c	a	b	c	a	a	b	a	d	b	b	c	a
a	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
d	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
b	1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
b	1	1	**0**	**0**	**0**	**0**	**0**	1	1	**0**	**0**	**0**	**0**	**0**	**0**	**0**
c	1	1	1	**0**	1	1	**0**	1	1	1	1	1	**0**	**0**	**0**	**0**
a	1	1	1	1	**0**	1	1	**0**	1	1	1	1	1	**0**	**0**	**0**

R^3	-	a	d	c	a	b	c	a	a	b	a	d	b	b	c	a
a	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
d	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
b	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
b	1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
c	1	1	**0**	**0**	**0**	**0**	**0**	**0**	1	**0**	**0**	**0**	**0**	**0**	**0**	**0**
a	1	1	1	**0**	**0**	1	**0**	**0**	**0**	1	**0**	1	**0**	**0**	**0**	**0**

Table 6.8: Simulation of nondeterministic finite automaton for approximate string matching using Levenshtein distace with at most $k = 3$ errors for pattern $P = adbbca$ and text $T = adcabcaabadbbca$

R^0		-	a	d	b	c	b	a	a	b	a	d	b	b	c	a
	a	1	**0**	1	1	1	1	**0**	**0**	1	**0**	1	1	1	1	**0**
	d	1	1	**0**	1	1	1	1	1	1	1	**0**	1	1	1	1
	b	1	1	1	**0**	1	1	1	1	1	1	1	**0**	1	1	1
	b	1	1	1	1	1	1	1	1	1	1	1	1	**0**	1	1
	c	1	1	1	1	1	1	1	1	1	1	1	1	1	**0**	1
	a	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**0**
S^0	a	1	1	**0**	1	1	1	1	1	1	1	**0**	1	1	1	1
	d	1	1	1	1	1	1	1	1	**0**	1	1	1	1	1	1
	b	1	1	1	**0**	1	1	1	1	1	1	1	**0**	1	1	1
	b	1	1	1	1	**0**	1	1	1	1	1	1	1	1	1	1
	c	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	a	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
R^1	a	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	d	1	**0**	**0**	**0**	1	1	**0**	**0**	**0**	**0**	**0**	**0**	1	1	**0**
	b	1	1	**0**	**0**	**0**	1	1	1	**0**	1	**0**	**0**	**0**	1	1
	b	1	1	1	**0**	**0**	**0**	1	1	1	1	1	**0**	**0**	**0**	1
	c	1	1	1	1	**0**	**0**	1	1	1	1	1	1	**0**	**0**	**0**
	a	1	1	1	1	1	1	**0**	1	1	1	1	1	1	**0**	**0**
S^1	a	1	1	**0**	1	1	1	1	1	1	1	**0**	1	1	1	1
	d	1	1	1	**0**	1	**0**	1	1	**0**	1	1	**0**	**0**	1	1
	b	1	1	1	**0**	1	1	1	1	**0**	1	1	**0**	**0**	1	1
	b	1	1	1	1	**0**	1	1	1	1	1	1	1	1	**0**	1
	c	1	1	1	1	1	1	**0**	1	1	1	1	1	1	1	**0**
	a	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
R^2	a	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	d	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	b	1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	b	1	1	**0**	**0**	**0**	**0**	**0**	1	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	c	1	1	1	**0**	**0**	**0**	**0**	1	1	1	1	**0**	**0**	**0**	**0**
	a	1	1	1	1	**0**	**0**	**0**	**0**	1	1	1	1	**0**	**0**	**0**
S^2	a	1	1	**0**	1	1	1	1	1	1	1	**0**	1	1	1	1
	d	1	1	1	**0**	1	**0**	1	1	**0**	1	1	**0**	**0**	1	1
	b	1	1	1	**0**	1	**0**	1	1	**0**	1	1	**0**	**0**	1	1
	b	1	1	1	1	**0**	1	1	1	1	1	1	1	1	**0**	1
	c	1	1	1	1	1	1	**0**	**0**	1	**0**	1	1	1	1	**0**
	a	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
R^3	a	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	d	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	b	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	b	1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	c	1	1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	a	1	1	1	**0**	**0**	**0**	**0**	**0**	1	**0**	1	**0**	**0**	**0**	**0**

Table 6.9: Simulation of nondeterministic finite automaton for approximate string matching using generalized Levenshtein distace with at most $k = 3$ errors for pattern $P = adbbca$ and text $T = adcabcaabadbbca$

7 Prefix and suffix automata

7.1 Prefix automata

Exercise 7.1
Create prefix automaton accepting set $Pref(abab)$.

Solution of this task is very simple:

1. Create a finite automaton accepting string $abab$.
2. Make all states final.

Transition diagram of the resulting prefix automaton is depicted in Fig. 7.1. □

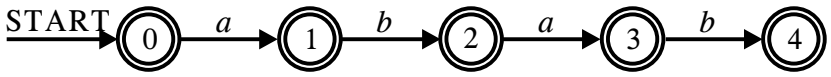

Figure 7.1: Transition diagram of the prefix automaton accepting set $Pref(abab)$ from Exercise 7.1

Exercise 7.2
Create prefix automaton accepting set of prefixes of set $S = \{abab, aabb\}$.

Solution of this task can be done in this way:

1. Create a deterministic finite automaton accepting set S.
2. Make all states final.

Transition diagram of the resulting prefix automaton is depicted in Fig. 7.2. □

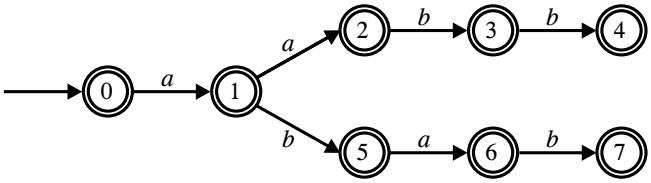

Figure 7.2: Transition diagram of the prefix automaton accepting set $Pref(abab, aabb)$ from Exercise 7.2

Exercise 7.3
Create prefix automaton accepting set of approximate prefixes from set $APref(abab)$ for Hamming distance with $k = 1$.

Solution of this task can be done in this way:

1. Create a deterministic finite automaton accepting set of strings having Hamming distance $k \leq 1$ from string $abab$.
2. Make all states final.

Transition diagram of the resulting approximate prefix automaton is depicted in Fig. 7.3. □

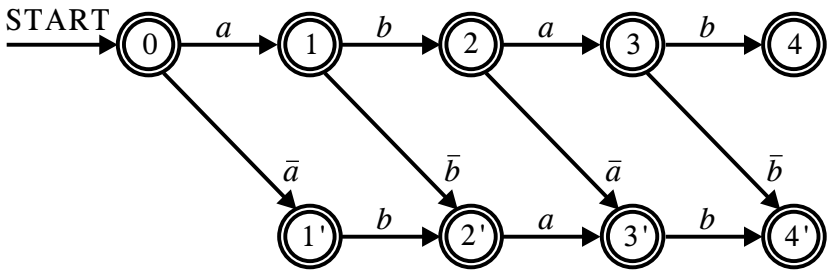

Figure 7.3: Transition diagram of the approximate prefix automaton from Exercise 7.3

Exercise 7.4
Create prefix automaton accepting set of approximate prefixes from set $APref(abab)$ for Levenshtein distance with $k = 1$.

Solution of this task can be done in this way:

1. Create a nondeterministic finite automaton M_1 accepting set of strings having Levenshtein distance $k \leq 1$ from string *abab*. Transition diagram of this automaton is depicted in Fig. 7.4.

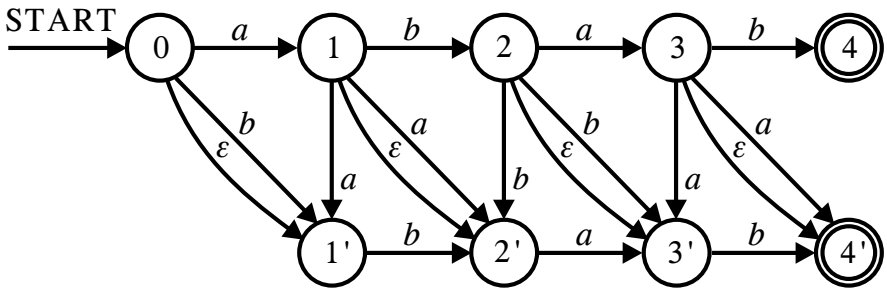

Figure 7.4: Transition diagram of the automaton M_1 from Exercise 7.4

2. Construct finite automaton M_2 without ε-transitions equivalent to automaton M_1. Transition diagram of this automaton is depicted in Fig. 7.5.

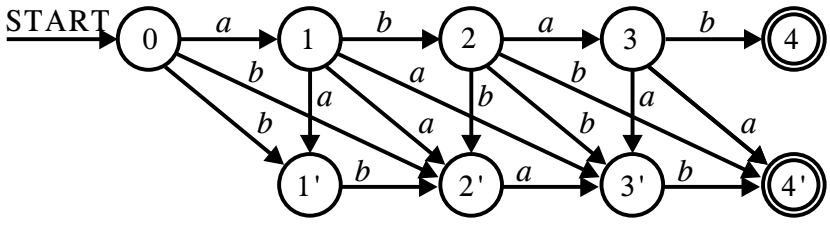

Figure 7.5: Transition diagram of the automaton M_2 after removal of ε-transitions from Exercise 7.4

3. Construct deterministic finite automaton M_3 equivalent to automaton M_2. Make all states final. Transition diagram of this automaton is depicted in Fig. 7.6. Transition tables of automata M_2 and M_3 are shown in Table 7.1. □

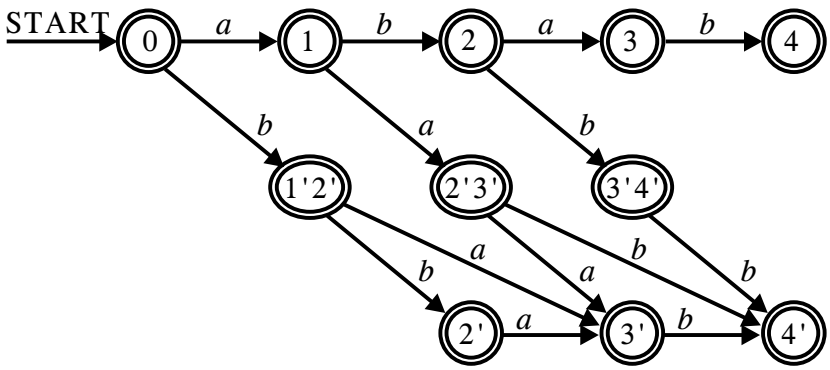

Figure 7.6: Transition diagram of the deterministic prefix automaton M_3 from Exercise 7.4

	a	b
0	1	$1', 2'$
1	$2', 3'$	2
2	3	$3', 4'$
3	$4'$	4
4		
$1'$		$2'$
$2'$	$3'$	
$3'$		$4'$
$4'$		

a)

	a	b
0	1	$1'2'$
1	$2'3'$	2
2	3	$3', 4'$
3	$4'$	4
4		
$1'2'$	$3'$	$2'$
$2'3'$	$3'$	$4'$
$3'4'$		$4'$
$2'$	$3'$	
$3'$		$4'$
$4'$		

b)

Table 7.1: Transition tables of automata M_2 and M_3 from Exercise 7.4

7.2 Suffix automata

Exercise 7.5
Create a deterministic suffix automaton for text $T = ababa$.
Construction of the suffix automaton can be done in four steps using ε-transitions based method.

1. Create a finite automaton accepting text T. Its transition diagram is depicted in Figure 7.7.

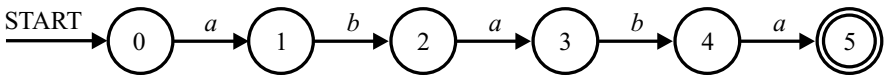

Figure 7.7: Transition diagram of the deterministic finite automaton accepting string *ababa* from Exercise 7.5 step 1

2. Insert ε-transitions from the initial state leading into all other states. The result of this step is a nondeterministic finite automaton accepting all suffixes of text T having transition diagram shown in Figure 7.8.

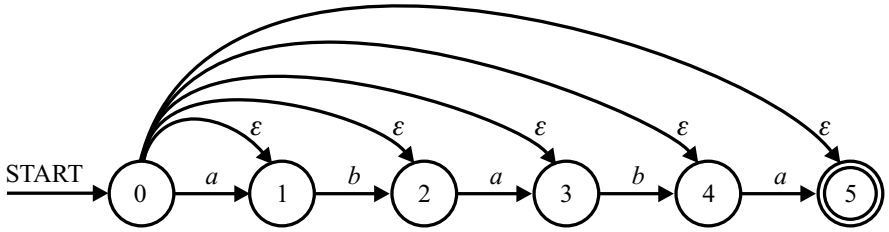

Figure 7.8: Transition diagram of the nondeterministic finite automaton accepting all suffixes of string *ababa* from Exercise 7.5 step 2

3. Remove all ε-transitions from the automaton created in the previous step. The result is a nondeterministic finite automaton without ε-transitions, which has transition diagram shown in Figure 7.9.

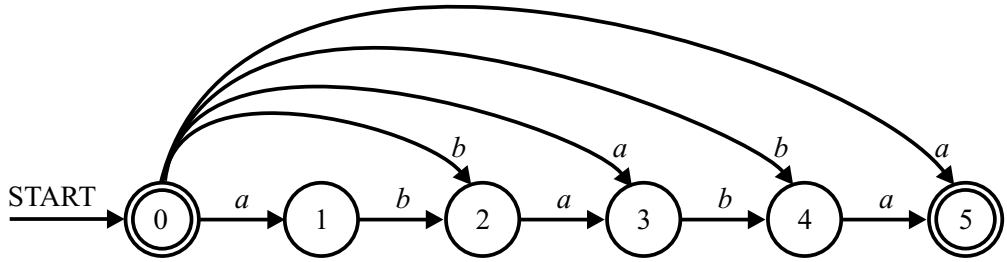

Figure 7.9: Transition diagram of the nondeterministic finite automaton without ε-transitions accepting all suffixes of the string *ababa* from Exercise 7.5 step 3

4. Transform the nondeterministic finite automaton to a deterministic one using subset construction. Transition tables of both (nondeterministic and deterministic) automata are shown in Table 7.2. The result is a deterministic finite automaton having transition diagram shown in Figure 7.10. □

	a	b
0	135	24
1		2
2	3	
3		4
4	5	
5		

	a	b
0	135	24
135		24
24	35	
35		4
4	5	
5		

Table 7.2: Transition tables of nondeterministic and deterministic suffix automata from Exercise 7.5

Exercise 7.6
Create a deterministic suffix automaton for set of strings $S = \{aba, abab, bab\}$.

Construction of the suffix automaton can be done in four steps using ε-transitions based method:

1. Create a finite automaton M_1 accepting strings from set S. Its transition diagram is depicted in Fig. 7.11.

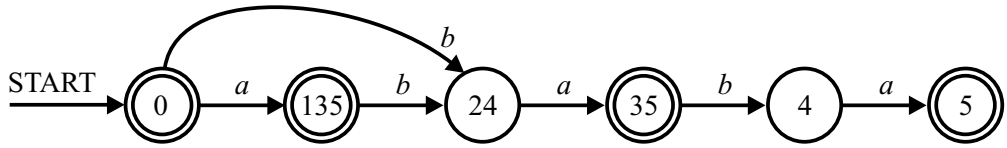

Figure 7.10: Transition diagram of the deterministic finite automaton accepting all suffixes of the string *ababa* from Exercise 7.5 step 4

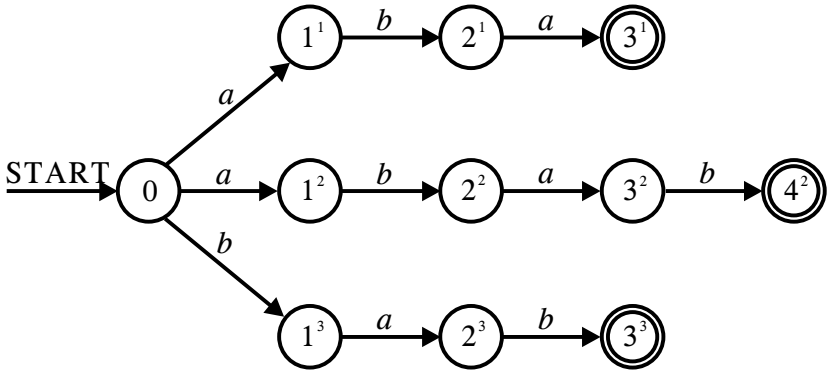

Figure 7.11: Transition diagram of the finite automaton M_1 from Exercise 7.6 step 1 accepting set $S = \{aba, abab, bab\}$

2. Insert ε-transitions from the initial state leading into all other states. The result of this step is a nondeterministic finite automaton accepting all suffixes of all strings in set S. Its transition diagram is depicted in Fig. 7.12.
3. Remove all ε-transitions from the automaton created in the previous step. The result is a nondeterministic finite automaton without ε-transitions having transition diagram shown in Fig. 7.13.
4. Transform the nondeterministic finite automaton to a deterministic one using subset construction. Transition tables of both (nondeterministic and deterministic) automata are shown in Table 7.3. The result is a deterministic finite automaton having transition diagram shown in Fig. 7.14. □

Exercise 7.7
Create a deterministic suffix automaton for set of approximate suffixes from $Suff(aabb)$ over alphabet $A = \{a, b\}$ for Hamming distance with $k = 1$.

Construction of the approximate suffix automaton can be done in four steps using ε-transitions based method:

1. Create a finite automaton accepting string *aabb* and all strings having Hamming distance $k = 1$ from it. Its transition diagram is depicted in Fig. 7.15.
2. Insert ε-transitions from the initial state leading into all states on the zero level (states 1, 2, 3 and 4). The result of this step is a nondeterministic finite automaton accepting all approximate suffixes of string *aabb*. Its transition diagram is depicted in Fig. 7.16.
3. Remove ε-transitions from the automaton created in the previous step. The result is a nondeterministic finite automaton without ε-transitions having transition diagram

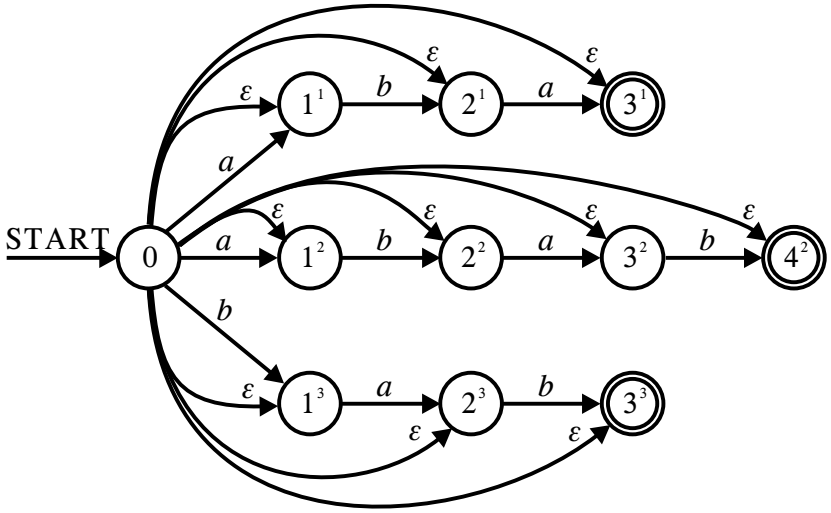

Figure 7.12: Transition diagram of the nondeterministic finite automaton accepting all suffixes of strings from set S from Exercise 7.6 step 2

depicted in Fig. 7.17.

4. Transform the nondeterministic finite automaton to a deterministic one using subset construction. Transition tables of both (nondeterministic and deterministic) automata are shown in Table 7.4. The result is a deterministic finite automaton having transition diagram shown in Fig. 7.18. □

Exercise 7.8
Create a deterministic suffix automaton accepting set of approximate suffixes from $Suff(aabb)$ over alphabet $A = \{a, b\}$ for Levenshtein distance with $k = 1$.

Construction of the approximate suffix automaton can be done in four steps using ε-transitions based method:

1. Create a finite automaton accepting string *aabb* and all strings having Levenshtein distance $k = 1$ from it. Its transition diagram is depicted in Fig. 7.19.
2. Insert ε-transitions from the initial state leading into all states on the zero level (states 1, 2, 3 and 4). The result of this step is a nondeterministic finite automaton accepting all approximate suffixes of string *aabb*. Its transition diagram is depicted in Fig. 7.20.
3. Remove ε-transitions from the automaton created in the previous step. The result is a nondeterministic finite automaton without ε-transitions having transition diagram depicted in Fig. 7.21.
4. Transform the nondeterministic finite automaton to a deterministic one using the subset construction. Transition tables of both (nondeterministic and deterministic) automata are shown in Table 7.5. The result is a deterministic finite (suffix) automaton having transition diagram shown in Fig. 7.22. □

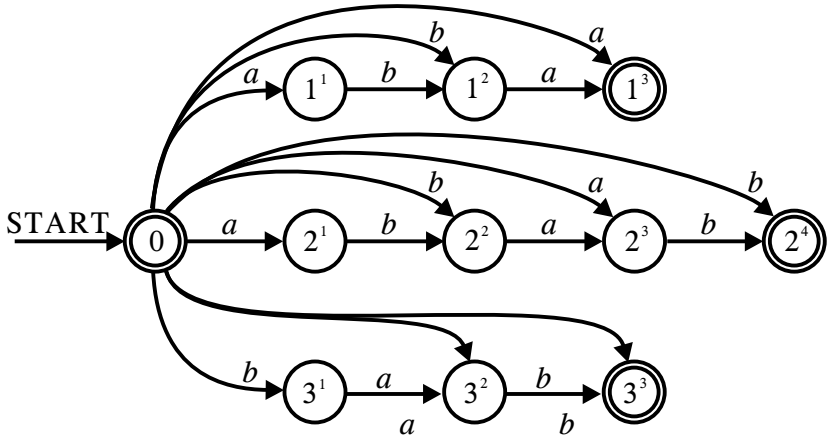

Figure 7.13: Transition diagram of the nondeterministic finite automaton without ε-transitions accepting all suffixes of set S from Exercise 7.6 step 3

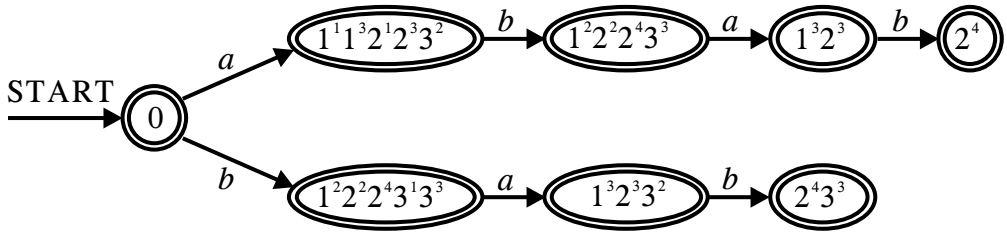

Figure 7.14: Transition diagram of the deterministic suffix automaton accepting all suffixes of set of strings $S = \{aba, abab, bab\}$ from Exercise 7.6

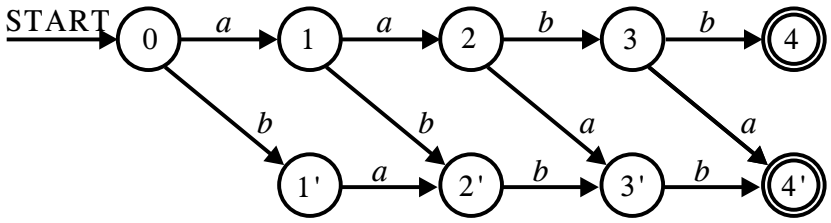

Figure 7.15: Transition diagram of the deterministic finite automaton accepting string *aabb* and all strings having Hamming distance $k = 1$ from it from Exercise 7.7

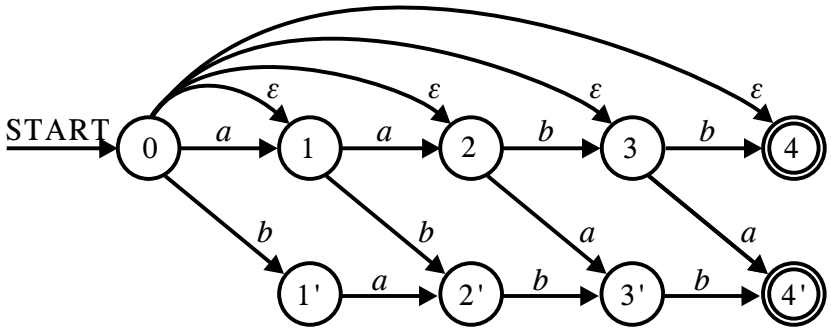

Figure 7.16: Transition diagram of the nondeterministic finite automaton accepting all approximate suffixes of string *aabb* from Exercise 7.7

	a	b
0	$1^1, 1^3, 2^1, 2^3, 3^2$	$1^2, 2^2, 2^4, 3^1, 3^3$
1^1		1^2
1^2	1^3	
1^3		
2^1		2^2
2^2	2^3	
2^3		2^4
2^4		
3^1	3^2	
3^2		3^3
3^3		

a)

	a	b
0	$1^1 1^3 2^1 2^3 3^2$	$1^2 2^2 2^4 3^1 3^3$
$1^1 1^3 2^1 2^3 3^2$		$1^2 2^2 2^4 3^3$
$1^2 2^2 2^4 3^1 3^3$	$1^3 2^3 3^2$	
$1^2 2^2 2^4 3^3$	$1^3 2^3$	
$1^3 2^3 3^2$		$2^4 2^3$
$1^3 2^3$		2^4
$2^4 2^3$		
2^4		

b)

Table 7.3: Transition tables of nondeterministic and deterministic suffix automata from Exercise 7.6

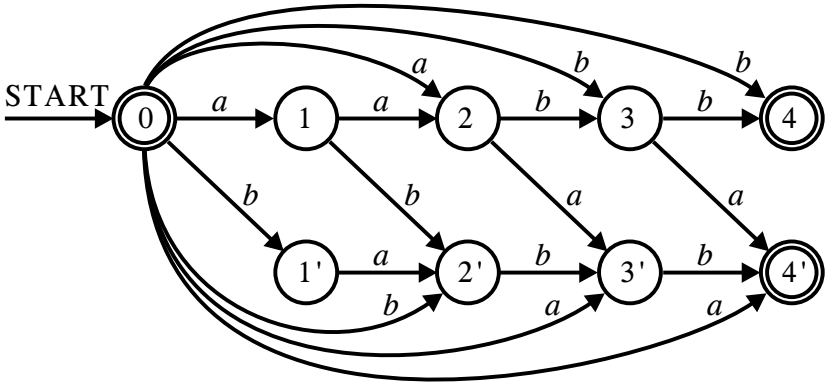

Figure 7.17: Transition diagram of the nondeterministic finite automaton without ε-transitions accepting all approximate suffixes of string *aabb* from Exercise 7.7

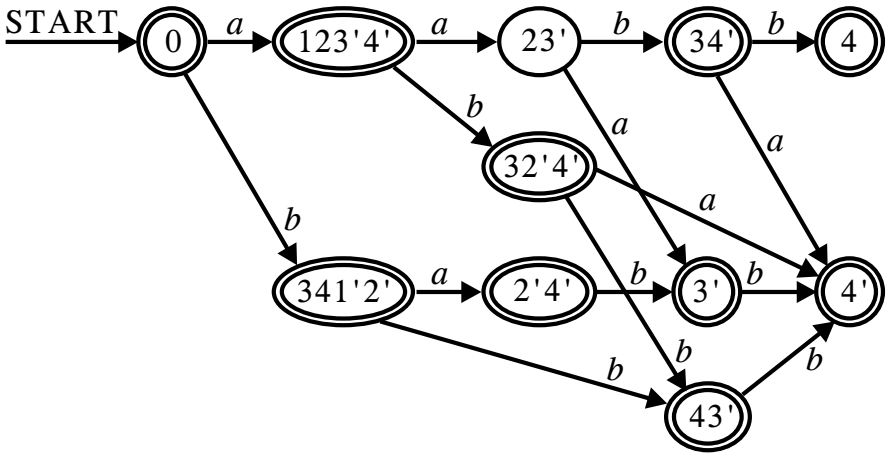

Figure 7.18: Transition diagram of the deterministic approximate suffix automaton accepting $ASuff(aabb)$ from Exercise 7.7

	a	b
0	$1, 2, 3', 4'$	$3, 4, 1', 2'$
1	2	$2'$
2	$3'$	3
3	$4'$	4
4		
$1'$	$2'$	
$2'$		$3'$
$3'$		$4'$
$4'$		

a)

	a	b
0	$123'4'$	$341'2'$
$123'4'$	$23'$	$32'4'$
$341'2'$	$2'4'$	$43'$
$23'$	$3'$	$34'$
$32'4'$	$4'$	$43'$
$2'4'$		$3'$
$43'$		$4'$
$3'$		$4'$
$34'$	$4'$	4
$4'$		
4		

b)

Table 7.4: Transition tables of both nondeterministic and deterministic approximate suffix automata for string *aabb* from Exercise 7.7

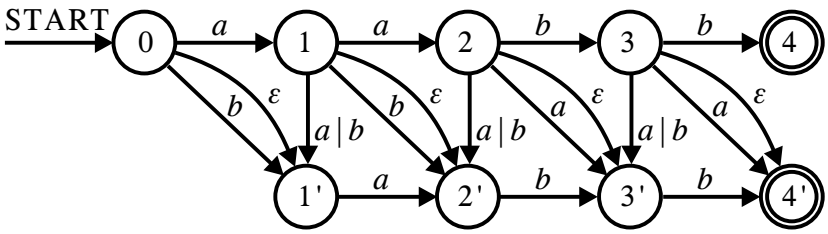

Figure 7.19: Transition diagram of the finite automaton accepting string *aabb* and all strings having Levenshtein distance $k = 1$ from it from Exercise 7.8

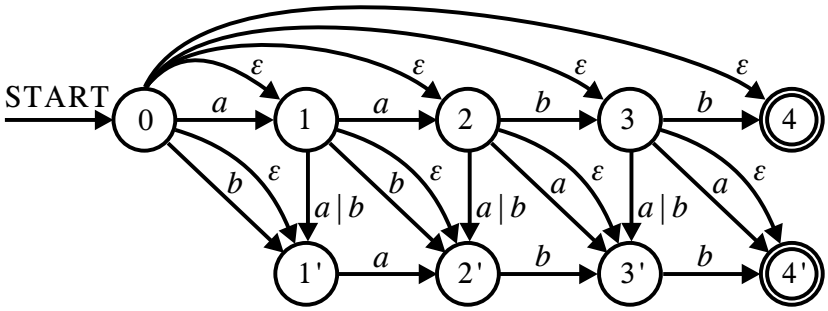

Figure 7.20: Transition diagram of the nondeterministic finite automaton accepting all approximate suffixes of string *aabb* from Exercise 7.8

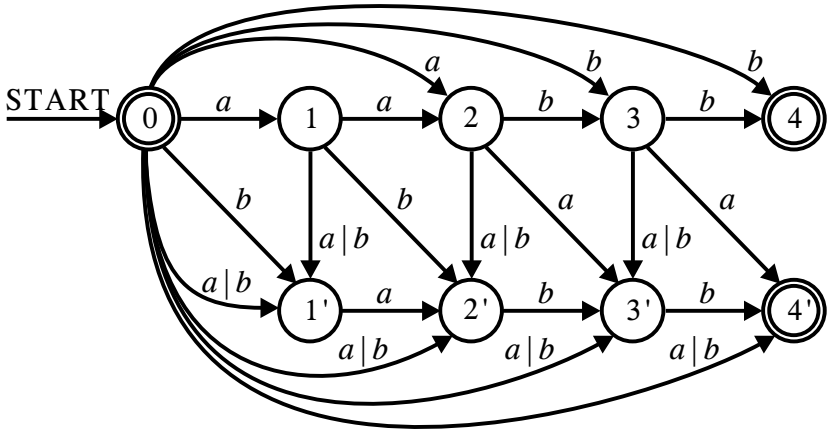

Figure 7.21: Transition diagram of the nondeterministic finite automaton without ε-transitions accepting all approximate suffixes of string *aabb* from Exercise 7.8

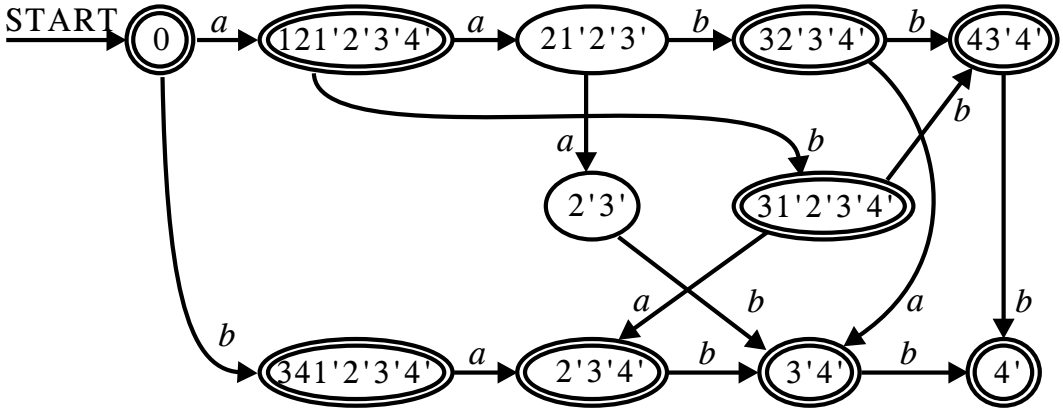

Figure 7.22: Transition diagram of the deterministic approximate suffix automaton accepting $ASuff(aabb)$ from Exercise 7.8

	a	b
0	$1, 2, 1', 2', 3', 4'$	$3, 4, 1', 2', 3', 4'$
1	$2, 1'$	$1', 2'$
2	$2', 3'$	$3, 2'$
3	$3', 4'$	$4, 3'$
4		
$1'$	$2'$	
$2'$		$3'$
$3'$		$4'$
$4'$		

a)

	a	b
0	$121'2'3'4'$	$341'2'3'4'$
$121'2'3'4'$	$21'2'3'$	$31'2'3'4'$
$341'2'3'4'$	$2'3'4'$	$43'4'$
$21'2'3'$	$2'3'$	$32'3'4'$
$31'2'3'4'$	$2'3'4'$	$43'4'$
$2'3'4'$		$3'4'$
$43'4'$		$4'$
$2'3'$		$3'4'$
$32'3'4'$	$3'4'$	$43'4'$
$3'4'$		$4'$
$4'$		

b)

Table 7.5: Transition tables of both nondeterministic and deterministic approximate suffix automata for string *aabb* from Exercise 7.8

8 Factor, factor oracle and subsequence automata

8.1 Factor automata

Exercise 8.1
Create a deterministic factor automaton for a text $T = abbcbbd$.

The construction of the factor automaton can be done in four steps using ε-transitions based method.

1. Create a finite automaton M_1 accepting all prefixes of text T. Its transition diagram is depicted in Figure 8.1.

Figure 8.1: Transition diagram of the deterministic finite automaton accepting all prefixes of the string *abbcbbd* from Exercise 8.1 step 1

2. Insert ε-transitions from the starting state leading into all other states except the last one. Result of this step is an automaton having transition diagram shown in Figure 8.2.

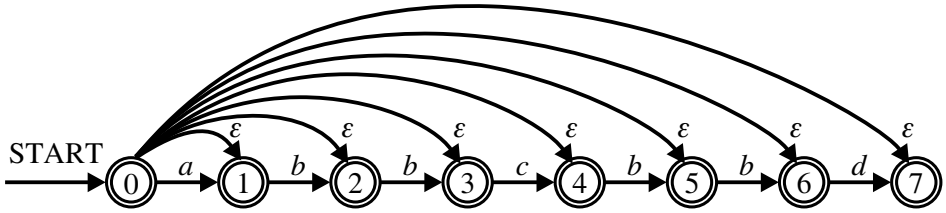

Figure 8.2: Transition diagram of the nondeterministic finite automaton accepting all factors of the string *abbcbbd* from Exercise 8.1 step 2

3. Remove all ε-transitions from the automaton created in the previous step. The result is a nondeterministic finite automaton without ε-transitions having transition diagram shown in Figure 8.3.

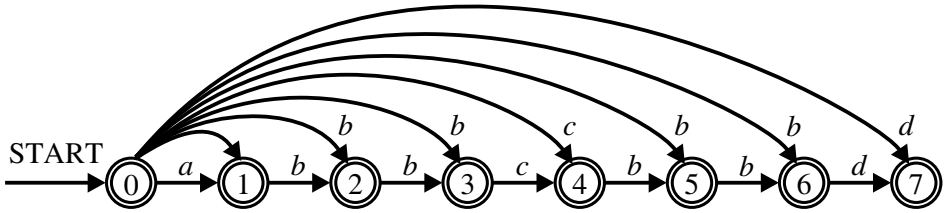

Figure 8.3: Transition diagram of the nondeterministic finite automaton without ε-transitions accepting all factors of the string *abbcbbd* from Exercise 8.1 step 3

4. Transform this automaton to a deterministic one using subset construction. Transition tables of both (nondeterministic and deterministic) automata are shown in the Table 8.1. The result is a deterministic finite automaton having transition diagram shown in Figure 8.4. □

	a	b	c	d
0	1	$2, 3, 5, 6$	4	7
1		2		
2		3		
3			4	
4		5		
5		6		
6				7
7				

	a	b	c	d
0	1	2356	4	7
1		2		
2		3		
2356		36	4	7
3			4	
36			4	7
4		5		
5		6		
6				7
7				

Table 8.1: Transition tables of both nondeterministic a deterministic finite automata from Exercise 8.1

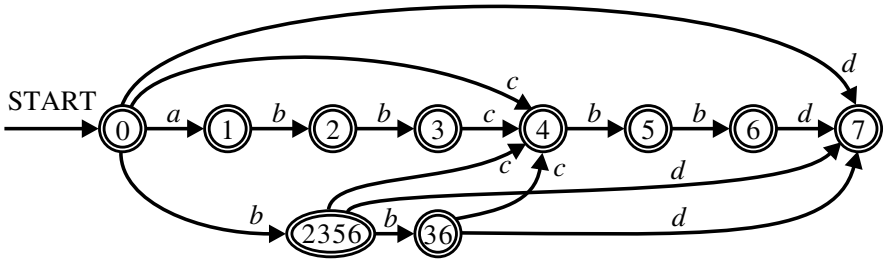

Figure 8.4: Transition diagram of the deterministic finite automaton accepting all factors of the string *abbcbbd* from Exercise 8.1 step 8.3

Exercise 8.2
Create a deterministic factor automaton for the same text as in Exercise 8.1 ($T = abbcbbd$). In this case, use the method based on a finite automaton with more initial states.

1. The construction starts with the same automaton, accepting all prefixes of the given text, as in previous case – Figure 8.1.
2. Modify this automaton like that all states are initial states, as it it is shown in Figure 8.5.

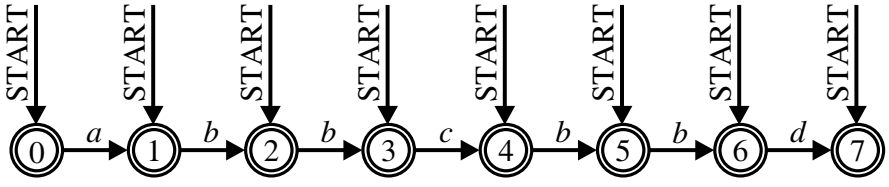

Figure 8.5: Transition diagram of the nondeterministic finite automaton with eight initial states accepting all prefixes of the string *abbcbbd* from Exercise 8.2 step 2

3. Transform this automaton to a deterministic one using subset construction. Transition tables of both (nondeterministic and deterministic) automata are shown in Table 8.2.

□

Anyone can see that both deterministic automata, created in this and previous example, are the same (except the name of the initial state). The transition diagram of this automaton

	a	b	c	d
0	1			
1		2		
2		3		
3			4	
4		5		
5		6		
6				7
7				

	a	b	c	d
0123456	1	2356	4	7
1		2		
2		3		
2356		36	4	7
3			4	
36			4	7
4		5		
5		6		
6				7
7				

Table 8.2: Transition tables of both nondeterministic and deterministic finite automata from Exercise 8.2

is shown in Figure 8.4.

Exercise 8.3
Create an approximate deterministic factor automaton for text $T = abbc$ over alphabet $A = \{a, b, c\}$ using Hamming distance with at most one error allowed.

In the first step, create a finite automaton accepting all exact an approximate prefixes of a given text. Its transition diagram is shown in Figure 8.6.

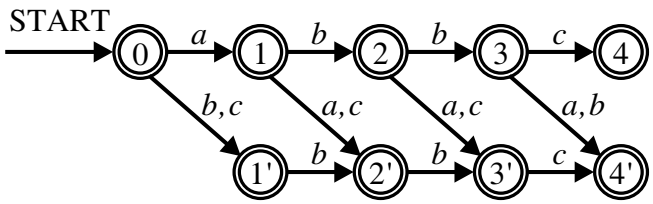

Figure 8.6: Transition diagram of the deterministic finite automaton accepting all approximate prefixes of the string *abbc* using Hamming distance with at most one error allowed from Exercise 8.3

After that, insert an ε-transitions from the initial state to the states $1, 2, 3, 4$. The result is the approximate nondeterministic factor automaton having transition diagram shown in Figure 8.7.

In the next step, remove the ε-transitions. It leads to the automaton having transition diagram shown in Figure 8.8.

In the last step, transform the automaton to the deterministic one using the subset construction. Transition tables of both (nondeterministic and deterministic) automata are shown in Table 8.3. The transition diagram of the deterministic automaton is shown in Figure 8.9. □

Exercise 8.4
Create an approximate deterministic factor automaton for text $T = aabba$ over alphabet $A = \{a, b\}$ using Levenshtein distance with at most one error allowed.

Construction of the factor automaton can be done in four steps using ε-transition based method.

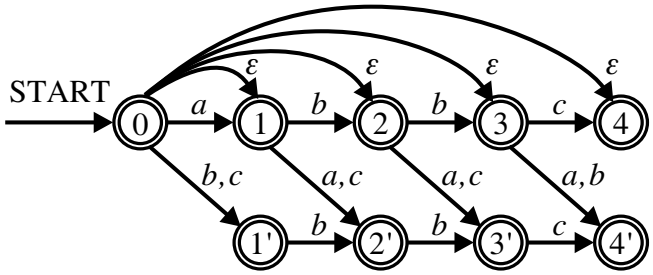

Figure 8.7: Transition diagram of the nondeterministic factor automaton with ε-transitions accepting all approximate factors of the string *abbc* using Hamming distance with at most one error allowed from Exercise 8.3

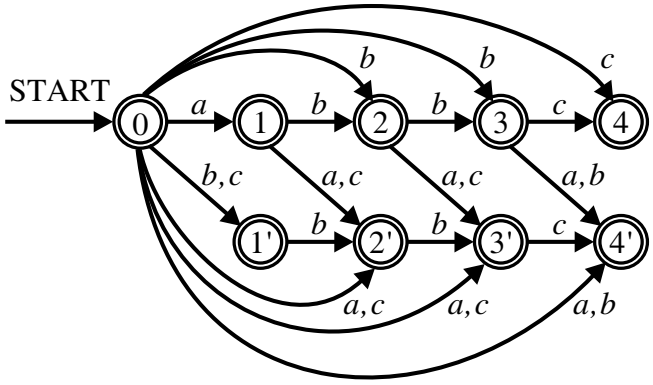

Figure 8.8: Transition diagram of the nondeterministic factor automaton accepting all approximate factors of the string *abbc* without ε-transitions using Hamming distance with at most one error allowed from Exercise 8.3

1. Create a finite automaton accepting all approximate prefixes of string *aabba* using Levenshtein distance $k = 1$ from it. Its transition diagram is shown in Figure 8.10.
2. Insert ε-transitions from the initial state leading into all states on the zero level (states 1,2,3,4,5). The result of this step is a nondeterministic finite automaton accepting all approximate factors of string *aabba*. Transition diagram of it is depicted in Figure 8.11.
3. Remove ε-transitions from the automaton created in previous step. The result is a nondeterministic finite automaton (factor automaton) without ε-transitions having transition diagram depicted in Fig. 8.12.
4. Transform the nondeterministic factor automaton to a deterministic one using the subset construction. Transition tables of both nondeterministic and deterministic factor automata are shown in Table 8.4.

It can be seen from the transition table (see Table 8.4 b) that sets of states (52'3'4'5', 2'3'4'5'), (43'4'5', 43'4'), (3'4'5', 3'4') and (54'5', 4'5',4') are equivalent and therefore minimised deterministic factor automaton has transition diagram depicted in Fig. 8.13. □

Exercise 8.5
Create an approximate deterministic factor automaton for string $T = abbc$ over ordered alphabet $A = \{a, b, c\}$ using Δ distance with at most one error allowed.

	a	b	c
0	$12^1 3^1 4^1$	$231^1 4^1$	$41^1 2^1 3^1$
1	2^1	2	2^1
2	3^1	3	3^1
3	4^1	4^1	4
4			
1^1		2^1	
2^1		3^1	
3^1			4^1
4^1			

	a	b	c
0	$12^1 3^1 4^1$	$231^1 4^1$	$41^1 2^1 3^1$
$12^1 3^1 4^1$	2^1	23^1	$2^1 4^1$
$231^1 4^1$	$3^1 4^1$	$32^1 4^1$	43^1
$41^1 2^1 3^1$		$2^1 3^1$	4^1
23^1	3^1	3	$3^1 4^1$
$32^1 4^1$	4^1	$3^1 4^1$	4
43^1			4^1
4			
2^1		3^1	
$2^1 4^1$		3^1	
$3^1 4^1$			4^1
$2^1 3^1$		3^1	4^1
3	4^1	4^1	4
3^1			4^1
4^1			

Table 8.3: Transition tables of both nondeterministic and deterministic factor automata from Exercise 8.3

	a	b
0	$1, 2, 5, 1', 2', 3', 4', 5'$	$3, 4, 1', 2', 3', 4', 5'$
1	$2, 1'$	$1', 2'$
2	$2', 3'$	$3, 2'$
3	$3', 4'$	$4, 3'$
4	$5, 4'$	$4', 5'$
5		
$1'$	$2'$	
$2'$		$3'$
$3'$		$4'$
$4'$	$5'$	
$5'$		

a)

	a	b
0	$1251'2'3'4'5'$	$341'2'3'4'5'$
$1251'2'3'4'5'$	$21'2'3'5'$	$31'2'3'4'$
$21'2'3'5'$	$2'3'$	$32'3'4'$
$341'2'3'4'5'$	$52'3'4'5'$	$43'4'5'$
$31'2'3'4'$	$2'3'4'5'$	$43'4'$
$32'3'4'$	$3'4'5'$	$43'4'$
$43'4'5'$	$54'5'$	$4'5'$
$43'4'$	$54'5'$	$4'5'$
$52'3'4'5'$	$5'$	$3'4'$
$2'3'4'5'$	$5'$	$3'4'$
$54'5'$	$5'$	
$4'5'$	$5'$	
$4'$	$5'$	
$2'3'$		$3'4'$
$3'4'5'$	$5'$	$4'$
$3'4'$	$5'$	$4'$
$5'$		

b)

Table 8.4: Transition tables of both nondeterministic and deterministic factor automata for string *aabba* from Exercise 8.4

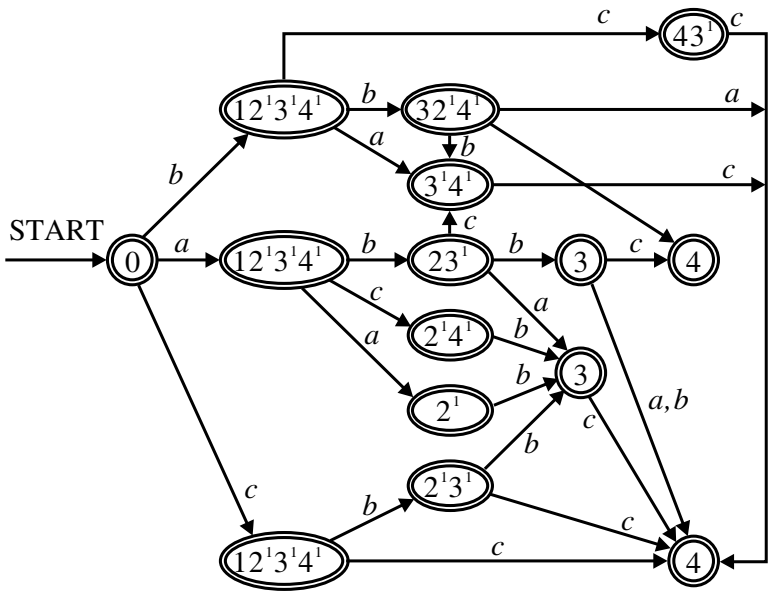

Figure 8.9: Transition diagram of the deterministic approximate factor automaton for the string *abbc* using Hamming distance with at most one error allowed from Exercise 8.3

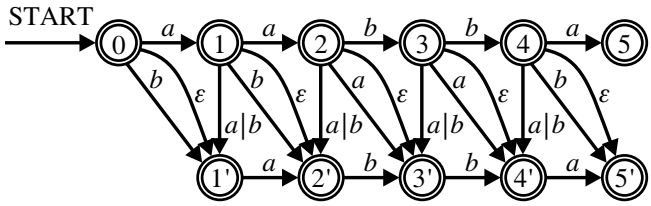

Figure 8.10: Transition diagram of finite automaton accepting all approximate prefixes of string *aabba* using Levenshtein distance $k = 1$ from Exercise 8.4

The construction of the factor automaton can be done in four steps using ε-transitions based method.

1. Create a finite automaton accepting all approximate prefixes of string $T = abbc$ using Δ distance $k = 1$ from it. Its transition diagram is depicted in Figure 8.14.
2. Insert ε-transitions from the initial state leading into states $\{1, 2, 3, 4\}$. The result of this step is a nondeterministic finite automaton accepting all approximate factors of string $T = abbc$. Transition diagram of it is depicted in Figure 8.15.
3. Remove all ε-transitions from the automaton created in the previous step. The result is a nondeterministic finite automaton (factor automaton) without ε-transitions having transition diagram depicted in Figure 8.16.
4. Transform this automaton to a deterministic one using the subset construction. Transition tables of both (nondeterministic and deterministic) factor automata are shown in the Table 8.5. The result is the deterministic factor automaton having transition diagram shown in Fig. 8.17.

   It can be seen from transition table (Table 8.5 b) that set of states {42'3', 2'3', 2'3'4'}, {3', 3'4', 43'} and {4,4'} are equivalent and therefore the minimised deterministic factor automaton has transition diagram depicted in Fig. 8.18. □

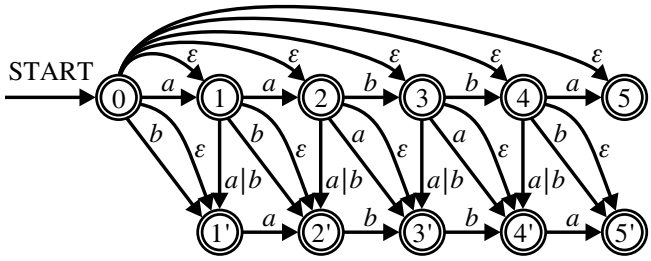

Figure 8.11: Transition diagram of the nondeterministic finite automaton accepting all approximate factors of string *aabba* from Exercise 8.4

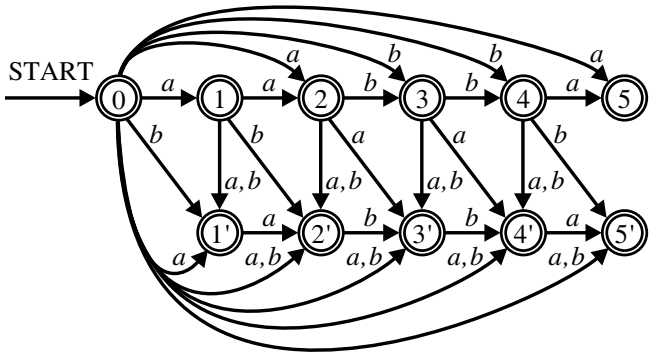

Figure 8.12: Transition diagram of the nondeterministic factor automaton without ε-transitions accepting all approximate factors of string *aabba* from Exercise 8.4

Exercise 8.6
Create an approximate deterministic factor automaton for string $T = abbc$ over ordered alphabet $A = \{a, b, c\}$ using Γ distance with at most one error allowed.

The construction of the factor automaton can be done in four steps using ε-transitions based method.

1. Create a finite automaton accepting all approximate prefixes of string $T = abbc$ using Γ distance $k = 1$ from it. Its transition diagram is depicted in Figure 8.19.
2. Insert ε-transitions from the initial state leading into states $\{1, 2, 3, 4\}$. The result of this step is a nondeterministic finite automaton accepting all approximate factors of string $T = abbc$. Transition diagram of it is depicted in Figure 8.20.
3. Remove all ε-transitions from the automaton created in the previous step. The result is a nondeterministic finite automaton (factor automaton) without ε-transitions having transition diagram depicted in Figure 8.21.
4. Transform this automaton to a deterministic one using the subset construction. Transition tables of both (nondeterministic and deterministic) factor automata are shown in the Table 8.6. The result is the deterministic factor automaton having transition diagram shown in Fig. 8.22.

   It can be seen from transition table (Table 8.6 b) that set of states {2', 2'4'}, {3', 3'4', 43'} and {4,4'} are equivalent and therefore the minimised deterministic factor automaton has transition diagram depicted in Fig. 8.23. □

	a	b	c
0	$1, 2', 3'$	$2, 3, 1', 4'$	$4, 2', 3'$
1	$2'$	2	$2'$
2	$3'$	3	$3'$
3		$4'$	4
4			
$1'$	$2'$	$2'$	$2'$
$2'$	$3'$	$3'$	$3'$
$3'$		$4'$	$4'$
$4'$			

a)

	a	b	c
0	$12'3'$	$231'4'$	$42'3'$
$12'3'$	$2'3'$	$23'4'$	$2'3'4'$
$231'4'$	$2'3'$	$32'4'$	$42'3'$
$23'4'$	$3'$	$34'$	$3'4'$
$32'4'$	$3'$	$3'4'$	$43'$
$34'$		$4'$	4
$42'3'$	$3'$	$3'4'$	$3'4'$
$2'3'$	$3'$	$3'4'$	$3'4'$
$2'3'4'$	$3'$	$3'4'$	$3'4'$
$3'$		$4'$	$4'$
$3'4'$		$4'$	$4'$
$43'$		$4'$	$4'$
4			
$4'$			

b)

Table 8.5: Transition tables of both nondeterministic and deterministic factor automata for string *abbc* from Exercise 8.5

	a	b	c
0	$1, 2', 3'$	$2, 3, 1', 4'$	$4, 2', 3'$
1	$2'$	2	$2'$
2	$3'$	3	$3'$
3		$4'$	4
4			
$1'$		$2'$	
$2'$		$3'$	
$3'$			$4'$
$4'$			

a)

	a	b	c
0	$12'3'$	$231'4'$	$42'3'$
$12'3'$	$2'$	$23'$	$2'4'$
$231'4'$	$3'$	$32'4'$	$43'$
$23'$	$3'$	3	$3'4'$
$32'4'$		$3'4'$	4
3		$4'$	4
4			
$4'$			
$2'$		$3'$	
$2'4'$		$3'$	
$3'$			$4'$
$3'4'$			$4'$
$43'$			$4'$
$42'3'$		$3'$	$4'$

b)

Table 8.6: Transition tables of both nondeterministic and deterministic factor automata for string *abbc* from Exercise 8.6

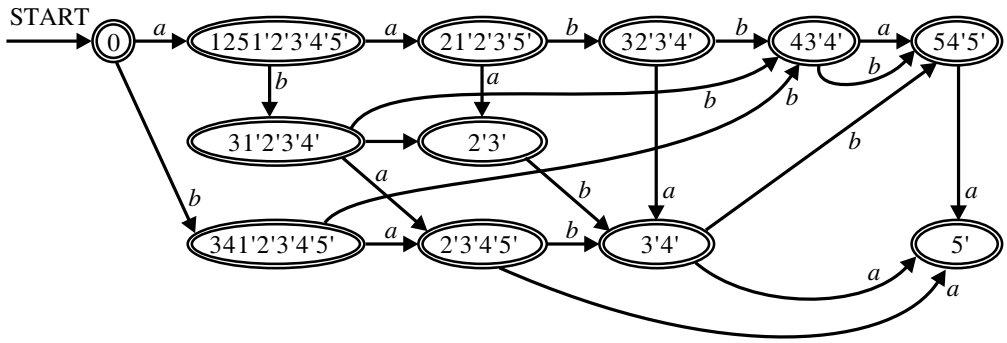

Figure 8.13: Transition diagram of the optimised deterministic factor automaton $AFact(aabba)$ from Exercise 8.4

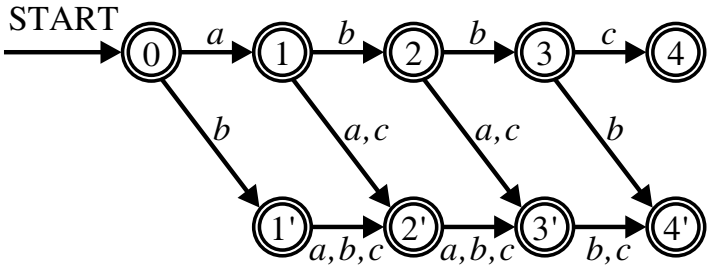

Figure 8.14: Transition diagram of the finite automaton accepting all approximate prefixes of string $abbc$ using Δ distance $k = 1$ from Exercise 8.5

Exercise 8.7
Create an approximate deterministic factor automaton for string $T = abbc$ over ordered alphabet $A = \{a, b, c\}$ using (Δ, Γ) distance with $(k, l) = (1, 2)$.

The construction of the factor automaton can be done in four steps using ε-transitions based method.

1. Create a finite automaton accepting all approximate prefixes of string $T = abbc$ using (Δ, Γ) distance (1,2) from it. Its transition diagram is depicted in Figure 8.24.
2. Insert ε-transitions from the initial state leading into states $\{1, 2, 3, 4\}$. The result of this step is a nondeterministic finite automaton accepting all approximate factors of string $T = abbc$. Transition diagram of it is depicted in Figure 8.25.
3. Remove all ε-transitions from the automaton created in the previous step. The result is a nondeterministic finite automaton (factor automaton) without ε-transitions having transition diagram depicted in Figure 8.26.
4. Transform this automaton to a deterministic one using the subset construction. Transition tables of both (nondeterministic and deterministic) factor automata are shown in the Table 8.7. The result is the deterministic factor automaton having transition diagram shown in Fig. 8.27.

   It can be seen from transition table (Table 8.7 b) that set of states {43'2", 3'2"}, {2'3", 2'4'3"}, {3',3'4', 3'4"}, {3", 4'3", 3"4", 43"} and {4,4', 4"} are equivalent and therefore the minimised deterministic factor automaton has transition diagram depicted in Fig. 8.28. □

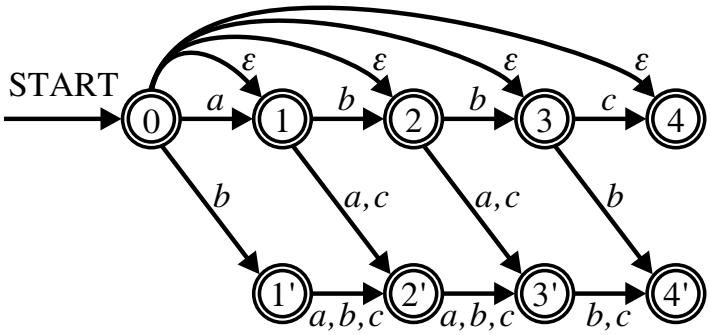

Figure 8.15: Transition diagram of the nondeterministic finite automaton accepting all approximate factors of the string *abbc* from Exercise 8.5

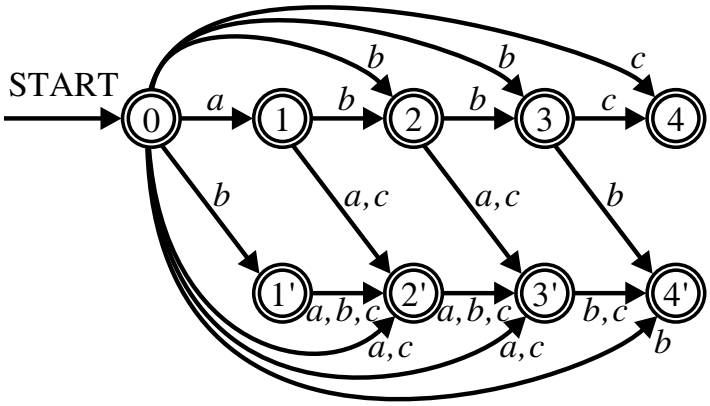

Figure 8.16: Transition diagram of the nondeterministic factor automaton without ε-transitions accepting all approximate factors of string *abbc* from Exercise 8.5

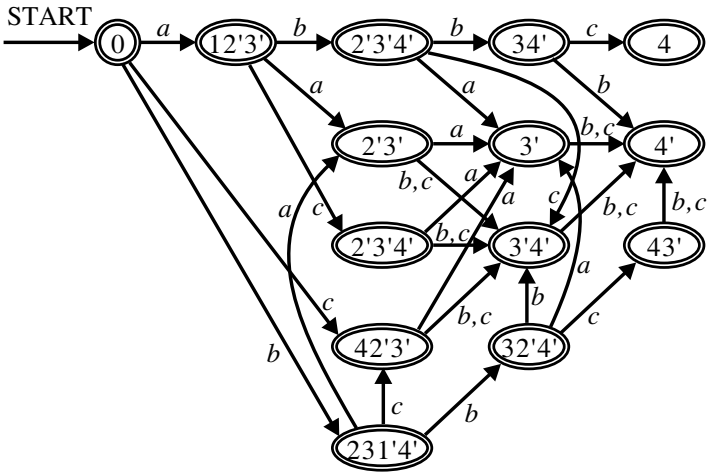

Figure 8.17: Transition diagram of the deterministic factor automaton from Exercise 8.5

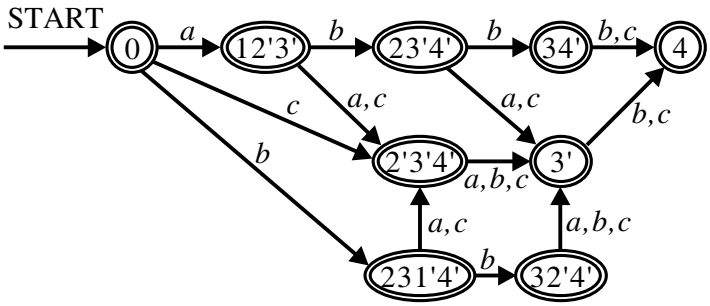

Figure 8.18: Transition diagram of the minimised deterministic factor automaton from Exercise 8.5

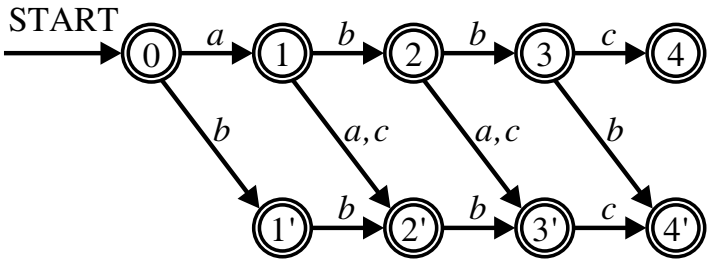

Figure 8.19: Transition diagram of the finite automaton accepting all approximate prefixes of string *abbc* using Γ distance $k = 1$ from Exercise 8.6

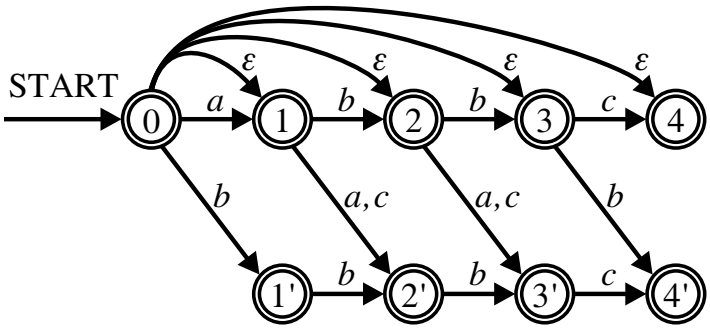

Figure 8.20: Transition diagram of the nondeterministic finite automaton accepting all approximate factors of the string *abbc* from Exercise 8.6

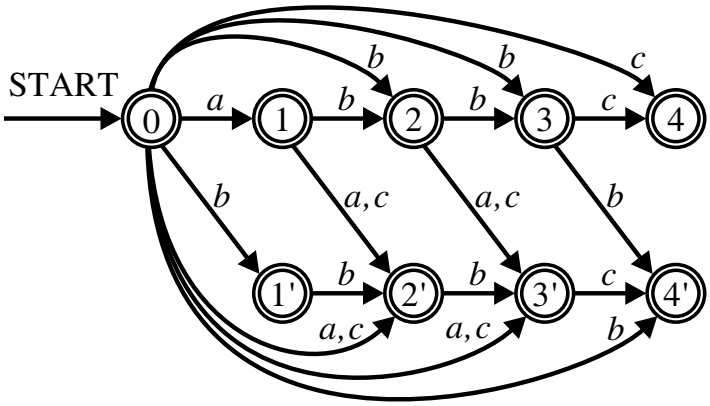

Figure 8.21: Transition diagram of the nondeterministic factor automaton without ε-transitions accepting all approximate factors of string *abbc* from Exercise 8.6

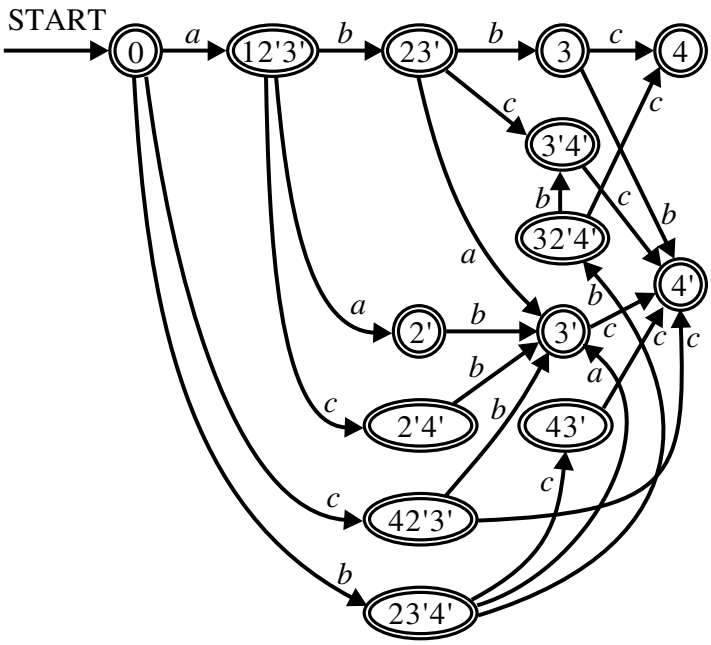

Figure 8.22: Transition diagram of the deterministic factor automaton from Exercise 8.6

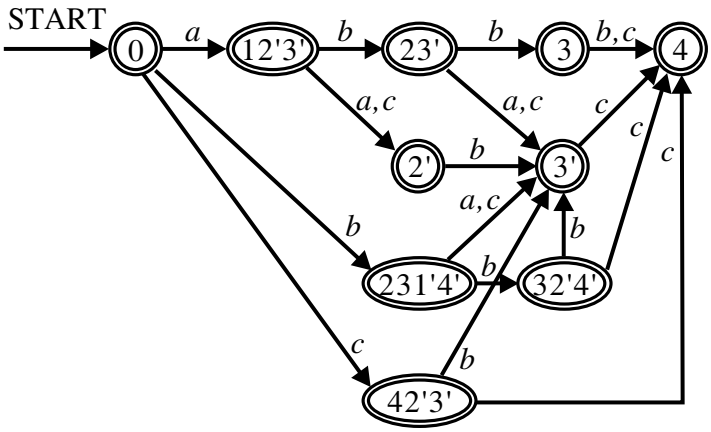

Figure 8.23: Transition diagram of the minimised deterministic factor automaton from Exercise 8.6

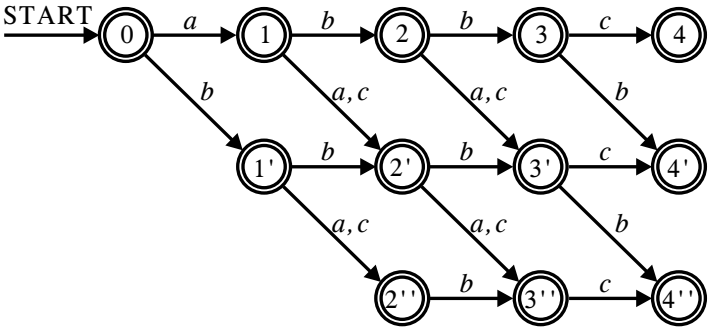

Figure 8.24: Transition diagram of the finite automaton accepting all approximate prefixes of string *abbc* using (Δ, Γ) distance (1,2) from Exercise 8.7

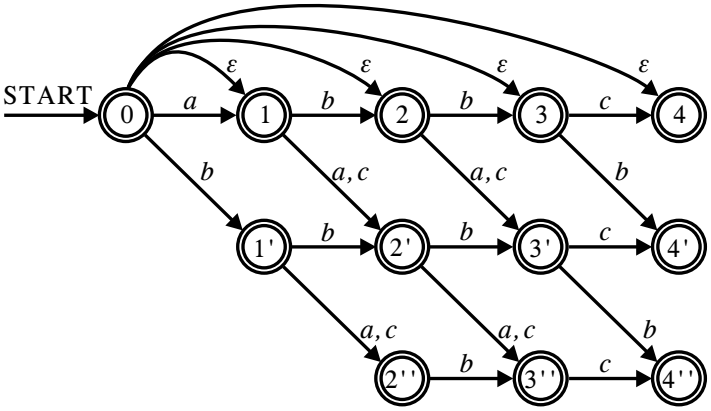

Figure 8.25: Transition diagram of the nondeterministic finite automaton accepting all approximate factors of the string *abbc* from Exercise 8.7

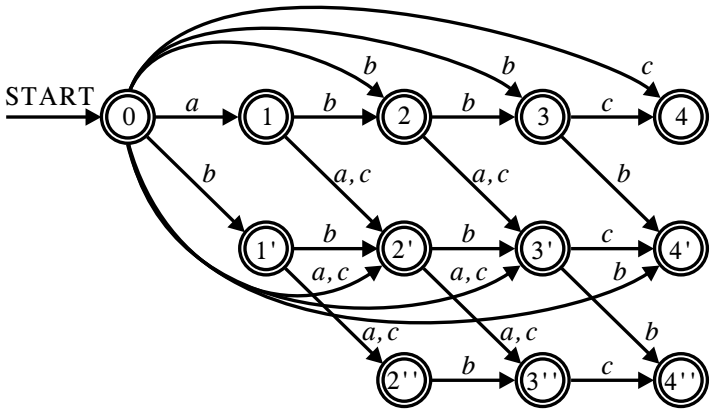

Figure 8.26: Transition diagram of the nondeterministic factor automaton without ε-transitions accepting all approximate factors of string *abbc* from Exercise 8.7

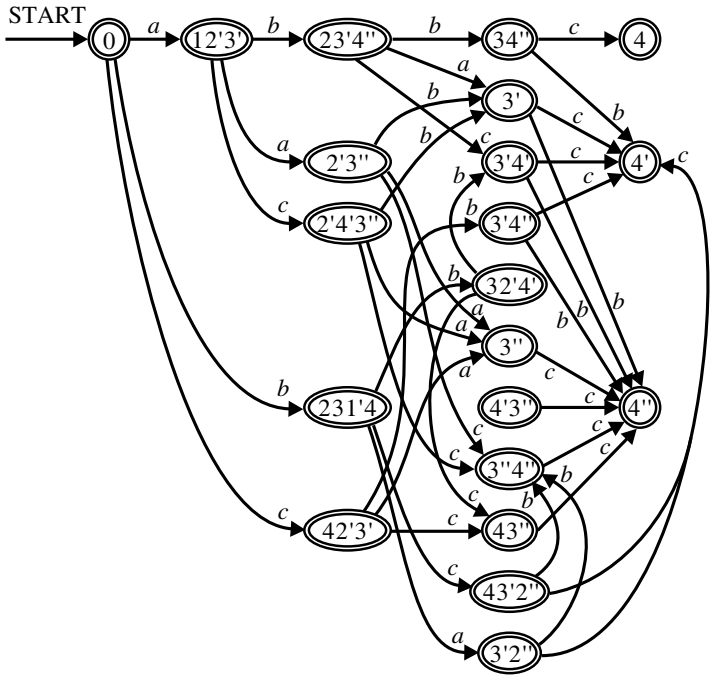

Figure 8.27: Transition diagram of the deterministic factor automaton from Exercise 8.7

	a	b	c
0	$1, 2', 3'$	$2, 3, 1', 4'$	$4, 2', 3'$
1	$2'$	2	$2'$
2	$3'$	3	$3'$
3		$4'$	4
4			
$1'$	$2''$	$2'$	$2''$
$2'$	$3''$	$3'$	$3''$
$3'$		$4''$	$4'$
$4'$			
$2''$		$3''$	
$3''$			$4''$
$4''$			

a)

	a	b	c
0	$12'3'$	$231'4'$	$42'3'$
$12'3'$	$2'3''$	$23'4''$	$2'4'3''$
$231'4'$	$3'2''$	$32'4'$	$43'2''$
$23'4''$	$3'$	$34''$	$3'4'$
$32'4'$	$3''$	$3'4'$	$43''$
$34''$		$4'$	4
$42'3'$	$3''$	$3'4''$	$4'3''$
$43'2''$		$3''4''$	$4'$
$43''$			$4''$
4			
$2'3''$	$3''$	$3'$	$3''4''$
$2'4'3''$	$3''$	$3'$	$3''4''$
$3'2''$		$3''4''$	$4'$
$3'$		$4''$	$4'$
$3'4'$		$4''$	$4'$
$3'4''$		$4''$	$4'$
$4'$			
$4'3''$			$4''$
$3''$			$4''$
$3''4''$			$4''$
$4''$			

b)

Table 8.7: Transition tables both of nondeterministic and deterministic factor automata for string *abbc* from Exercise 8.7

8.2 Factor oracle automata

We give some examples of various factor oracle automata.

Exercise 8.8
Let us construct factor oracle automaton for text $T = abbbaab$. The nondeterministic factor automaton M_N has transition diagram depicted in Fig. 8.29. Transition diagram of factor oracle automaton $M_O(T)$ is shown in Fig. 8.30. This automaton accepts all factors of text T and some subsequences of T:

$aba, abaa, abba, abbaa$

which are not factors of T. □

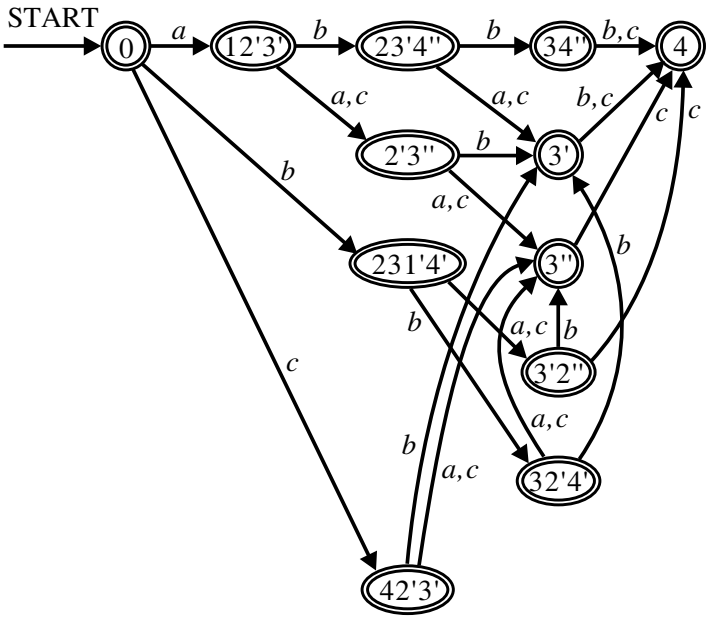

Figure 8.28: Transition diagram of the minimised deterministic factor automaton from Exercise 8.7

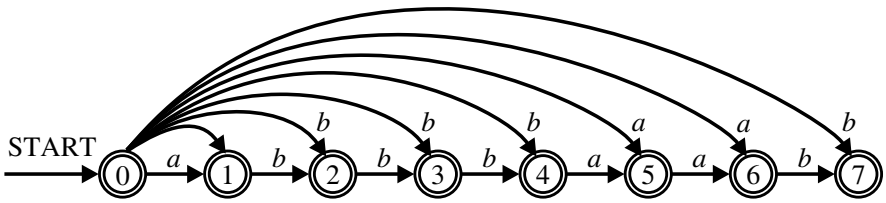

Figure 8.29: Transition diagram of nondeterministic factor automaton M_N from Exercise 8.8

Exercise 8.9
Let us construct factor oracle automaton for text $T = abbcabc$. The nondeterministic factor automaton $M_N(T)$ has transition diagram depicted in Fig. 8.31. Transition diagram of factor oracle automaton $M_O(T)$ is shown in Fig. 8.32. This automaton is reporting factor abc before its fist occurrence. □

Exercise 8.10
Let us construct factor oracle automaton for text $T = abbbabaaab$. The deterministic factor automaton $M_D(T)$ has transition diagram shown in Fig. 8.33. To construct factor oracle automaton M_O we identify pairs of corresponding states:

$(2346A, 26A), (34, 3), (57, 5)$, and $(89, 8)$.

Resulting automaton has transition diagram depicted in Fig. 8.34. The back end part of the transition diagram is shown in the same Figure when we select for merging states of automaton $M_D(T)$ pair (89,9) instead of pair (89,8). The resulting factor oracle automaton M'_O has less transitions than automaton M_O. This automaton is recognising factor abc before its fist occurrence. □

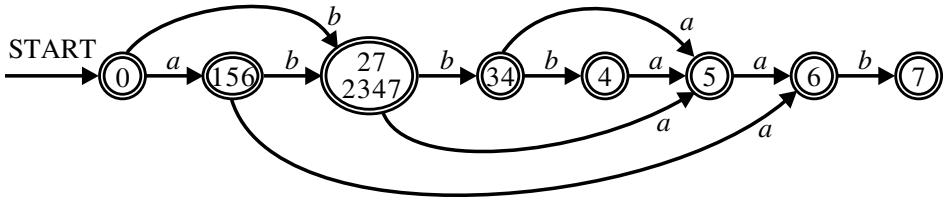

Figure 8.30: Transition diagram of factor oracle automaton $M_O(T)$ from Exercise 8.8

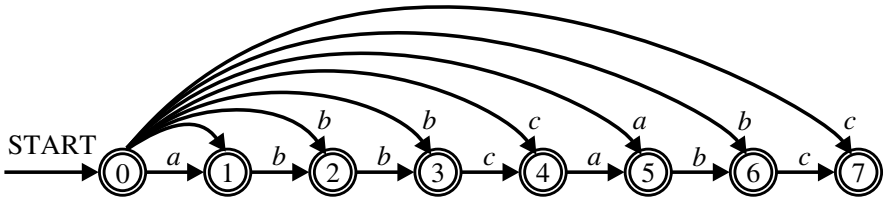

Figure 8.31: Transition diagram of nondeterministic factor automaton $M_N(T)$ from Exercise 8.9

8.3 Subsequence automata

Exercise 8.11
Create a deterministic subsequence automaton for a text $T = abbc$.

The construction of the subsequence automaton can be done in four steps using ε-transitions based method. It is based on a fact, that all subsequences of this text can be described by the following regular expression:

$$(a + \varepsilon)(b + \varepsilon)(b + \varepsilon)(c + \varepsilon)$$

1. Create a prefix automaton for the text T and add ε-transitions between each adjacent states. Its transition diagram is depicted in Figure 8.35.
2. Remove all ε-transitions from the automaton created in the previous step. The result is a nondeterministic subsequence automaton without ε-transitions, having transition diagram shown in Figure 8.36.
3. Transform the nondeterministic finite automaton to a deterministic one using subset construction. Transition tables of both (nondeterministic and deterministic) automata are described in Tables 8.8.

   The result is a deterministic subsequence automaton having transition diagram shown in Figure 8.37.

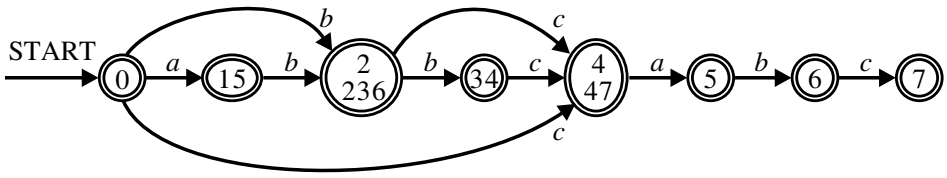

Figure 8.32: Transition diagram of factor oracle automaton $M_O(T)$ from Exercise 8.9

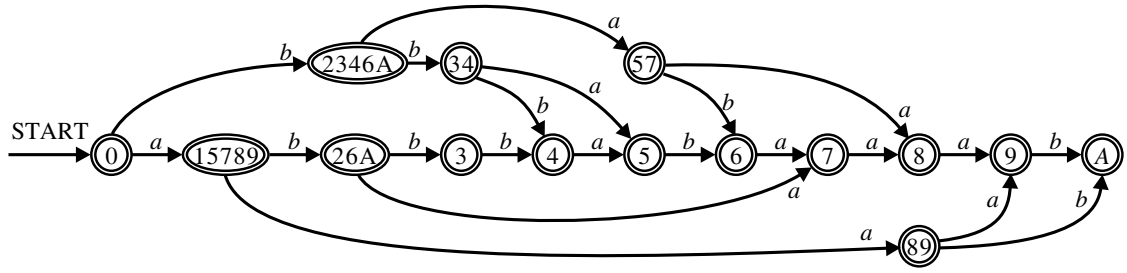

Figure 8.33: Transition diagram of deterministic factor automaton M_D from Exercise 8.10

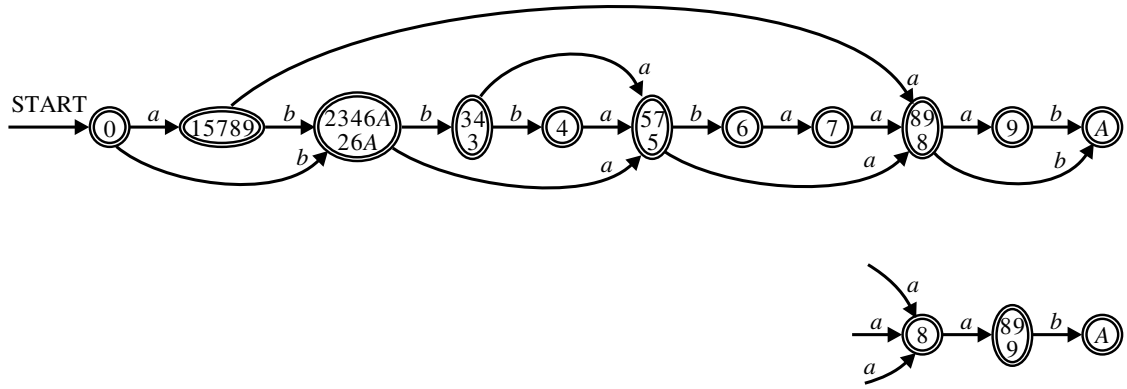

Figure 8.34: Transition diagram of factor oracle automaton M_O and fragment of transition diagram of M'_O from Exercise 8.10

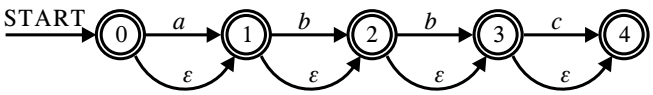

Figure 8.35: Transition diagram of the nondeterministic subsequence automaton with ε-transitions for the string *abbc* from Exercise 8.11 step 1

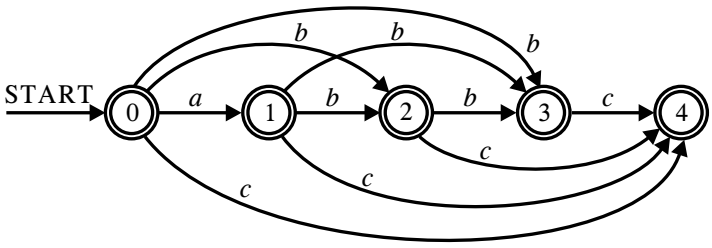

Figure 8.36: Transition diagram of the nondeterministic subsequence automaton without ε-transitions for the string *abbc* from Exercise 8.11 step 2

	a	b	c
0	1	23	4
1		23	4
2		3	4
3			4
4			

	a	b	c
0	1	23	4
1		23	4
23		3	4
3			4
4			

Table 8.8: Transition tables of both nondeterministic and deterministic subsequence automata from Exercise 8.11

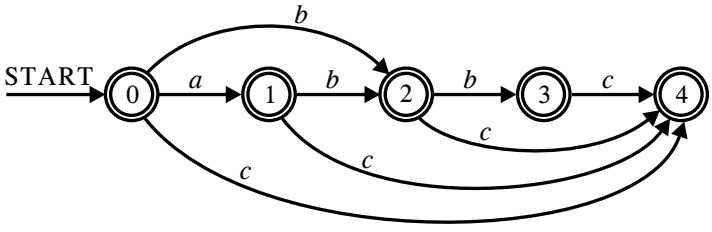

Figure 8.37: Transition diagram of the deterministic subsequence automaton for the string *abbc* from Exercise 8.11 step 3

9 Borders and border arrays

In this section, the computation of borders and border arrays is being done using the algorithms presented in tutorial [MHP, Chapter 4].

The first step of these algorithms is the construction of nondeterministic and consequently corresponding deterministic suffix or factor automaton for a given string x or a given set of strings S. Since the suffix automaton and the factor automaton constructed for the same string or the same set of strings differ only in that the suffix automaton has final only such states that correspond to suffixes and the factor automaton has final all the states we can build only suffix automaton for x or S, respectively, and use it for both for the computation of borders and for the computation of border array with that during computation of border array we do not take into account finality of the states of the automaton.

The second step of the algorithms is an extraction of backbone of just built deterministic suffix automaton. This can be done easily in the way we let the automaton process string x, the strings from S, respectively, and the states and the transtions used during processing lie on the backbone of the automaton. So, we remove all not used transitions and not visited states and obtain thus the backbone of the automaton.

The remaining steps differ according to problem being solved and are all based on analysis of d-subsets of states on the backbone. Therefore it is necessary to memorise d-subsets during determinisation of the suffix automaton.

Let us remind that *border* of a string $x \in A^+$ is every prefix of x, which is simultaneously its suffix. The set of all borders of the string x is denoted $bord(x)$ and the longest border of x is denoted $Border(x)$. The *border array* $\beta[1..n]$ of the string x is a vector of the length of the longest borders of all prefixes of x.

Moreover, *border of a set of strings* $S = \{x_1, x_2, \ldots, x_{|S|}\}$ is every prefix of some $x_i \in S$ which is the suffix of some $x_j \in S, i, j \in < 1, |S| >$. The set of all borders of the set S is denoted $mbord(S)$. The longest border which is the suffix of $x_i, i \in < 1, |S| >$ belongs to the set $mBorder(S)$. The *mborder* array $m\beta[1..n]$ of the set S is a vector of the longest borders of all prefixes of strings from S.

The *backbone* of the factor automaton M of a set of strings S is a part of a factor automaton M enabling sequences of transitions for all strings from the set S starting in the initial state and nothing else.

The *depth* of the state of the factor automaton on its backbone is the number of transitions which are necessary in order to reach the state q from the initial state.

Exercise 9.1
Let us have string $x = aabaabaa$ over alphabet $A = \{a, b\}$.

1. Compute set of all borders of string x $bord(x)$ and the longest border $Border(x)$ of string x.

2. Compute border array $\beta[1..|x|]$ of string x.

First, we construct nondeterministic suffix automaton for string x. This automaton has the transition table shown in Table 9.1 a) and the transition diagram depicted in Fig. 9.1.

Next, we determinise the automaton and thus we obtain deterministic suffix automaton which has the transition table shown in Table 9.1 b) and the transition diagram depicted in Fig. 9.2. During determinisation we memorise d-subsets. In the next step, we extract

	a	b
0	124578	36
1	2	
2		3
3	4	
4	5	
5		6
6	7	
7	8	
8		

a)

	a	b
0	124578	36
124578	258	36
258		36
36	47	
47	58	
58		6
6	7	
7	8	
8		

b)

Table 9.1: Transition tables of both the nondeterministic and deterministic suffix automata for $x = aabaabaa$ from Exercise 9.1

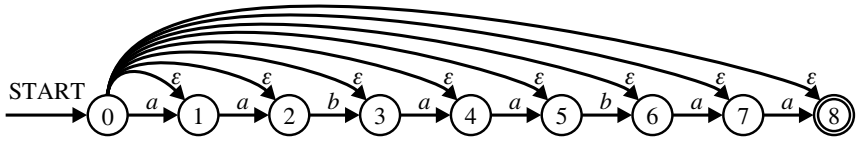

Figure 9.1: Transition diagram of the nondeterministic suffix automaton for $x = aabaabaa$ from Exercise 9.1

backbone of the deterministic suffix automaton in the way we process string x and remove all not used transitions ($0 \xrightarrow{b} 36$, $124578 \xrightarrow{b} 36$ drawn with dashed lines).

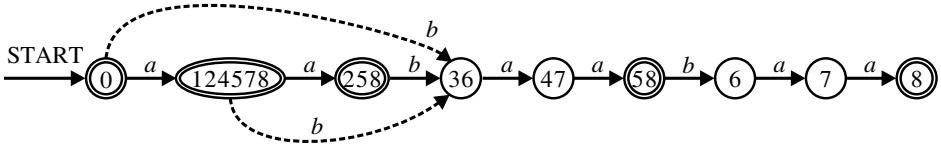

Figure 9.2: Transition diagram of the deterministic suffix automaton for $x = aabaabaa$ from Exercise 9.1

To compute all borders of x we search sequences of transition on the backbone leading from the initial state to some final state except the very last one. We find following sequences of transitions:

$0 \xrightarrow{\varepsilon} 0$
$0 \xrightarrow{a} 124578$
$0 \xrightarrow{a} 124578 \xrightarrow{a} 258$
$0 \xrightarrow{a} 124578 \xrightarrow{a} 258 \xrightarrow{b} 36 \xrightarrow{a} 47 \xrightarrow{a} 58$

By concatenating labels of transitions of each found sequence we obtain following strings ε, a, aa, $aabaa$. These strings are borders of x and the longest border is $aabaa$. So, we can write $bord(x) = \{\varepsilon, a, aa, aabaa\}$ and $Border(x) = aabaa$.

To compute border array of string x we do analysis of d-subsets of the states on the backbone. First, we create an empty array β of length $|x| = 6$ and initialise all its elements by 0. Next, we perform the analysis of the states on the backbone having d-subset with more than one item, what are states 124578, 258, 36, 47 and 58 in order from left to right and compute values of already not set elements of β.

When analysing the current d-subset we set elements of β on positions given by the items in the d-subset and at the same time containing 0 to value equal to the depth of the state (also expressed by the first item in the d-subset). The process of computation is shown in the next table:

Analysed state	Values of border array elements
124578	$\beta[2] = 1,\ \beta[4] = 1,\ \beta[5] = 1,\ \beta[7] = 1,\ \beta[8] = 1$
258	$\beta[5] = 2,\ \beta[8] = 2$
36	$\beta[6] = 3$
47	$\beta[7] = 4$
58	$\beta[8] = 5$

The resulting border array $\beta(aabaabaa)$ is summed up in the next table:

i	1	2	3	4	5	6	7	8
symbol	a	a	b	a	a	b	a	a
$\beta[i]$	0	1	0	1	2	3	4	5

Exercise 9.2
Let us have string $x = aaaaaaa$ over alphabet $A = \{a, b\}$.

1. Compute set of all borders $bord(x)$ of string x and the longest border $Border(x)$ of string x.
2. Compute border array $\beta[1..|x|]$ of string x.

First, we construct nondeterministic suffix automaton for string x. Next, we determinise the automaton and thus we obtain deterministic suffix automaton which has the transition diagram depicted in Fig. 9.3. During determinisation we memorise d-subsets. In the next step, we extract backbone of the deterministic suffix automaton in the way we process string x and remove all not used transitions and visited states. We see we use all transitions and visit all states when processing x, hence the backbone is the same as the automaton itself.

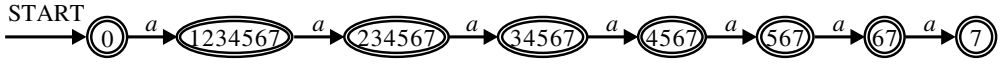

Figure 9.3: Transition diagram of the deterministic suffix automaton for $x = aaaaaaa$ from Exercise 9.2

To compute all borders of x we search sequences of transition on the backbone leading from the initial state to some final state except the very last one. We find following sequences of transitions:

$0 \xrightarrow{\varepsilon} 0$
$0 \xrightarrow{a} 1234567$
$0 \xrightarrow{a} 1234567 \xrightarrow{a} 234567$
$0 \xrightarrow{a} 1234567 \xrightarrow{a} 234567 \xrightarrow{a} 34567$
$0 \xrightarrow{a} 1234567 \xrightarrow{a} 234567 \xrightarrow{a} 34567 \xrightarrow{a} 4567$
$0 \xrightarrow{a} 1234567 \xrightarrow{a} 234567 \xrightarrow{a} 34567 \xrightarrow{a} 4567 \xrightarrow{a} 567$
$0 \xrightarrow{a} 1234567 \xrightarrow{a} 234567 \xrightarrow{a} 34567 \xrightarrow{a} 4567 \xrightarrow{a} 567 \xrightarrow{a} 67$

By concatenating labels of transitions of each found sequence we obtain following strings ε, a, aa, aaa, $aaaa$, $aaaaa$, $aaaaaa$ and $aaaaaaa$. These strings are borders of x and the longest one $aaaaaaa$ is the border of x. So, we can write $bord(x) = \{\varepsilon, a, aa, aaa, aaaa, aaaaa, aaaaaa\}$ and $Border(x) = aaaaaa$.

To compute border array of string x we do analysis of d-subsets of the states on the backbone. First, we create an empty array β of length $|x| = 7$ and initialise all its elements by 0. Next, we perform the analysis of the states on the backbone having d-subset with more than one item, what are states 1234567, 234567, 34567, 4567, 567 and 67 in order from left to right and compute values of β. The process of computation is shown in the next table:

Analysed state	Values of border array elements
1234567	$\beta[2] = 1, \beta[3] = 1, \beta[4] = 1, \beta[5] = 1, \beta[6] = 1, \beta[7] = 1$
234567	$\beta[3] = 2, \beta[4] = 2, \beta[5] = 2, \beta[6] = 2, \beta[7] = 2$
34567	$\beta[4] = 3, \beta[5] = 3, \beta[6] = 3, \beta[7] = 3$
4567	$\beta[5] = 4, \beta[6] = 4, \beta[7] = 4$
567	$\beta[6] = 5, \beta[7] = 5$
67	$\beta[7] = 6$

The resulting border array $\beta(abaababa)$ is summed up in the next table:

i	1	2	3	4	5	6	7
symbol	a	a	a	a	a	a	a
$\beta[i]$	0	1	2	3	4	5	6

Exercise 9.3
Let us have string $x = abaabaabaaba$ over alphabet $A = \{a, b\}$.

1. Compute set of all borders $bord(x)$ of string x and the longest border $Border(x)$ of string x.

2. Compute border array $\beta[1..|x|]$ of string x.

First, we construct nondeterministic suffix automaton for string x. Next, we determinise the automaton and thus we obtain deterministic suffix automaton which has the transition diagram depicted in Fig. 9.4 (X, Y and Z represent numbers 10, 11 and 12, respectively). During determinisation we memorise d-subsets. In the next step, we extract backbone of the deterministic suffix automaton in the way we process string x and remove all not used transitions ($0 \xrightarrow{b} 258Y$, $134679XZ \xrightarrow{b} 47X$ drawn with dashed lines).

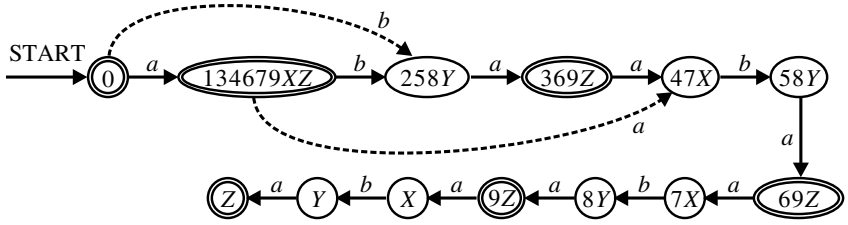

Figure 9.4: Transition diagram of the deterministic suffix automaton for $x = abaabaabaaba$ from Exercise 9.3

To compute all borders of x we search sequences of transition on the backbone leading from the initial state to some final state except the very last one. We find following sequences of transitions:

$0 \xrightarrow{\varepsilon} 0$

$0 \xrightarrow{a} 134679XZ$

$0 \xrightarrow{a} 134679XZ \xrightarrow{b} 258Y \xrightarrow{a} 369Z$

$0 \xrightarrow{a} 134679XZ \xrightarrow{b} 258Y \xrightarrow{a} 369Z \xrightarrow{a} 47X \xrightarrow{b} 58Y \xrightarrow{a} 69Z$

$0 \xrightarrow{a} 134679XZ \xrightarrow{b} 258Y \xrightarrow{a} 369Z \xrightarrow{a} 47X \xrightarrow{b} 58Y \xrightarrow{a} 69Z \xrightarrow{a} 7X \xrightarrow{b} 8Y \xrightarrow{a} 9Z$

By concatenating labels of transitions of each found sequence we obtain following strings ε, a, aba, $abaaba$ and $abaabaaba$. These strings are borders of x and the longest one $abaabaaba$ is the border of x. So, we can write $bord(x) = \{\varepsilon, a, aba, abaaba, abaabaaba\}$ and $Border(x) = abaabaaba$.

To compute border array of string x we do analysis of d-subsets of the states on the backbone. First, we create an empty array β of length $|x| = 12$ and initialise all its elements by 0. Next, we perform the own analysis of the states on the backbone having d-subset with more than one item, what are states $134679XZ$,, $258Y$, $369Z$, $47X$, $58Y$, $69Z$, $7X$, $8Y$ and $9Z$, in order from left to right and compute values of β. The process of computation is shown in the next table:

Analysed state	Values of border array elements
$13679XZ$	$\beta[3] = 1, \beta[6] = 1, \beta[7] = 1, \beta[9] = 1, \beta[X] = 1, \beta[Z] = 1$
$258Y$	$\beta[5] = 2, \beta[8] = 2, \beta[Y] = 2$
$369Z$	$\beta[6] = 3, \beta[9] = 3, \beta[Z] = 3$
$47X$	$\beta[7] = 4, \beta[X] = 4$
$58Y$	$\beta[8] = 5, \beta[Y] = 5$
$69Z$	$\beta[9] = 6, \beta[Z] = 6$
$7X$	$\beta[X] = 7$
$8Y$	$\beta[Y] = 8$
$9Z$	$\beta[Z] = 9$

The resulting border array $\beta(abaabaabaaba)$ is summed up in the next table:

i	1	2	3	4	5	6	7	8	9	X	Y	Z
symbol	a	b	a	a	b	a	a	b	a	a	b	a
$\beta[i]$	0	0	1	0	2	3	4	5	6	7	8	9

Exercise 9.4
Let us have string $x = abbabbabba$ over alphabet $A = \{a, b\}$.

1. Compute set of all borders $bord(x)$ of string x and the longest border $Border(x)$ of string x.
2. Compute border array $\beta[1..|x|]$ of string x.

First, we construct nondeterministic suffix automaton for string x. Next, we determinise the automaton and thus we obtain deterministic suffix automaton which has the transition diagram depicted in Fig. 9.5 (X represents number 10). During determinisation we memorise d-subsets. In the next step, we extract backbone of the deterministic suffix automaton in the way we process string x and remove all not used transitions and not visited states ($0 \xrightarrow{b} 235689$, $235689 \xrightarrow{b} 369$, $235689 \xrightarrow{a} 47X$ and the state 235689 drawn with dashed lines).

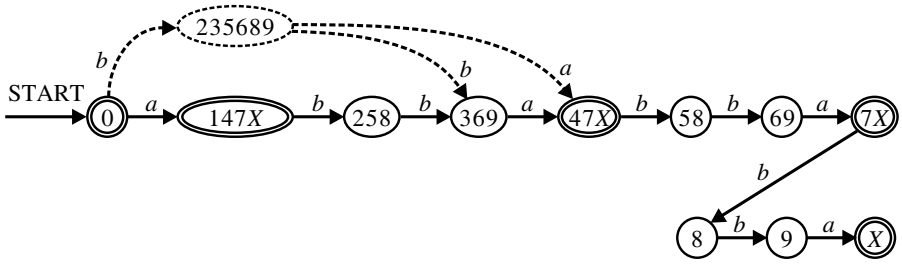

Figure 9.5: Transition diagram of the deterministic suffix automaton for $x = abbabbabba$ from Exercise 9.4

To compute all borders of x we search sequences of transition on the backbone leading from the initial state to some final state except the very last one. We find following sequences of transitions:

$0 \xrightarrow{\varepsilon} 0$
$0 \xrightarrow{a} 147X$
$0 \xrightarrow{a} 147X \xrightarrow{b} 258 \xrightarrow{b} 369 \xrightarrow{a} 47X$
$0 \xrightarrow{a} 147X \xrightarrow{b} 258 \xrightarrow{b} 369 \xrightarrow{a} 47X \xrightarrow{b} 58 \xrightarrow{b} 69 \xrightarrow{a} 7X$

By concatenating labels of transitions of each found sequence we obtain following strings ε, a, $abba$ and $abbabba$. These strings are borders of x and the longest one $abbabba$ is the border of x. So, we can write $bord(x) = \{\varepsilon, a, abba, abbabba\}$ and $Border(x) = abbabba$.

To compute border array of string x we do analysis of d-subsets of the states on the backbone. First, we create an empty array β of length $|x| = 10$ and initialise all its elements by 0. Next, we perform the analysis of the states on the backbone having d-subset with more than one item, what are states $147X$, 258, 369, $47X$, 58, 69 and $7X$ in order from left to right and compute values of β. The process of computation is shown in the next table:

Analysed state	Values of border array elements
$147X$	$\beta[4]=1, \beta[7]=1, \beta[X]=1$
258	$\beta[5]=2, \beta[8]=2$
369	$\beta[6]=3, \beta[9]=3$
$47X$	$\beta[7]=4, \beta[X]=4$
58	$\beta[8]=5$
69	$\beta[9]=6$
$7X$	$\beta[X]=7$

The resulting border array $\beta(abbabbabba)$ is summed up in the next table:

i	1	2	3	4	5	6	7	8	9	X
symbol	a	b	b	a	b	b	a	b	b	a
$\beta[i]$	0	0	0	1	2	3	4	5	6	7

Exercise 9.5
Let us have set of strings $S = \{aabba, abbaa\}$ over alphabet $A = \{a, b\}$.

1. Compute set of all borders *mbord*(S) of string S and the longest border *mBorder*(S) of set of strings S.

2. Compute border array $m\beta$ of set of strings S.

First, we construct nondeterministic suffix automaton for set of strings S. This automaton has the transition table shown in Table 9.2 a) and the transition diagram is depicted in Fig. 9.6.

	a	b
0	$124'55'$	$2'33'4$
1	2	$2'$
2		3
$2'$		$3'$
3		4
$3'$	$4'$	
4	5	
$4'$	$5'$	
5		
$5'$		

a)

	a	b
0	$124'55'$	$2'33'4$
$124'55'$	$25'$	$2'3$
$2'33'4$	$4'5$	$3'4$
$25'$		3
$2'3$		$3'4$
3		4
$3'4$	$4'5$	
4	5	
$4'5$	$5'$	
5		
$5'$		

b)

Table 9.2: Transition tables of the nondeterministic and the deterministic suffix automata for $S = \{aabba, abbaa\}$ from Exercise 9.5

Next, we determinise the automaton and thus we obtain deterministic suffix automaton which has the transition table shown in Table 9.2 b) and the transition diagram is depicted in Fig. 9.7. During determinisation we save d-subsets. In the next step, we extract backbone of the deterministic suffix automaton in the way we process strings os set S and remove all

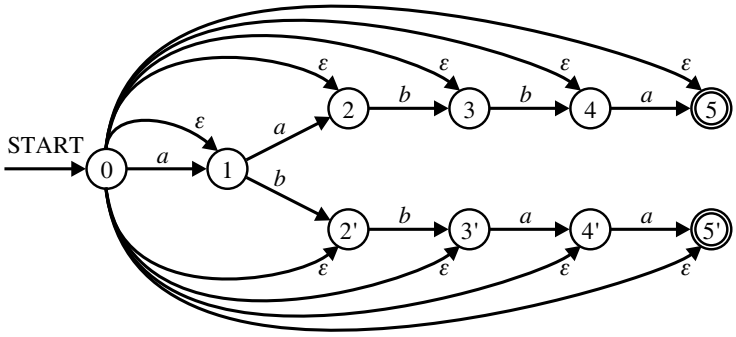

Figure 9.6: Transition diagram of the nondeterministic suffix automaton for set of strings $S = \{aabba, abbaa\}$ from Exercise 9.5

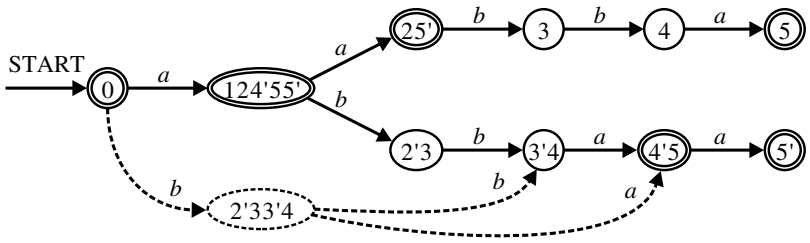

Figure 9.7: Transition diagram of the deterministic suffix automaton for $S = \{aabba, abbaa\}$ from Exercise 9.5

not used transitions and not visited states ($0 \xrightarrow{b} 2'33'4$, $2'33'4 \xrightarrow{b} 3'4$, $2'33'4 \xrightarrow{a} 4'5$ and the state $2'33'4$ - drawn with dashed lines).

To compute all borders of set S we search sequences of transition on the backbone leading from the initial state to some final state except the very last ones. We find following sequences of transitions:

$0 \xrightarrow{\varepsilon} 0$
$0 \xrightarrow{a} 124'55'$
$0 \xrightarrow{a} 124'55' \xrightarrow{a} 25'$
$0 \xrightarrow{a} 124'55' \xrightarrow{b} 2'3 \xrightarrow{b} 3'4 \xrightarrow{a} 4'5$

By concatenating labels of transitions of each found sequence we obtain following strings ε, a, aa, $abba$. These strings are borders of set S and the longest one $abba$ is the border of set S. So, we can write $mbord(S) = \{\varepsilon, a, aa, abba\}$ and $mBorder(S) = abba$.

To compute border array of set S we do analysis of d-subsets of the states on the backbone. First, we create an empty array $m\beta$ of length 9 what is number of different nonempty prefixes of S and hence the number of states (except the initial one) of the nondeterministic suffix automaton for S.

We initialise all elements of $m\beta$ by 0. Next, we perform the own analysis of the states on the backbone having d-subset with more than one item, what are states $124'55'$, $25'$, $2'3$, $3'4$ and $4'5$ in order from left to right with increasing depth of states. The process of computation is shown in the Table 9.3.

Analysed state	Values of mborder array elements
$124'55'$	$m\beta[2] = 1, m\beta[4'] = 1, m\beta[5] = 1, m\beta[5'] = 1$
$25'$	$m\beta[5'] = 2$
$2'3$	$m\beta[3] = 2$
$3'4$	$m\beta[4] = 3$
$4'5$	$m\beta[5] = 4$

Table 9.3: Computation of $m\beta(s)$ from Exercise 9.5

The resulting mborder array $m\beta(S)$ is shown in the next table:

state	1	2	$2'$	3	$3'$	4	$4'$	5	$5'$
symbol	a	a	b	b	b	b	a	a	a
$m\beta$[state]	0	1	0	2	0	3	1	4	2

Exercise 9.6
Let us have set of strings $S = \{aabaab, ababab\}$ over alphabet $A = \{a, b\}$.

1. Compute set of all borders *mbord*(S) of string S and the longest border *mBorder*(S) of set of strings S.
2. Compute border array $m\beta$ of set of strings S.

First, we construct nondeterministic suffix automaton for set of strings S. Next, we determinise the automaton and thus we obtain deterministic suffix automaton which has the transition diagram depicted in Fig. 9.8. During determinisation we memorise d-subsets. In the next step, we extract backbone of the deterministic suffix automaton in the way we process strings os set S and remove all not used transitions ($0 \xrightarrow{b} 2'34'66'$ - drawn with dashed lines).

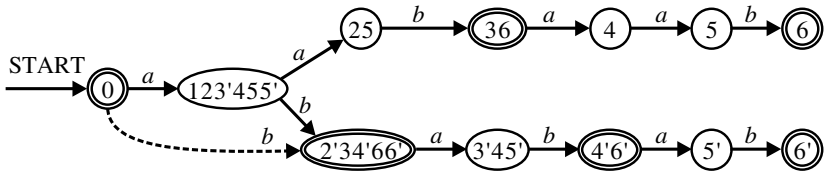

Figure 9.8: Transition diagram of the deterministic suffix automaton for set of strings $S = \{aabaab, ababab\}$ from Exercise 9.6

To compute all borders of set S we search sequences of transition on the backbone leading from the initial state to some final state except the very last ones. We find following sequences of transitions:

$0 \xrightarrow{\varepsilon} 0$
$0 \xrightarrow{a} 123'455' \xrightarrow{b} 2'34'66'$
$0 \xrightarrow{a} 123'455' \xrightarrow{a} 25 \xrightarrow{b} 36$
$0 \xrightarrow{a} 123'455' \xrightarrow{b} 2'34'66' \xrightarrow{a} 3'45' \xrightarrow{b} 4'6'$

By concatenating labels of transitions of each found sequence we obtain following strings ε, a, aa, $abba$. These strings are borders of set S and the longest one $abba$ is the border of set S. So, we can write $mbord(S) = \{\varepsilon, ab, aab, abab\}$ and $mBorder(S) = abab$.

To compute border array of set S we do analysis of d-subsets of the states on the backbone. First, we create an empty array $m\beta$ of length 11 what is number of different nonempty prefixes of S and hence the number of states (except the initial one) of the nondeterministic suffix automaton for S.

We initialise all elements of $m\beta$ by 0. Next, we perform the own analysis of the states on the backbone having d-subset with more than one item, what are states $123'455'$, $2'34'66'$, 25, $3'45'$, 36 and $4'6'$ in order from left to right with increasing depth of states. The process of computation is shown in the following table:

Analysed state	Values of mborder array elements
$123'455'$	$m\beta[3'] = 1, m\beta[2] = 1, m\beta[4] = 1, m\beta[5] = 1, m\beta[5'] = 1$
$2'34'66'$	$m\beta[4'] = 2, m\beta[3] = 2, m\beta[6] = 2, m\beta[6'] = 2$
25	$m\beta[5] = 2$
$3'45'$	$m\beta[5'] = 3,\ m\beta[4] = 3$
36	$m\beta[6] = 3$
$4'6'$	$m\beta[6'] = 4$

The resulting mborder array $m\beta(S)$ is shown in the following table:

state	1	2	$2'$	3	$3'$	4	$4'$	5	$5'$	6	$6'$
symbol	a	a	b	b	a	a	b	a	a	b	b
$m\beta$[state]	0	1	0	2	1	3	2	2	3	3	4

10 Repetitions in text

10.1 Exact repetitions

Exercise 10.1
Find repetitions in string $x = abcaabc$. Nondeterministic factor automaton for x has transition diagram depicted in Fig. 10.1.

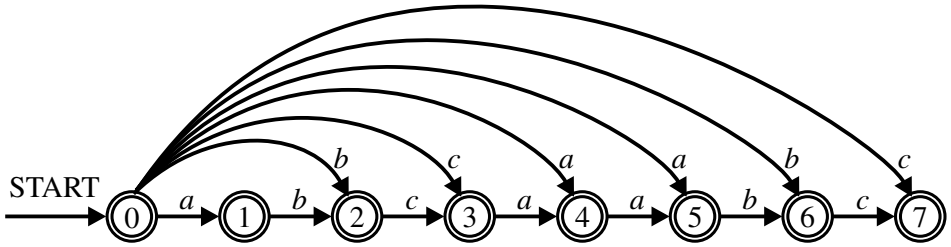

Figure 10.1: Transition diagram of a nondeterministic factor automaton for $x = abcaabc$

Transition tables of both nondeterministic and deterministic factor automata are shown in Table 10.1.

	a	b	c
0	$1, 4, 5$	$2, 6$	$3, 7$
1		2	
2			3
3	4		
4	5		
5		6	
6			7
7			

	a	b	c
0	145	26	37
145	5	26	
26			37
37	4		
4	5		
5		6	
6			7
7			

Table 10.1: Transition tables both of nondeterministic and deterministic factor automata from Exercise 10.1

Transition diagram of deterministic factor automaton is depicted in Fig. 10.2. Repetition

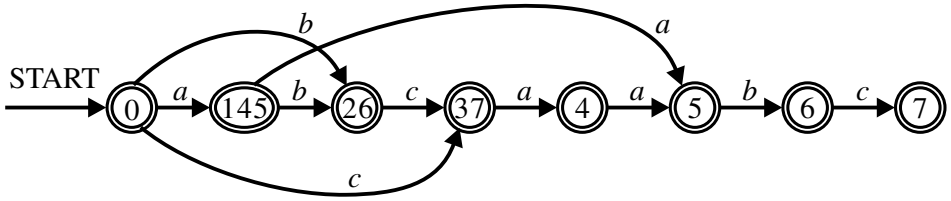

Figure 10.2: Transition diagram of deterministic factor automaton for $x = abcaabc$

table is shown in Table 10.2. It can be seen from this table that all repetitions are with gap and that maximum repeating factor is $r = abc$. All other repeating factors are factors of r.□

$d-$subset	Factor	Repetitions
145	a	$(1,F),(4,G),(5,G)$
26	ab	$(2,F),(6,G)$
37	abc	$(3,F),(7,G)$

Table 10.2: Repetition table for string $x = abcaabc$ from Exercise 10.1

Exercise 10.2
Find repetitions in string $x = abcabc$.
Nondeterministic factor automaton for x has transition diagram depicted in Fig. 10.3. Transitions tables of nondeterministic and deterministic factor automata are shown in Table 10.3. Transition diagram of deterministic factor automaton is depicted in Fig. 10.4. Repetition

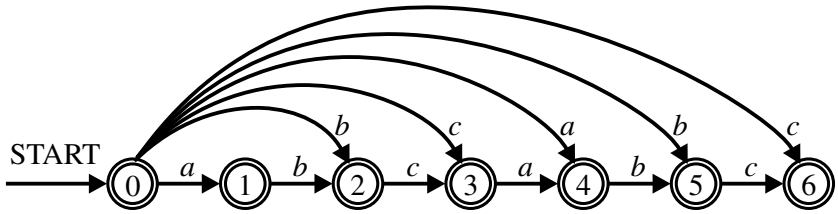

Figure 10.3: Transition diagram of nondeterministic factor automaton for $x = abcabc$ from Exercise 10.2

	a	b	c
0	$1,4$	$2,5$	$3,6$
1		2	
2			3
3	4		
4		5	
5			6
6			

	a	b	c
0	14	25	36
14		25	
25			36
36	4		
4		5	
5			6
6			

Table 10.3: Transition tables of both nondeterministic and deterministic factor automata for string $abcabc$ from Exercise 10.2

table is shown in Table 10.4. It can be seen from this table that almost all repetitions are with gap but repetitions of maximum repeating factor $r = abc$. Factor abc is repeating as a square and $x = (abc)^2$. □

Exercise 10.3
Find repetitions in string $x = ababa$. Nondeterministic factor automaton for x has transition diagram depicted in Fig. 10.5.
Transition tables of both nondeterministic and deterministic factor automata are shown in Table 10.5.

Transition diagram of deterministic factor automaton is depicted in Fig. 10.6.
Repetition table has this form:

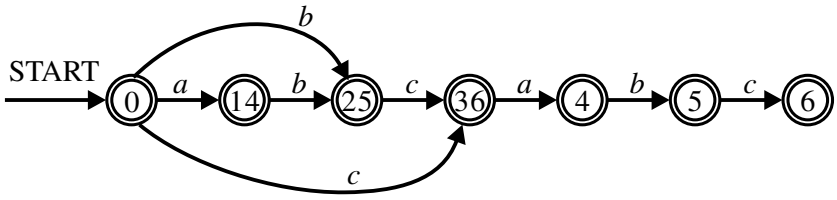

Figure 10.4: Transition diagram of deterministic factor automaton for $x = abcabc$ from Exercise 10.2

d-subset	Factor	Repetitions
14	a	$(1, F), (4, G)$
25	ab	$(2, F), (5, G)$
36	abc	$(3, F), (6, S)$

Table 10.4: Repetition table for string $x = abcabc$ from Exercise 10.2

d-subset	Factor	Repetitions
135	a	$(1, F), (3, G), (5, G)$
24	ab	$(2, F), (4, S)$
35	aba	$(3, F), (5, O)$

It can be seen from this table that string $x = ababa$ contains all kinds of repetitions:
a is repeating with gap,
$(ab)^2$ is a square,
aba is repeating with overlapping.
The maximum repeating factor is $r = aba$. □
We will show classes of strings with large number of repetitions in following exercises.

Exercise 10.4
Find repetitions in string $x = aaaaaa = a^6$. Nondeterministic factor automaton for x has transition diagram depicted in Fig. 10.7.
Transitions tables of both deterministic and nondeterministic factor automata are shown in Table 10.6. Transition diagram of deterministic factor automaton is depicted in Fig. 10.8. Repetition table is shown in Table 10.7. It can be seen from this table that in string $x = a^6$ are all kinds of repetitions. Even there is a cube $(aa)^3$. The maximum repeating factor is $r = a^5$.

□

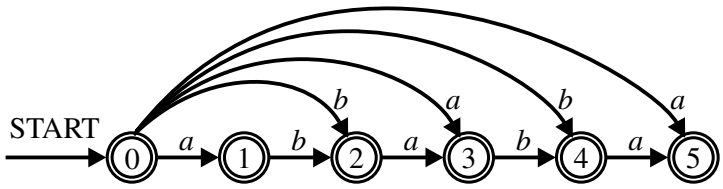

Figure 10.5: Transition diagram of nondeterministic factor automaton for string $x = ababa$ from Exercise 10.3

	a	b
0	$1, 3, 5$	$2, 4$
1		2
2	3	
3		4
4	5	
5		

	a	b
0	135	24
135		24
24	35	
35		4
4	5	

Table 10.5: Transition tables of both nondeterministic and deterministic factor automata for string *abcabc* from Exercise 10.3

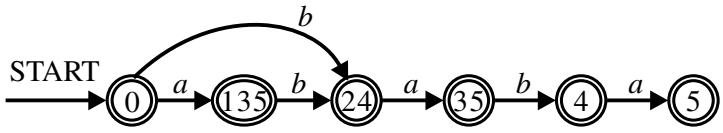

Figure 10.6: Transition diagram of deterministic factor automaton for $x = ababa$ from Exercise 10.3

Exercise 10.5
Find repetitions in string $x = ababab = (ab)^3$. Nondeterministic factor automaton for x has transition diagram depicted in Fig. 10.9. Transitions tables of both deterministic and nondeterministic factor automata are shown in Table 10.8. Transition diagram of deterministic factor automaton is depicted in Fig. 10.10. Repetition table is shown in Table 10.9. It can be seen from this table that in string $x = (ab)^3$ are all kinds of repetitions. String x itself is a cube. The maximum repeating factor is $r = abab$. □

Exercise 10.6
Find repetitions in string $x = abaababa$. This string is Fibonacci string f_5. Nondeterministic factor automaton for x has transition diagram depicted in Fig. 10.11. Transitions tables of both deterministic and nondeterministic factor automata are shown in Table 10.10. Transition diagram of deterministic factor automaton is depicted in Fig. 10.12. Repetition table is shown in Table 10.11. It can be seen from this table that in string f_5 are all repetitions with gaps and squares. The maximum repeating factor is $r = aba$. □

Exercise 10.7
Find repetitions in set of strings $X = \{abaababa, ababab\} = \{f_5, (ab)^3\}$ (see Exercises 10.5 and 10.6). Moreover, find the longest common factor of both strings. Nondeterministic

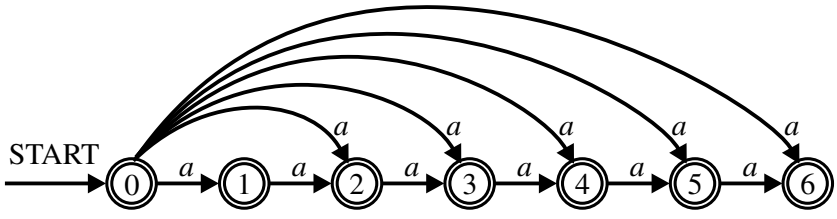

Figure 10.7: Transition diagram of nondeterministic factor automaton for string $x = a^6$ from Exercise 10.4

	a
0	$1, 2, 3, 4, 5, 6$
1	2
2	3
3	4
4	5
5	6
6	

	a
0	123456
123456	23456
23456	3456
3456	456
456	56
56	6
6	

Table 10.6: Transition tables of both nondeterministic and deterministic factor automata for string *abcabc* from Exercise 10.4

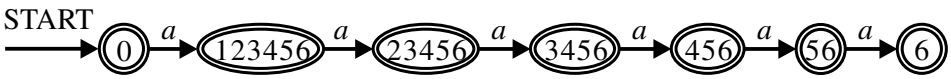

Figure 10.8: Transition diagram of factor automaton for $x = a^6$ from Exercise 10.4

factor automaton for X has transition diagram depicted in Fig. 10.13. Transition table of nondeterministic factor automaton is shown in Table 10.12. A part of transition table of deterministic factor automaton (it contains only rows for multiple states) is shown in Table 10.13. Corresponding part of transition diagram is depicted in Fig. 10.14. It can be seen from automaton depicted in Fig. 10.14 that the longest common factor of both strings is $LCF(f_5, (ab)^3) = ababa$. This fact is indicated by state 85'. All factors of string $LCF(f_5, (ab)^3)$ are also common factors of both strings. The repetition table is shown in Table 10.14. □

10.2 Approximate repetitions

Exercise 10.8
Find exact and approximate repetitions in string $x = abcabd$, with Hamming distance $k = 1$. Transition diagram of nondeterministic approximate factor automaton is depicted in Fig. 10.15. Transition table of this automaton is shown in Table 10.15.

We construct during the determinisation of the nondeterministic approximate factor automaton a multiple front end of deterministic factor automaton containing states correspond-

d-subset	Factor	Repetitions
123456	a	$(1, F), (2, S), (3, G), (4, G), (5, G), (6, G)$
23456	aa	$(2, F), (3, O), (4, S), (5, G), (6, G)$
3456	aaa	$(3, F), (4, O), (5, O), (6, S)$
456	$aaaa$	$(4, F), (5, O), (6, O)$
56	$aaaaa$	$(5, F), (6, O)$

Table 10.7: Repetition table for string $x = a^6$ from Exercise 10.4

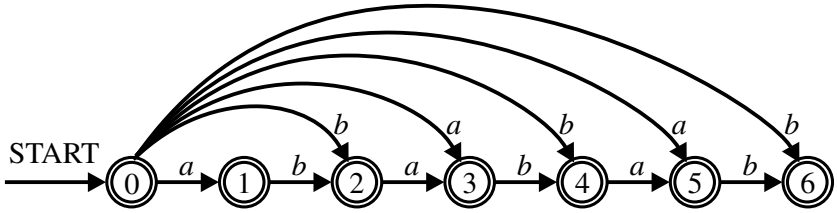

Figure 10.9: Transition diagram of nondeterministic factor automaton for string $x = (ab)^3$ from Exercise 10.5

	a	b
0	1, 3, 5	2, 4, 6
1		
2		
3		
4		
5		
6		

	a	b
0	135	246
135		246
246	35	
35		46
46	5	
5		6
6		

Table 10.8: Transition tables of both nondeterministic and deterministic factor automata for string *abcabc* from Exercise 10.5

ing to d-subsets having these properties:

- contain more states from level 0 (exact repetitions),
- contain both states from level 0 and level 1 (approximate repetitions).

Transition table of multiple front end of deterministic factor automaton is in Table 10.16. Transition diagram of the multiple front end of approximate deterministic factor automaton is depicted in Fig. 10.16. Repetition table is shown in Table 10.17.

Exercise 10.9
Find exact and approximate repetitions in string $x = abcabc$ over alphabet $A = \{a, b, c, d\}$ with

1. Levenshtein distance $k = 1$,
2. Δ distance $k = 1$,
3. Γ distance $k = 1$,
4. (Δ, Γ) distance $k = 2$, $l = 1$.

The solution of this Exercise is left to the reader. □

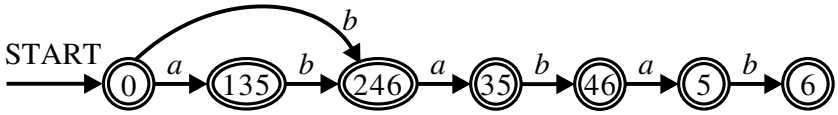

Figure 10.10: Transition diagram of factor automaton for $x = (ab)^3$ from Exercise 10.5

d-subset	Factor	Repetitions
135	a	$(1, F), (3, G), (5, G)$
246	ab	$(2, F), (4, S), (6, G)$
35	aba	$(3, F), (5, O)$
46	$abab$	$(4, F), (6, O)$

Table 10.9: Repetition table for string $x = (ab)^3$ from Exercise 10.5

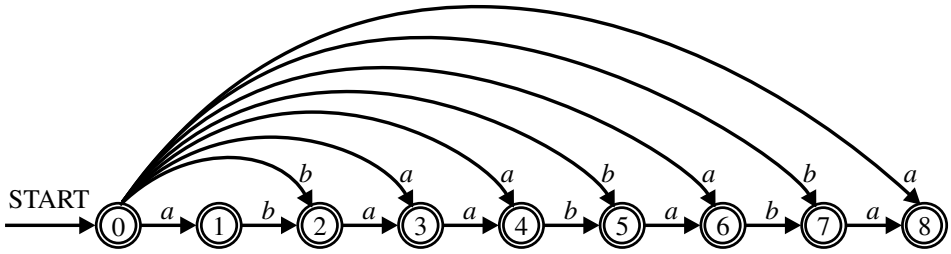

Figure 10.11: Transition diagram of nondeterministic factor automaton for string $x = f_5$ from Exercise 10.6

	a	b
0	$1, 3, 4, 6, 8$	$2, 5, 7$
1		2
2	3	
3	4	
4		5
5	6	
6		7
7	8	
8		

	a	b
0	13468	257
13468	4	257
257	368	
368	4	7
4		5
5	6	
6		7
7	8	
8		

Table 10.10: Transition tables of both nondeterministic and deterministic factor automata for string $x = abaababa$ from Exercise 10.6

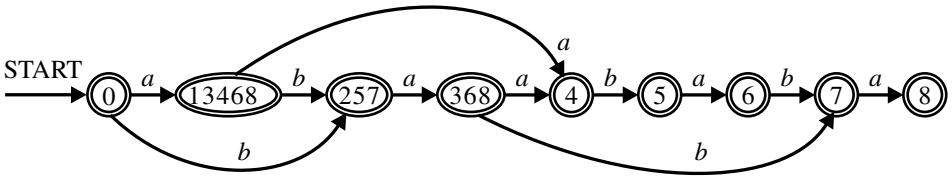

Figure 10.12: Transition diagram of factor automaton for $x = f_5$ from Exercise 10.6

d-subset	Factor	Repetitions
13468	a	$(1, F), (3, G), (4, G), (6, G), (8, G)$
257	ab	$(2, F), (5, G), (7, G)$
368	aba	$(3, F), (6, S), (8, G)$

Table 10.11: Repetition table for string $x = f_5$ from Exercise 10.6

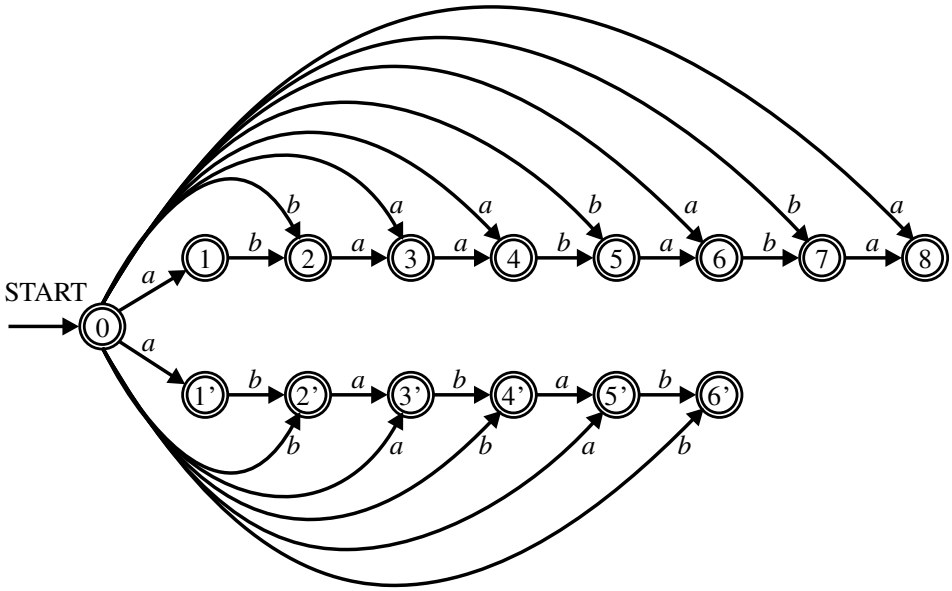

Figure 10.13: Transition diagram of nondeterministic factor automaton for strings from set $X = \{f_5, (ab)^3\}$ from Exercise 10.7

	a	b
0	$1, 3, 4, 6, 8, 1', 3', 5'$	$2, 5, 7, 2', 4', 6'$
1		2
2	3	
3	4	
4		5
5	6	
6		7
7	8	
8		
$1'$		$2'$
$2'$	$3'$	
$3'$		$4'$
$4'$	$5'$	
$5'$		$6'$
$6'$		

Table 10.12: Transition table of the nondeterministic factor automaton from Exercise 10.7

	a	b
0	$134681'3'5'$	$2572'4'6'$
$134681'3'5'$	4	$2572'4'6'$
$2572'4'6'$	$3683'5'$	
$3683'5'$	4	$74'6'$
$74'6'$	$85'$	
$85'$		$6'$

Table 10.13: Part of transition table of the deterministic factor automaton from Exercise 10.7

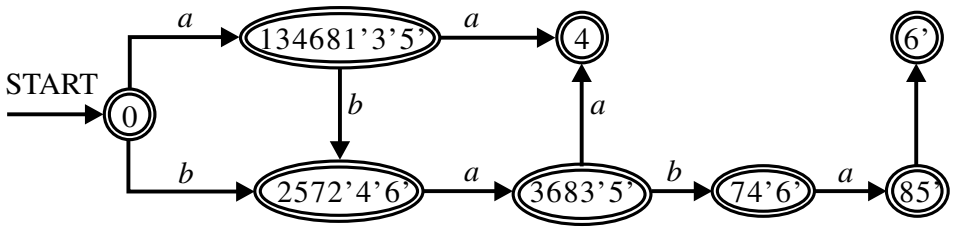

Figure 10.14: Part of transitions diagram of deterministic factor automaton for set $X = \{f_5, (ab)^3\}$ from Exercise 10.7

$d-$subset	Factor	Repetitions
$134681'3'5'$	a	$(1,F),(1',F),(3,G),(4,S),(6,G),(8,G),(3',G),(5',G)$
$2572'4'6'$	b	$(2,F),(2',F),(5,G),(7,G),(4',G),(6',G)$
$2572'4'6'$	ab	$(2,F),(2',F),(5,G),(7,S),(4',S),(6',S)$
$3683'5'$	ba	$(3,F),(3',F),(6,G),(8,S),(5',S)$
$3683'5'$	aba	$(3,F),(3',F),(6,S),(8,O),(5',O)$
$74'6'$	bab	$(7,F),(4',F),(6,O)$
$74'6'$	$abab$	$(7,F),(4',F),(6,O)$
$85'$	$baba$	$(8,F),(5',F)$
$85'$	$ababa$	$(8,F),(5',F)$

Table 10.14: Repetition table for set $X = \{f_5, (ab)^3\}$ from Exercise 10.7

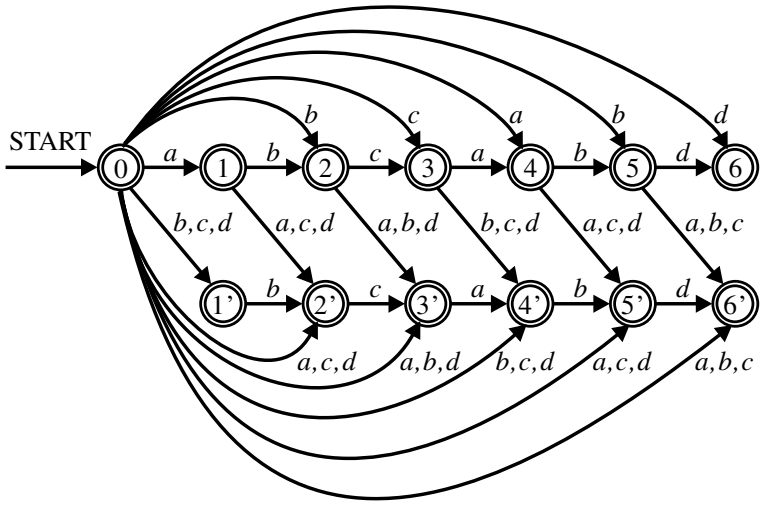

Figure 10.15: Transition diagram of the nondeterministic approximate factor automaton for $x = abcabd$, Hamming distance $k = 1$, from Exercise 10.8

	a	b	c	d
0	$1, 4, 2', 3', 5', 6'$	$2, 5, 1', 3', 4', 6'$	$3, 1', 2', 4', 5', 6'$	$6, 1', 2', 3', 4', 5'$
1	$2'$	2	$2'$	$2'$
2	$3'$	$3'$	3	$3'$
3	4	$4'$	$4'$	$4'$
4	$5'$	5	$5'$	$5'$
5	$6'$	$6'$	$6'$	6
6				
$1'$		$2'$		
$2'$			$3'$	
$3'$	$4'$			
$4'$		$5'$		
$5'$				$6'$
$6'$				

Table 10.15: Transition table of the nondeterministic approximate factor automaton from Exercise 10.8

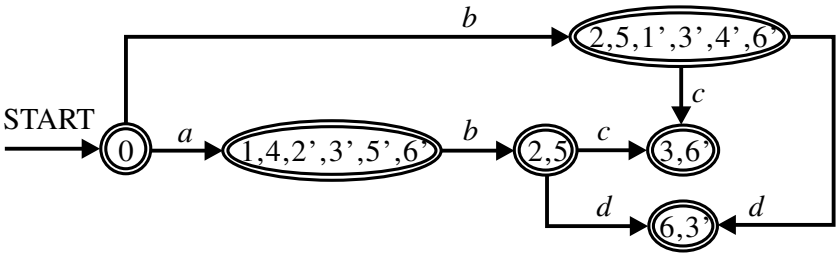

Figure 10.16: Interesting part of the deterministic approximate factor automaton for string $x = abcabd$, Hamming distance $k = 1$ from Exercise 10.8

	a	b	c	d
0	$1, 4, 2', 3', 5', 6'$	$2, 5, 1', 3', 4', 6'$	$3, 1', 2', 4', 5', 6'$	$6, 1', 2', 3', 4', 5'$
$1, 4, 2', 3', 5', 6'$	$2', 4', 5'$	$2, 5$	$2', 3', 5'$	$2', 5', 6'$
$2, 5, 1', 3', 4', 6'$	$3', 4', 6'$	$2', 3', 5', 6'$	$3, 6'$	$6, 3'$
$3, 1', 2', 4', 5', 6'$	4	$2', 4', 5'$	$3', 4'$	$4', 6'$
$6, 1', 2', 3', 4', 5'$	$4'$	$2', 5'$	$3'$	$6'$
$2, 5$	$3', 6'$	$3', 6'$	$3, 6'$	$6, 3'$
$3, 6'$	4	$4'$	$4'$	$4'$
$6, 3'$	$4'$			

Table 10.16: Transition table of multiple front end of the deterministic approximate factor automaton from Exercise 10.8

d-subset	Factor	Repetitions
2,5	ab	$(2, F), (5, G, 0)$
3,6'	abc	$(3, F), (6, S, 1)$
6,3'	abd	$(6, F), (3', S, 1)$

Table 10.17: Repetition table for string $x = abcabd$ from Exercise 10.8

11 Simulation of searching automata, *MP* and *KMP* algorithms

Some examples of Knuth-Morris-Pratt (*KMP*) searching automata which are simulators of nondeterministic finite automata for exact pattern matching of one pattern (*SFOECO* automata) are shown in this chapter. Moreover this approach is shown for an approximate pattern matching of one pattern.

11.1 *KMP* searching automata

Exercise 11.1
Construct *KMP* searching automaton for pattern $P = abab$. First we compute border array for pattern $P = abab$. The transition diagram of the nondeterministic factor is depicted in Fig. 11.1. Table 11.1 is the transition table of the deterministic factor automaton. Transition diagram of the deterministic factor automaton is depicted in Fig. 11.2.

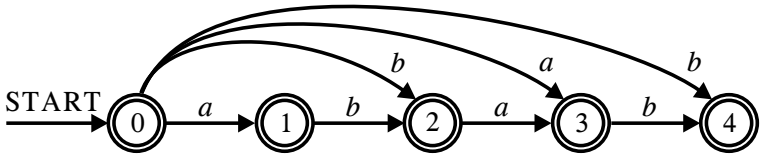

Figure 11.1: Transition diagram of the nondeterministic factor automaton for pattern $P = abab$ from Exercise 11.1

	a	b
0	13	24
13		24
24	3	
3		4
4		

Table 11.1: Transition table of the deterministic factor automaton from Exercise 11.1

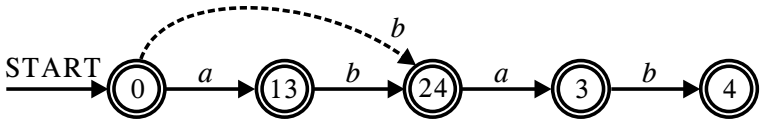

Figure 11.2: Transition diagram of the deterministic factor automaton for pattern $P = abab$ from Exercise 11.1, dashed transition is out of the backbone

The analysis of d-subsets on the backbone of the deterministic factor automaton is shown in this table:

Analysed state	Value of border array element
13	$\varphi[3] = 1$
24	$\varphi[4] = 2$

Let us recall that the elements of border array are equal to the fail function. Values of the fail function φ (equal to the border array elements) are shown in this table:

j	1	2	3	4
symbol	a	b	a	b
$\varphi[j]$	0	0	1	2

Transition diagram of *MP* searching automaton with fail function φ is shown in Fig. 11.3. Now the optimisation of fail function φ takes place. It holds that $\delta(a) = \delta(aba)$ and therefore $\varphi_{opt}(aba) = \varphi(a) = 0$. Transition diagram of *KMP* automaton with optimised fail function φ_{opt} is shown in Fig. 11.4. Compare this automaton with automaton from Exercise 4.1. □

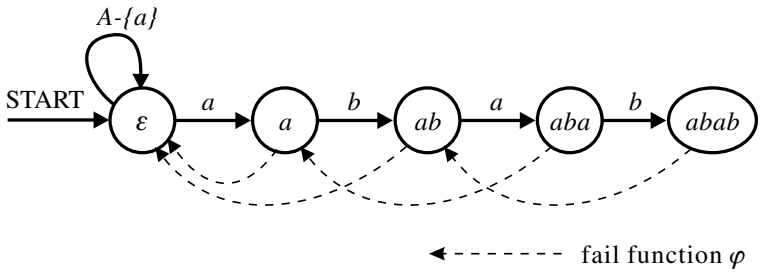

Figure 11.3: Transition diagram of the *MP* searching automaton for pattern $P = abab$ and fail function φ from Exercise 11.1

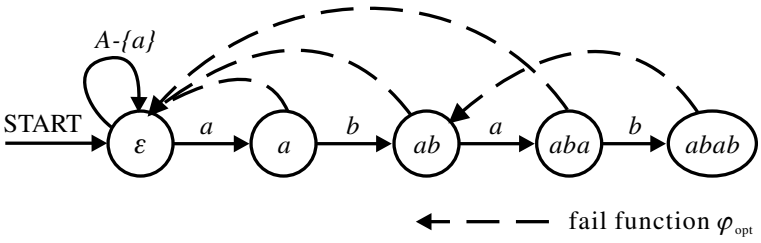

Figure 11.4: Transition diagram of the *KMP* searching automaton for pattern $P = abab$ and fail function φ_{opt} from Exercise 11.1

Exercise 11.2
Construct *KMP* searching automaton for pattern $P = ppppp$ (five p). First we compute the border array (fail function φ) for pattern $P = ppppp$. Table 11.2 is the transition table of the deterministic factor automaton for the pattern P. The transition diagram of the deterministic factor automaton is depicted in Fig. 11.5.

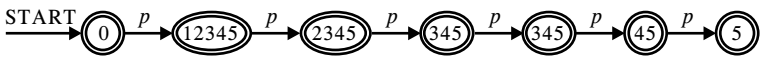

Figure 11.5: Transition diagram of the deterministic factor automaton for pattern $P = abab$ from Exercise 11.2

The analysis of d-subsets of the deterministic factor automaton (the automaton is composed of the backbone only) is shown in Table 11.3. Values of the fail function φ are shown

	p
0	12345
12345	2345
2345	345
345	45
45	5
5	

Table 11.2: Transition table of the deterministic factor automaton from Exercise 11.2

Analysed state	Values of border array elements
12345	$\varphi[2] = \varphi[3] = \varphi[4] = \varphi[5] = 1$
2345	$\varphi[3] = \varphi[4] = \varphi[5] = 2$
345	$\varphi[4] = \varphi[5] = 3$
45	$\varphi[5] = 4$

Table 11.3: Table of border array for pattern $P = ppppp$ from Exercise 11.2

in Table 11.4.

Transition diagram of *MP* searching automaton is depicted in Fig. 11.6. In this figure is shown fail function φ and fail function φ_{opt} of *KMP* automaton. The optimised fail function φ_{opt} is computed using the following fact $\delta(p) = \delta(pp) = \delta(ppp) = \delta(pppp)$ and therefore $\varphi_{opt}(pp) = \varphi_{opt}(ppp) = \varphi_{opt}(pppp) = \varphi(p) = 0$. □

Exercise 11.3

Construct *KMP* searching automaton for pattern $P = blablabla$. First we compute the border array (fail function φ) for pattern $P = blablabla$. Table 11.5 is the transition table of the deterministic factor automaton for the pattern P. Transition diagram of the deterministic factor automaton is depicted in Fig. 11.7. The analysis of d-subsets on the backbone of the deterministic factor automaton is shown in the Table 11.6. Values of the fail function φ (equal to the border array elements) are shown in this table:

j	1	2	3	4	5	6	7	8	9
symbol	b	l	a	b	l	a	b	l	a
$\varphi[j]$	0	0	0	1	2	3	4	5	6

Transition diagrams of *MP* and *KMP* automata are depicted in Fig. 11.8. □

j	1	2	3	4	5
symbol	p	p	p	p	p
$\varphi[j]$	0	1	2	3	4

Table 11.4: Table of fail function from Exercise 11.2

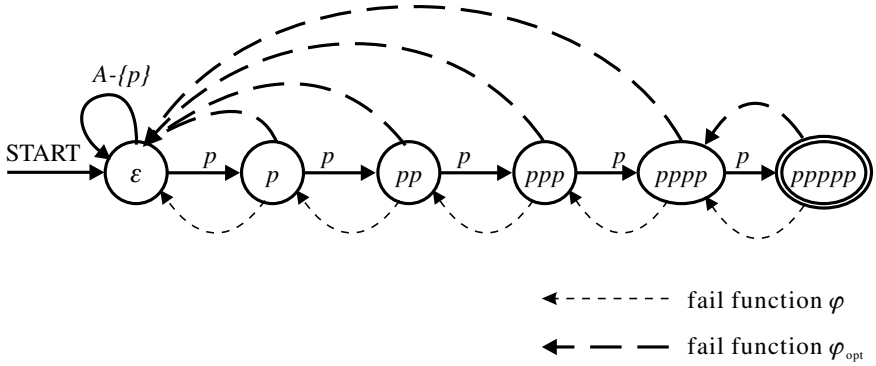

Figure 11.6: Transition diagrams of the *MP* and *KMP* searching automata for pattern $P =$ *ppppp* from Exercise 11.2

	a	b	l
0	369	147	258
147			258
258	369		
369		47	
47			58
58	69		
69		7	
7			8
8	9		

Table 11.5: Transition table of the deterministic factor automaton from Exercise 11.3

11.2 Approximate searching automaton and fail function

Exercise 11.4
Pattern $P =$ *abab* over alphabet $A = \{a, b\}$ is given. Construct *KMP* approximate searching automaton for pattern P with Hamming distance $k = 1$. Nondeterministic searching automaton (*SFORCO* automaton, see [TSA, Chapter 2]) has transition diagram depicted in Fig. 11.9.

First the approximate fail function will be computed. To do this the following approach will be used. Construct approximate prefix automaton for it with Hamming distance equal to 1. Its transition table is shown in Table 11.7 and its transition diagram is depicted in Fig. 11.10.

First we compute the approximate border array (fail function φ) for the pattern P. We

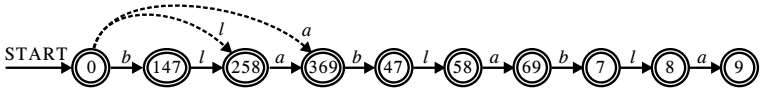

Figure 11.7: Transition diagram of the deterministic factor automaton for pattern $P =$ *blablabla* from Exercise 11.2, dashed transitions are out of the backbone

Analysed state	Values of border array elements
147	$\varphi[4] = \varphi[7] = 1$
258	$\varphi[5] = \varphi[8] = 2$
369	$\varphi[9] = \varphi[6] = 3$
47	$\varphi[7] = 4$
58	$\varphi[8] = 5$
69	$\varphi[9] = 6$

Table 11.6: Analysis of d-subsets of deterministic factor automaton from Exercise 11.3

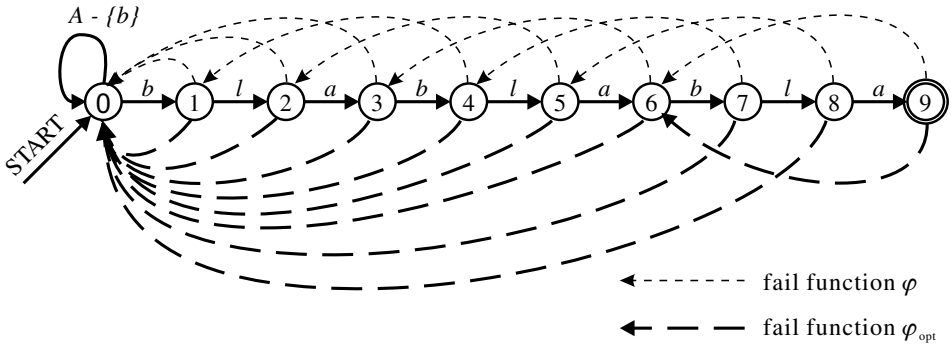

Figure 11.8: Transition diagrams of the *MP* and *KMP* searching automata for pattern $P =$ *blablabla* from Exercise 11.3

construct approximate factor automaton for pattern $P = abab$ and Hamming distance equal to 1. Transition diagram of this automaton with ε-transitions is depicted in Fig. 11.11. Transition diagram of the approximate factor automaton after elimination ε–transitions is depicted in Fig. 11.12. Transition table of this automaton is shown in Table 11.2a).

Deterministic approximate factor automaton has transition diagram depicted in Fig. 11.13. Transitions and states drawn by dashed lines are out of the backbone. Its transition table is shown in Table 11.2b).

The next operation should be computation of "approximate" border array. But this way is not straightforward, because there are some states $(2'3'4', 3', 4')$ to which more than one sequence of transitions from the initial state are leading for different strings. This situation is shown in Table 11.2.

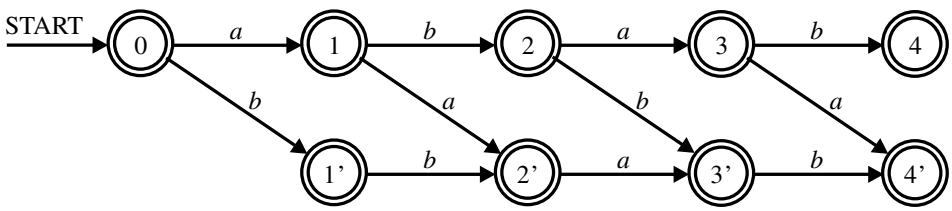

Figure 11.9: Transition diagram of the nondeterministic searching automaton (*SFORCO* automaton) from Exercise 11.4

	a	b
0	1	$1'$
1	$2'$	2
2	3	$3'$
3	$4'$	4
4		
$1'$		$2'$
$2'$	$3'$	
$3'$		$4'$
$4'$		

Table 11.7: Transition table of approximate prefix automaton from Exercise 11.4

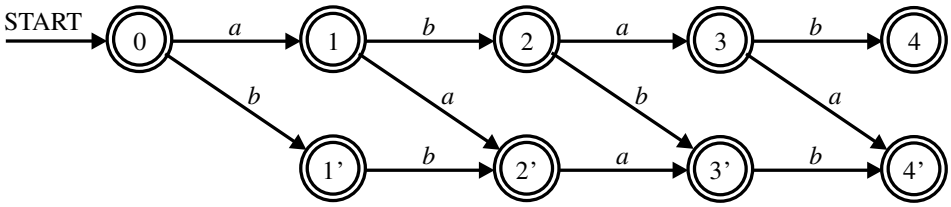

Figure 11.10: Transition diagram of the prefix automaton from Exercise 11.4

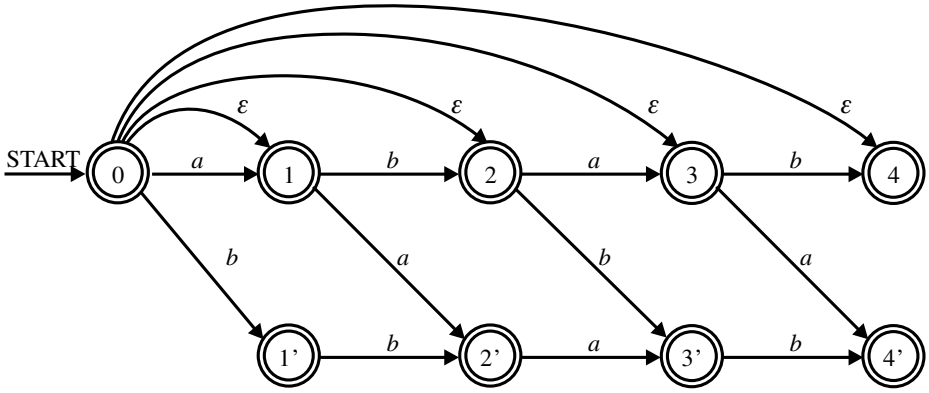

Figure 11.11: Transition diagram of the approximate factor automaton with ε-transitions from Exercise 11.4

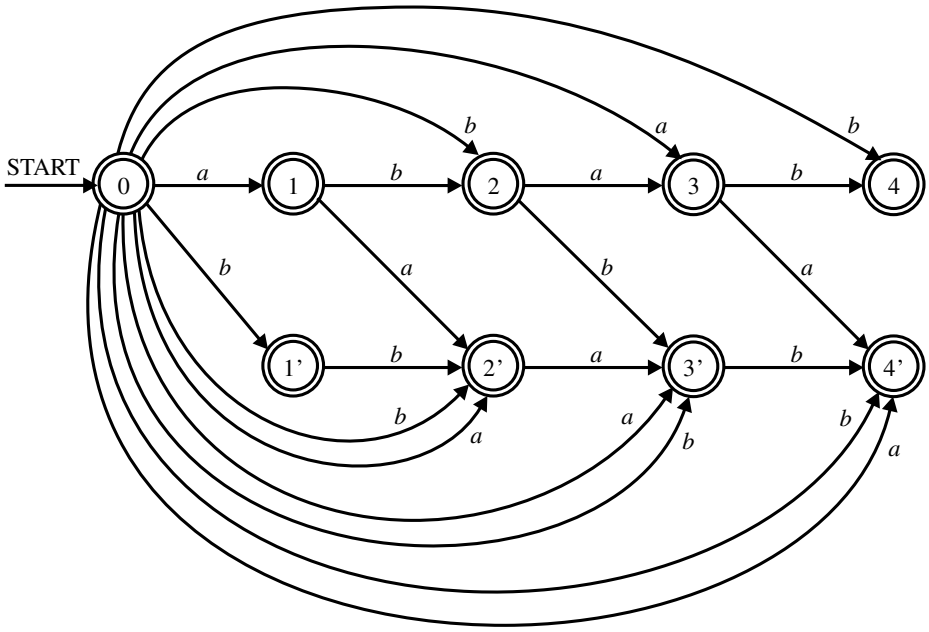

Figure 11.12: Transition diagram of the approximate nondeterministic factor automaton from Exercise 11.4 after elimination of ε–transitions

	a	b
0	$132'3'4'$	$241'2'3'4'$
1	$2'$	2
2	3	$3'$
3	$4'$	4
4		
$1'$		$2'$
$2'$	$3'$	
$3'$		$4'$
$4'$		

a)

	a	b
0	$132'3'4'$	$241'2'3'4'$
$132'3'4'$	$2'3'4'$	$244'$
$241'2'3'4'$	$33'$	$2'3'4'$
$244'$	3	$3'$
$2'3'4'$	$3'$	$4'$
$33'$	$4'$	$44'$
4		
$3'$		$4'$
$4'$		

b)

Table 11.8: Transition tables of both nondeterministic and deterministic approximate factor automata from Exercise 11.4

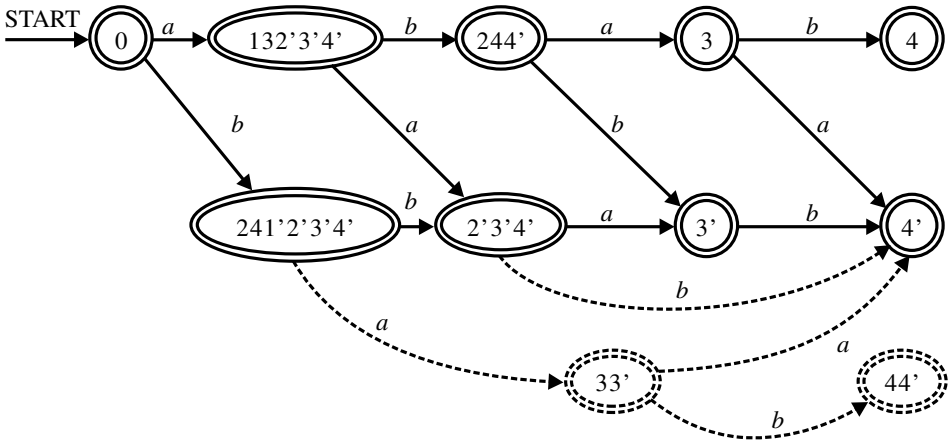

Figure 11.13: Transition diagram of the deterministic approximate factor automaton for string *abab* with Hamming distance equal to 1 from Exercise 11.4

State	Sequences of labels of transitions from the initial state
$2'3'4'$	aa bb
$3'$	abb aaa baa
$4'$	$abaa$ $abbb$ $aaab$ $bbab$

Table 11.9: Sequences of transitions for some states of approximate factor automaton from Exercise 11.4

This conflict situation can be solved by the following approaches:

1. To reconstruct the backbone of the approximate factor automaton in order to obtain an automaton having only one sequence of transitions to each state from the initial state.
2. To make elements of the border array (and in the same time the fail function) dependent on the sequence of transitions from the initial state.

The reconstruction of the backbone of the approximate factor automaton can be done by splitting of states. Each state in question is splitted into the number of states equal to the number of sequences leading from the initial state to it (see Table 11.2). Using this approach, we obtain the backbone of the factor automaton having transition diagram depicted in Figure 11.14. We can see that this automaton has a tree-like form. It means that the set of all strings having maximum Hamming distance equal to one from pattern $P = abbb$ is treated as a finite set $H_1 = \{abab, bbab, aaab, abbb, abaa\}$. This task can be solved by Aho-Corasick algorithm described in [TSA, Chapter 5].

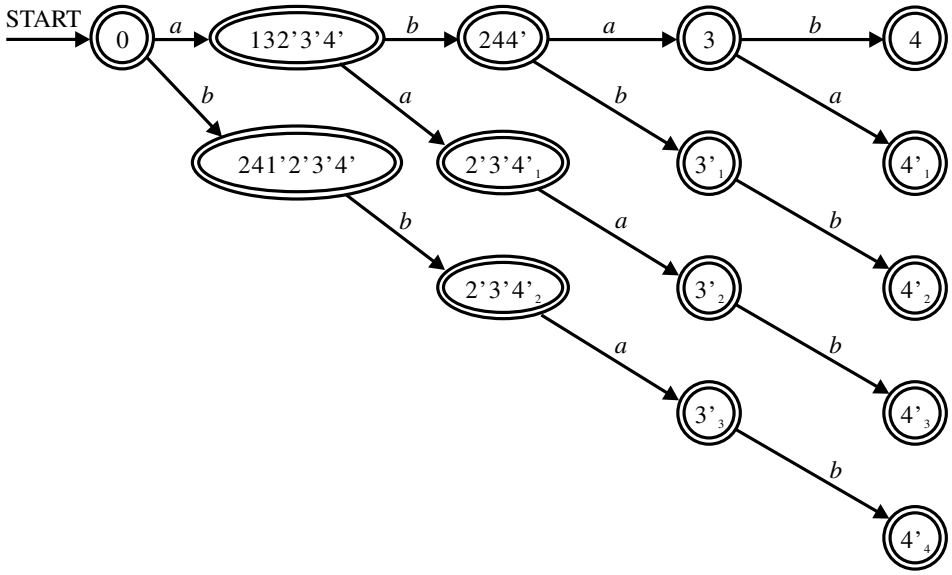

Figure 11.14: Transition diagram of the backbone of approximate factor automaton after splitting states 2'3'4', 3' and $4'$ from Exercise 11.4

The second approach, making the fail function dependent on the sequence of transitions from the initial state, can be explained using the backbone of the approximate factor automaton having transition diagram depicted in Fig. 11.13.

It can be mentioned that it is necessary to define fail function only for states $4, 1', 2', 3'$ and $4'$, because in all other states with exception of the state 1' the transitions for all symbols of the alphabet exist. Therefore in the next overview of the computation of fail function we can take into account only d-subsets containing above mentioned states. The overview is shown in Table 11.10.

Now the fail function φ can be computed. It depends on the state and moreover on the labels on paths leading to this state from the initial state. Values of the fail function depending on states and the labels of a respective path is shown in Table 11.11. It can be seen that some suffixes of strings labelling the paths from the initial state are enough to

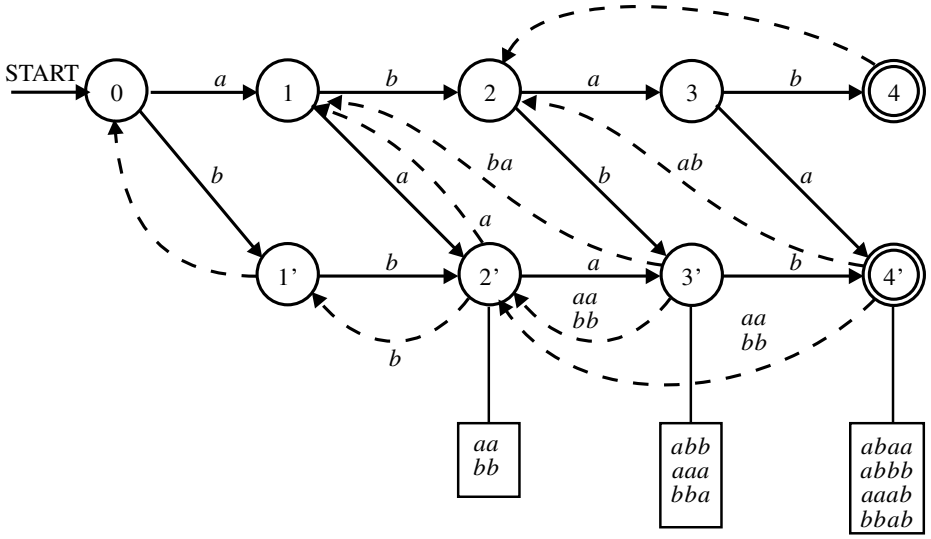

Figure 11.15: Transition diagram of the *MP* search Hamming automaton with added fail function φ from Exercise 11.4

d-subset	String	Values of border array elements
$132'3'4'$	a	$\varphi[2'] = \varphi[3'] = \varphi[4'] = 1$
$244'$	ab	$\varphi[4] = \varphi[4'] = 2$
$241'2'3'4'$	b	$\varphi[4] = \varphi[2'] = \varphi[3'] = \varphi[4'] = 1'$
$2'3'4'$	aa	$\varphi[3'] = \varphi[4'] = 2'$
	bb	$\varphi[3'] = \varphi[4'] = 2'$

Table 11.10: Computation of approximate border array for pattern $P = abab$ and Hamming distance equal to one from Exercise 11.4

distinguish values of the fail function. This suffixes are on the right of the symbol '|' used in the heading of the table.

In Fig. 11.15 is depicted Hamming automaton with added the fail function φ. Some transitions corresponding to the fail functions are labelled by suffixes of strings labelling the paths from the initial state used for the distinguishing values of the fail function. Moreover, there is list of strings for which the paths are leading from the initial state to states $2', 3'$ and $4'$. Let us mention that the fail function φ is not possible to optimise and therefore $\varphi_{opt} = \varphi$.

	ε	$a\|a$	$b\|b$	$a\|bb$	$a\|aa$	$b\|ba$	$ab\|aa$	$ab\|bb$	$aa\|ab$	$bb\|ab$
4	2									
$1'$	0									
$2'$		1	$1'$							
$3'$				$2'$	$2'$	1				
$4'$							$2'$	$2'$	2	2

Table 11.11: Values of fail function φ depending on the state and the path leading from the initial state to the state in question from Exercise 11.4

12 Simulation of searching automata, *AC* algorithm

Some examples of Aho-Corasick (AC) searching automata which are simulations of nondeterministic finite automata for exact matching of a finite set of patterns (*SFFECO* automata) are shown in this Chapter.

12.1 *AC* searching automata

Exercise 12.1
Construct AC searching automaton for searching set of patterns P={*abba,baba*} over alphabet $A = \{a, b\}$. Nondeterministic searching automaton for the finite set of pattern P (*SFFECO* automaton, see [TSA, Chapter 2]) has transition diagram depicted in Fig. 12.1. First the

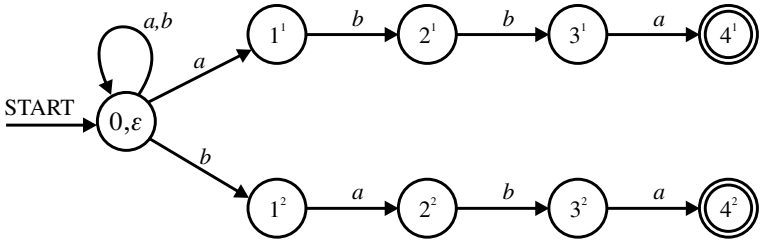

Figure 12.1: Transition diagram of the nondeterministic searching automaton (*SFFECO* automaton) for the set of patterns P={*abba,baba*} from Exercise 12.1

m-border array $m\beta$ will be computed. To do this the following approach will be used.

The starting point is the construction of prefix automaton for the set of patterns P={*abba, baba*}. Its transition diagram is depicted in Fig. 12.2.

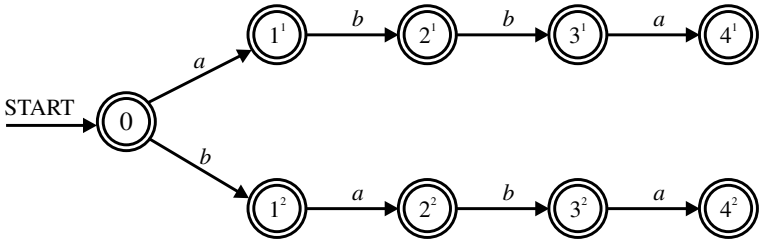

Figure 12.2: Transition diagram of the prefix automaton for the set of patterns P={*abba,baba*} from Exercise 12.1

The next step is the addition of ε-transitions from the initial state to all other states of the prefix automaton. The resulting automaton is a factor automaton with ε-transitions having transition diagram depicted in Fig. 12.3.

The nondeterministic factor automaton is obtained by the removal of ε-transitions. The resulting automaton has transition diagram depicted in Fig. 12.4.

The last step is determinisation of the factor automaton. Transition tables of both nondeterministic and deterministic factor automata are shown in Table 12.1. Transition diagram of the deterministic factor automaton having transition table shown in Table 12.1 is depicted in Fig. 12.5. Note that dashed lines are out the backbone of the factor automaton. An analysis of the multiple d-subsets is shown in Table 12.2. The resulting m-border array $m\beta(P)$

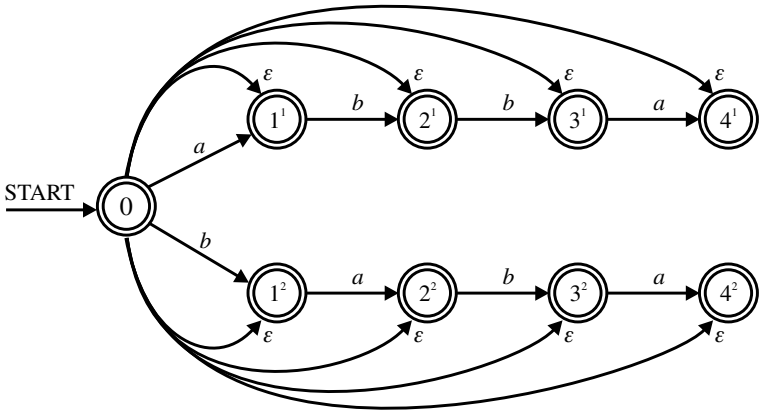

Figure 12.3: Transition diagram of the factor automaton with ε-transitions for the set of patterns P=$\{abba, baba\}$ from Exercise 12.1

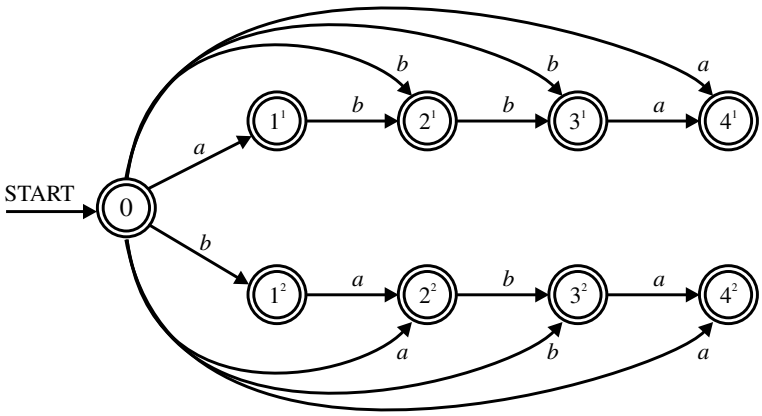

Figure 12.4: Transition diagram of the nondeterministic factor automaton for the set of patterns P=$\{abba, baba\}$ from Exercise 12.1

is shown in Table 12.3. Transition diagram of AC automaton is depicted in Fig. 12.6. Fail function φ_{opt} is shown only in case when it is different from fail function φ. □

	a	b
0	$1^1, 4^1, 2^2, 4^2$	$2^1, 3^1, 1^2, 3^2$
1^1		2^1
2^1		3^1
3^1	4^1	
4^1		
1^2	2^2	
2^2		3^2
3^2	4^2	
4^2		

a)

	a	b
0	$1^1 4^1 2^2 4^2$	$2^1 3^1 1^2 3^2$
$1^1 4^1 2^2 4^2$		$2^1 3^2$
$2^1 3^1 1^2 3^2$	$4^1 2^2 4^2$	3^1
$2^1 3^2$	4^2	3^1
3^1	4^1	
4^1		
$4^1 2^2 4^2$		3^2
3^2	4^2	
4^2		

b)

Table 12.1: Transition tables of both nondeterministic and deterministic factor automata for the set of patterns P={*abba,baba*} from Exercise 12.1

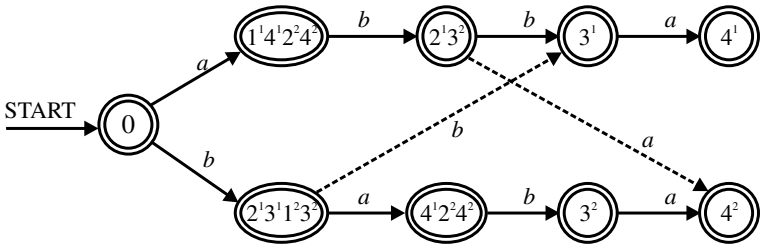

Figure 12.5: Transition diagram of the deterministic factor automaton for the set of patterns P={*abba,baba*} from Exercise 12.1

Analysed state	Values of m-border array elements
$1^1 4^1 2^2 4^2$	$m\beta[4^1] = m\beta[4^2] = m\beta[2^2] = 1^1$
$2^1 3^2$	$m\beta[3^2] = 2^1$
$2^1 3^1 1^2 3^2$	$m\beta[2^1] = m\beta[3^1] = m\beta[4^1] = 1^2$
$4^1 2^2 4^2$	$m\beta[4^1] = m\beta[4^2] = 2^2$

Table 12.2: Analysis of the multiple d-subsets of the backbone of the factor automaton from Exercise 12.1

State	1^1	2^1	3^1	4^1	1^2	2^2	3^2	4^2
Symbol	a	b	b	a	b	a	b	a
$m\beta$[state]	0	1^2	1^2	2^2	0	1^1	2^1	2^2

Table 12.3: The m-border array $m\beta(P)$ from Example 12.1

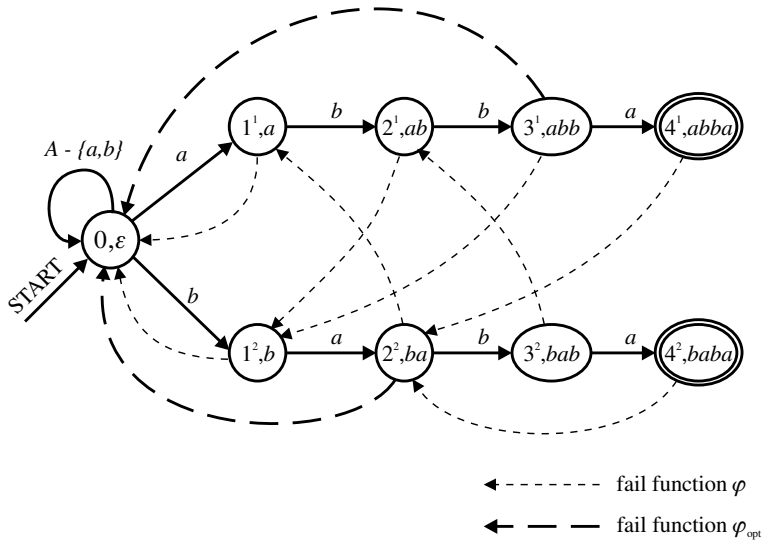

Figure 12.6: Transition diagram of AC searching automaton for the set of patterns P=$\{abba, baba\}$ from Exercise 12.1

Exercise 12.2
Construct AC searching automaton for searching of the set of patterns P={*k,ok,rok,brok, obrok*}. Nondeterministic searching automaton for the finite set of pattern P (*SFFECO* automaton, see [TSA, Chapter 2]) has transition diagram depicted in Fig. 12.7.

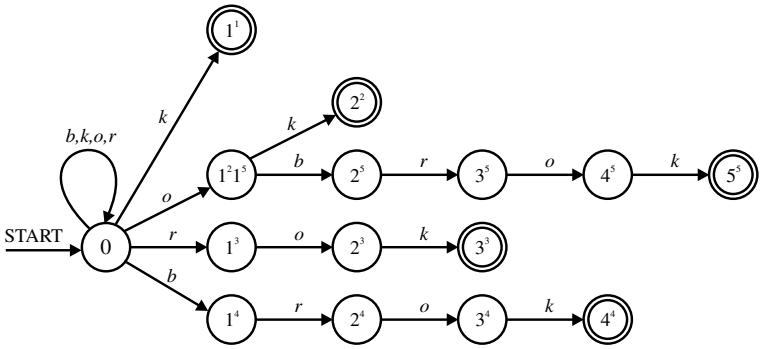

Figure 12.7: Transition diagram of the nondeterministic searching (*SFFOECO*) automaton for the set of patterns P={*k,ok,rok,brok,obrok*} from Exercise 12.2

Construction of the prefix automaton and factor automaton with ε-transitions is left to the reader. Nondeterministic factor automaton after the removal of ε-transitions has transition diagram depicted in Fig. 12.8.

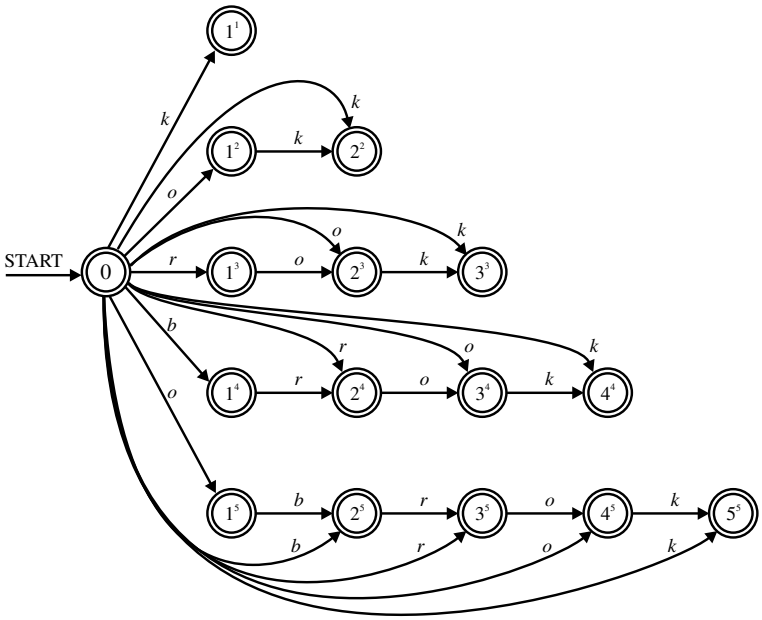

Figure 12.8: Transition diagram of the nondeterministic factor automaton for the set of patterns P={*k,ok,rok,brok,obrok*} from Exercise 12.2

Next step is a determinisation of the factor automaton. Transition tables of both nondeterministic and deterministic factor automata are shown in Table 12.4. Transition diagram of the deterministic factor automaton is depicted in Fig. 12.9. An analysis of the multiple d-subsets is shown in Table 12.5. The resulting m-border array $m\beta(P)$ is shown in Table 12.6.

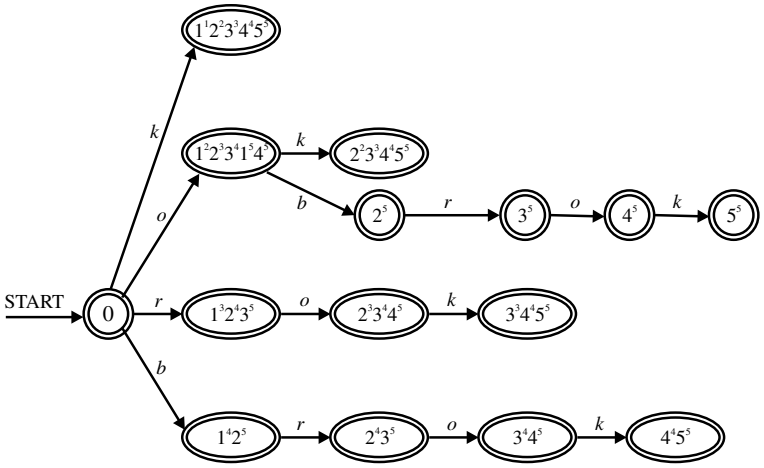

Figure 12.9: Transition diagram of the deterministic factor automaton for the set of patterns P={*k,ok,rok,brok,obrok*} from Exercise 12.2

Transition diagram of the AC automaton is depicted in Fig. 12.10.Fail function φ_{opt} is shown

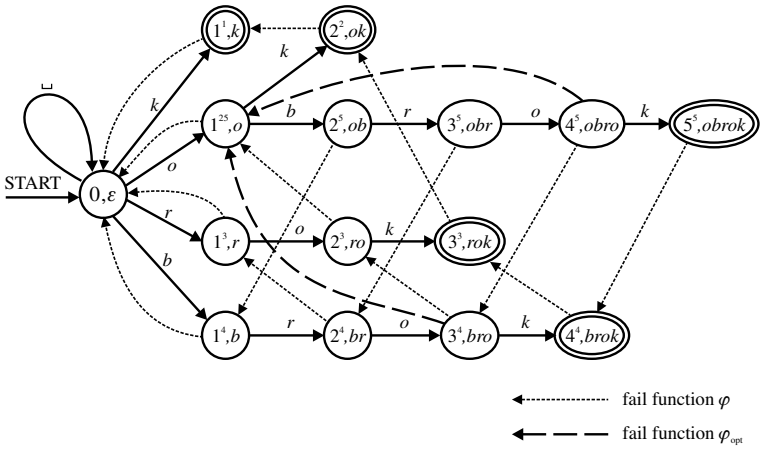

Figure 12.10: Transition diagram of AC searching automaton for the set of patterns P={*k,ok,rok,brok,obrok*} from Exercise 12.2

only in case, when it differs from fail function φ. For text $T = ob␣rok␣obroky$ this automaton makes the following sequence of transitions:
$\varepsilon \xrightarrow{o} o \xrightarrow{b} ob - \to b - \to \varepsilon \xrightarrow{␣} \varepsilon \xrightarrow{r} r \xrightarrow{o} ro \xrightarrow{k} \underline{rok} - \to \underline{ok} - \to \underline{k} - \to \varepsilon \xrightarrow{␣} \varepsilon \xrightarrow{o}$
$o \xrightarrow{b} ob \xrightarrow{r} obr \xrightarrow{o} obro \xrightarrow{k} \underline{obrok} - \to \underline{brok} - \to \underline{rok} - \to \underline{ok} - \to \underline{k} - \to \varepsilon \xrightarrow{y} \varepsilon$
The found elements of the set of patterns P are underlined. □

	b	k	o	r
0	$1^4, 2^5$	$1^1, 2^2, 3^3, 4^4, 5^5$	$1^2, 2^3, 3^4, 1^5, 4^5$	$1^3, 2^4, 3^5$
1^1				
1^2		2^2		
2^2				
1^3			2^3	
2^3		3^3		
3^3				
1^4				2^4
2^4			3^4	
3^4		4^4		
4^4				
1^5	2^5			
2^5				3^5
3^5			4^5	
4^5		5^5		
5^5				

a)

	b	k	o	r
0	$1^4 2^5$	$1^1 2^2 3^3 4^4 5^5$	$1^2 2^3 3^4 1^5 4^5$	$1^3 2^4 3^5$
$1^1 2^2 3^3 4^4 5^5$				
$1^2 2^3 3^4 1^5 4^5$	2^5	$2^2 3^3 4^4 5^5$		
$2^2 3^3 4^4 5^5$				
$1^3 2^4 3^5$			$2^3 3^4 4^5$	
$2^3 3^4 4^5$				$2^4 3^5$
$3^3 4^4 5^5$				$2^4 3^5$
$1^4 2^5$				$2^4 3^5$
$2^4 3^5$			$3^4 4^5$	
$3^4 4^5$		$4^4 5^5$		
$4^4 5^5$				
2^5				3^5
3^5			4^5	
4^5		5^5		
5^5				

b)

Table 12.4: Transition tables of both nondeterministic and deterministic factor automata for the set of patterns P={*k,ok,rok,brok,obrok*} from Exercise 12.2

Analyzed state	Values of m-border array elements
$1^1 2^2 3^3 4^4 5^5$	$m\beta[2^2] = m\beta[3^3] = m\beta[4^4] = m\beta[5^5] = 1^1$
$1^2 2^3 3^4 5^5 4^5$	$m\beta[2^3] = m\beta[3^4] = m\beta[4^5] = 1^{25}$
$2^2 3^3 4^4 5^5$	$m\beta[3^3] = m\beta[4^4] = m\beta[5^5] = 2^2$
$1^3 2^4 3^5$	$m\beta[2^4] = m\beta[3^5] = 1^3$
$2^3 3^4 4^5$	$m\beta[3^4] = m\beta[4^5] = 2^3$
$3^3 4^4 5^5$	$m\beta[4^4] = m\beta[5^5] = 3^3$
$1^4 2^5$	$m\beta[2^5] = 1^4$
$2^4 3^5$	$m\beta[3^5] = 2^4$
$3^4 4^5$	$m\beta[4^5] = 3^4$
$4^4 5^5$	$m\beta[5^5] = 4^4$

Table 12.5: Analysis of multiple d-subsets of the deterministic factor automaton for the set of patterns P={*k,ok,rok,brok,obrok*} from Exercise 12.2

State	1^1	2^1	3^1	4^1	1^2	2^2	3^2	4^2
Symbol	a	b	b	a	b	a	b	a
$m\beta$[state]	0	1^2	1^2	2^2	0	1^1	2^1	2^2

Table 12.6: m-border array $m\beta$ for the set of patterns P={*k,ok,rok,brok,obrok*} from Exercise 12.2

Exercise 12.3
Construct AC searching automaton for searching of the set of patterns P={*hers,he,his,she*}. Nondeterministic searching automaton for the finite set of patterns P (*SFFECO* automaton, see [TSA, Chapter 2]) has transition diagram depicted in Fig. 12.11.

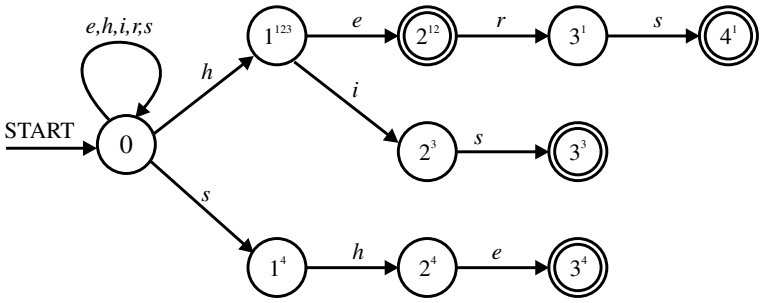

Figure 12.11: Transition diagram of the nondeterministic searching automaton *SFFOECO* automaton for the set of patterns $P = \{hers, he, his, she\}$ from Exercise 12.3

Construction of the prefix automaton and factor automaton with ε-transitions is left to the reader. Nondeterministic factor automaton after the removal of ε-transitions has transition diagram depicted in Fig. 12.12.

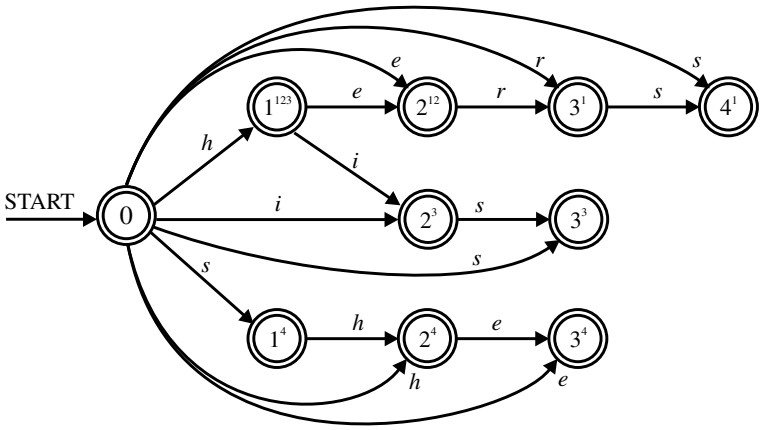

Figure 12.12: Transition diagram of the nondeterministic factor automaton *SFFOECO* automaton for the set of patterns P={*hers,he,his,she*} from Exercise 12.3

The next step is a determinisation of factor automaton. Transition tables of both nondeterministic and deterministic factor automata are shown in Table 12.7. Transition diagram of the deterministic factor automaton is depicted in Fig. 12.13. An analysis of the multiple d-subsets is shown in Table 12.8. The resulting m-border array $m\beta(P)$ is shown in the Table 12.9. The transition diagram of the resulting AC automaton is depicted in Fig. 12.14. As each state of the automaton corresponds to a prefix of some element of some the set of patterns P, we add this prefixes to the labeling of states. Fail function φ_{opt} is in this case equal to the fail function φ. For text T=*reshersche* this automaton makes the following sequence of transitions:
$\varepsilon \xrightarrow{r} \varepsilon \xrightarrow{e} \varepsilon \xrightarrow{s} s \xrightarrow{h} sh \xrightarrow{e} \underline{she} - \rightarrow \underline{he} \xrightarrow{r} her \xrightarrow{s} \underline{hers} - \rightarrow s - \rightarrow \varepsilon \xrightarrow{c} \varepsilon \xrightarrow{h} h \xrightarrow{e} \underline{he}$

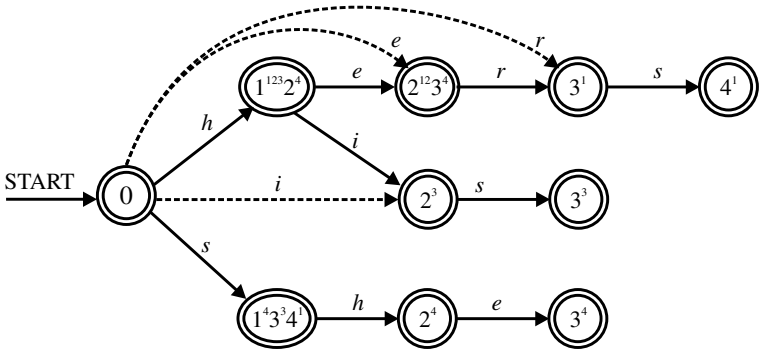

Figure 12.13: Transition diagram of the deterministic factor automaton for the set of patterns $P = \{hers, he, his, she\}$ from Exercise 12.3

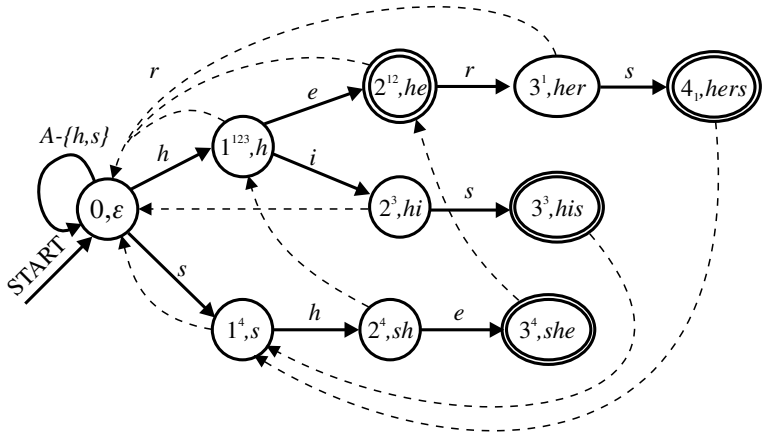

Figure 12.14: Transition diagram of AC searching automaton for set of patterns $P = \{hers, he, his, she\}$ from Exercise 12.3

The found elements of the set of patterns are underlined. □

	e	h	i	r	s
0	$2^{12}, 3^4$	$1^{123}, 2^4$	2^3	3^1	$1^4, 3^3, 4^1$
1^{123}	2^{12}		2^3		
2^{12}			2^3		
3^1					4^1
4^1					
2^3					3^3
3^3					
1^4		2^4			
2^4	3^4				
3^4					

a)

	e	h	i	r	s
0	$2^{12}3^4$	$1^{123}2^4$	2^3	3^1	$1^4 3^3 4^1$
$1^{123}2^4$	$2^{12}3^4$		2^3		
$2^{12}3^4$				3^1	
3^1					4^1
4^1					
2^3					3^3
3^3					
1^4		2^4			
2^4	3^4				
3^4					

b)

Table 12.7: Transition tables of both nondeterministic and deterministic factor automata for the set of patterns P={*hers,he,his,she*} from Exercise 12.3

Analysed state	Values of mborder array elements
$1^{123}2^4$	$m\beta[2^4] = 1^{123}$
$2^{12}3^4$	$m\beta[3^4] = 2^{12}$
$1^4 3^3 4^1$	$m\beta[3^3] = m\beta[4^1] = 1^4$

Table 12.8: Analysis of multiple d-subsets of the deterministic factor automaton for the set of patterns $P = \{$*hers,he,his,she*$\}$ from Exercise 12.3

State	1^{123}	1^4	2^{12}	2^3	2^4	3^1	3^3	3^4	4^1
Symbol	h	s	e	i	h	r	s	e	s
$m\beta$[state]	0	0	0	0	1^{123}	0	1^4	2^{12}	1^4

Table 12.9: m-border array $m\beta(P)$ for $P = \{\{$*hers,he,his,she*$\}$ from Exercise 12.3

13 Backward pattern matching of one pattern

13.1 Boyer–Moore–Horspool algorithm

The simplest method of backward searching is shown here. It is *BMH* (Boyer–Moore–Horspool) algorithm, which use the heuristic called "bad character shift". The computation of shifts is very simple. Shifts are computed for all symbols of an alphabet and are equal to the distance symbol in the pattern from the end of the pattern.

Exercise 13.1
Let pattern be $P = abcaba$ over alphabet $A = \{a, b, c, d\}$. We compute the table of shifts (BCS table) for BMH algorithm. Moreover we make search in text $T = bcbaabcabab$. First the nondeterministic factor automaton is constructed for reversed pattern P^R. Its transition diagram is depicted in Fig. 13.1. It is enough to compute the first row of the transition table

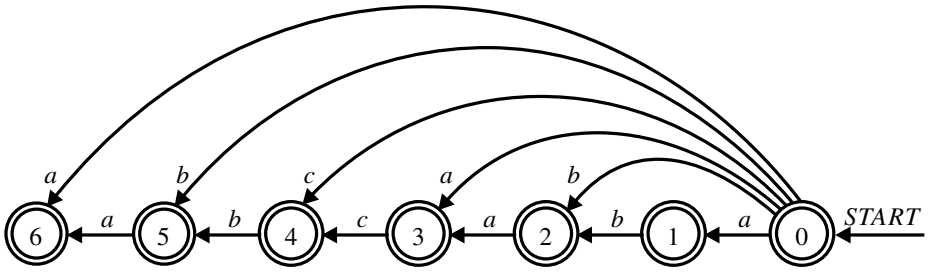

Figure 13.1: Transition diagram of the nondeterministic factor automaton for pattern $P^R = abacab$ from Exercise 13.1

of the deterministic factor automaton:

	a	b	c
0	136	25	4

It is selected for each symbol of the alphabet a state having the shortest distance from the end of the pattern greater than one and the one is subtracted. The shift for each symbol is computed this way. The result is the BCS table of shifts:

symbol	a	b	c
$BCSshift$	2	1	3

For symbols of the alphabet which do not occur in the pattern is the shift equal to the length of the pattern. In our case the shift for symbol d is equal to 6. This procedure can be written in the form of program in this way:

(1) $for\ symbol := symb[1]\ to\ symb[|A|]\ do\ shift\ [symbol] := m;$
(2) $for\ j := 1\ to\ m-1\ do\ shift\ [P[j]] := m - j;$

The computation of shifts for pattern $P = abcaba$ is shown in this table:

	Initial	for loop(2)				
	setting (1)	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$
a	6	5			②	
b	6		4			①
c	6			③		
d	⑥					

Circles show the computed shifts for all symbols.
Now the search of pattern $P = abcaba$ in text $T = bcbaabcabab$ is performed:

bcbaabcabab	mismatch
abcaba	shift[*b*]=1
bcbaabcabab	mismatch
abcaba	shift[*c*]=3
bcbaabcabab	match
abcaba	shift[*a*]=2

The shift is now behind the end of the text and the search is finished.

13.2 Looking for repeated suffixes

The heuristics used for the looking for repeated suffixes is called "good suffix shift" (GSS) and the tool for this approach is the backbone of the factor automaton.

Exercise 13.2
Let pattern be $P = babba$. We compute GSS table for the pattern P. We start by constructing the backbone of the factor automaton for reversed pattern $P^R = (babba)^R = abbab$. Transition diagram of the nondeterministic factor automaton for pattern *abbab* is depicted in Fig. 13.2. Its deterministic equivalent has transition diagram depicted in Fig. 13.3. Deterministic factor

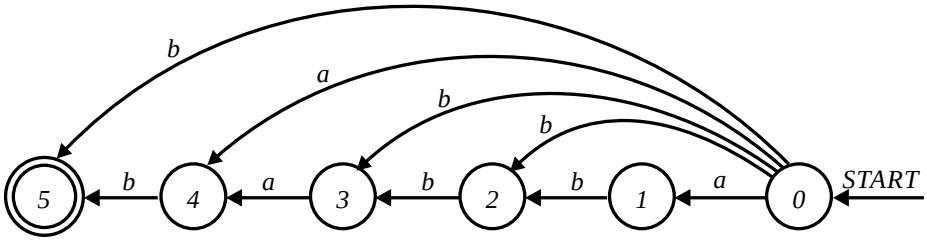

Figure 13.2: Transition diagram of the nondeterministic factor automaton for pattern $P^R = abbab$ from Exercise 13.2

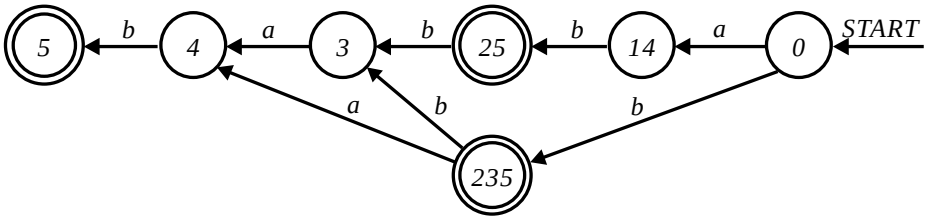

Figure 13.3: Transition diagram of the deterministic factor automaton for pattern $P^R = abbab$ from Exercise 13.2

automaton for pattern *abbab* shows this facts:

a) suffix *a* is repeating at positions 1 and 4,

b) suffix *ab* is repeating at positions 2 and 5,

c) factor b is repeating at positions 2,3 and 5.

From the point of view of our task only the repetitions of suffixes are interesting. The repetitions are given by multiple states on the backbone of the factor automaton. State 235 and the transitions connected with it are outside of the backbone and may be removed. Transition diagram of the backbone of factor automaton is depicted in Fig 13.4. The suffix repetition table has the following form:

d-subset	suffix	repetitions
14	a	$(1,F),(4,G)$
25	ba	$(2,F),(5,G)$

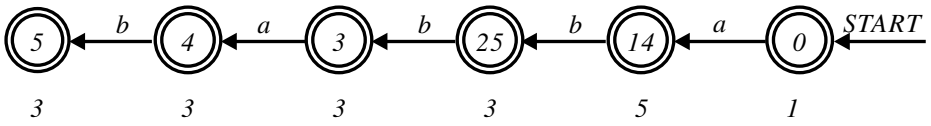

Figure 13.4: Backbone of the factor automaton for pattern $P^R = abbab$ from Exercise 13.2

The GSS table has the following form:

state	0	14	25	3	4	5
suffix	ε	a	ba	bba	$abba$	$babba$
GSS(state)	1	3	3	5	5	5
GSS_{opt}(state)	1	5	3	5	5	5

There is a possible optimisation of the GSS table for state 14. The suffix a is repeating three positions to the left but with the same symbol in front of it. Therefore we can enlarge the GSS shift to 5. □

Exercise 13.3
Let pattern be $P = ababa$. We compute GSS table for the pattern P. We start by constructing backbone for the reversed pattern $P^R = ababa$. We construct factor automaton M_1 for this reversed pattern. Transition diagram of nondeterministic factor automaton M_1 is depicted in Fig. 13.5.All its states are final states. The next step is the construction of equiva-

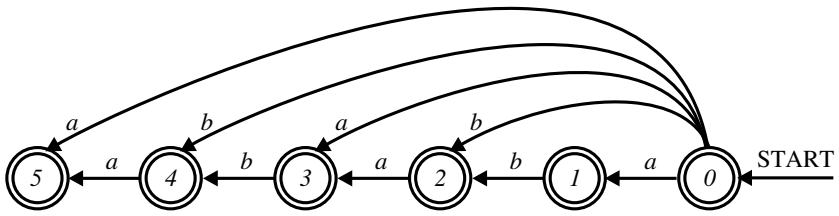

Figure 13.5: Transition diagram of nondeterministic factor automaton M_1 for reversed pattern $P^R = ababa$ from Exercise 13.3

lent deterministic factor automaton M_2. During this construction we save created d–subsets. Transition tables of both nondeterministic factor automaton M_1 and its deterministic equivalent M_2 are shown in Table 13.1. Transition diagram of deterministic factor automaton M_2 is depicted in Fig. 13.6.To obtain backbone of factor automaton M_2, we remove transitions from state 0 it state 24 for input symbol b. Now we construct the suffix repetition table. It has the following form:

	a	b
0	$1,3,5$	$2,4$
1		2
2	3	
3		4
4	5	
5		

a) transition table of M_1

	a	b
0	135	24
135		24
24	35	
35		4
4	5	
5		

b) transition table of M_2

Table 13.1: Transition tables of factor automata M_1 and M_2 from Exercise 13.3

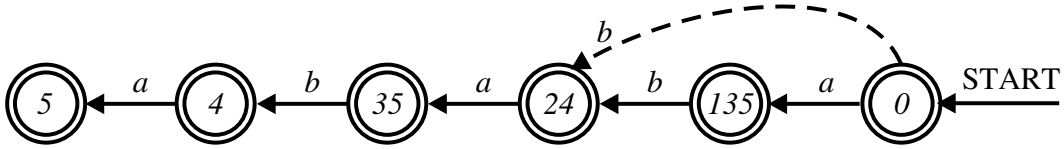

Figure 13.6: Transition diagram of deterministic factor automaton M_2 for $P^R = ababa$ from Exercise 13.3

d-subset	Suffix	Repetitions
135	a	$(1,F),(3,G),(5,G)$
24	ba	$(2,F),(4,S)$
35	aba	$(3,F),(5,O)$

□

Sem prijde vsuvka zbytek Ex. 7.4+Ex.7.5 z Vol II.

Exercise 13.4

Let pattern be $P = aabaa$. We compute the *GSS* table. Transition diagrams of the nondeterministic factor automaton and the backbone of deterministic factor automaton for pattern P are depicted in Fig. 13.7. This pattern has two suffixes a and aa, which are also prefixes.

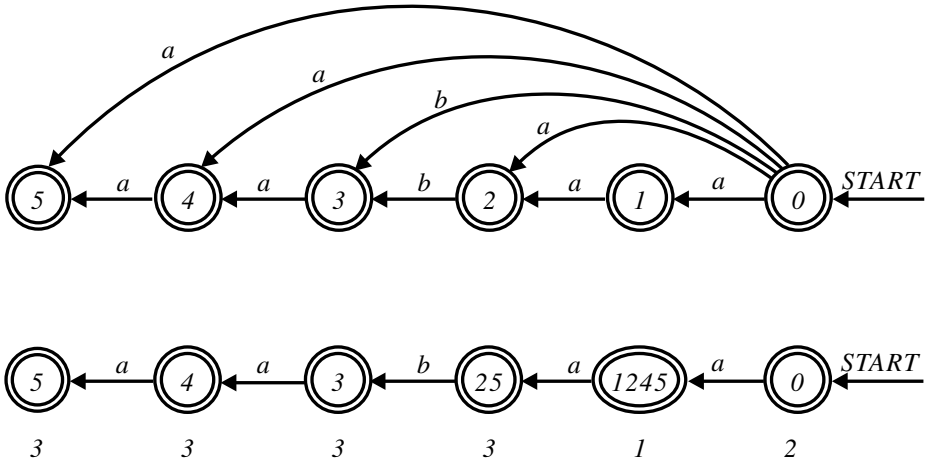

Figure 13.7: Transition diagrams of the nondeterministic factor automaton and deterministic factor automaton for pattern $P^R = aabaa$ from Exercise 13.4

Suffix repetition table has the following form:

d-subset	Suffix	Repetitions
1245	a	$(1,F),(2,S),(4,G),(5,S)$
25	aa	$(2,F),(5,G)$

According to the longest repeated suffix aa the shift in states 25,3,4 and 5 is computed. It has value 3 (5-2=3). The shift in state 0 is 2, because the last but one symbol is equal to the last one and it is again a. The shift in state 1245 is 1, because the found suffix a is repeating in the distance 1. Resulting GSS table has the following form:

state	0	1245	25	3	4	5
GSS(state)	2	1	3	3	3	3

□

Exercise 13.5
Let pattern be $P = bbaba$. Compute the GSS table. Transition diagrams of the nondeterministic factor automaton and the backbone of deterministic factor automaton for pattern P are depicted in Fig. 13.8.In this case two suffixes a and ba are repeating factors of the pattern.

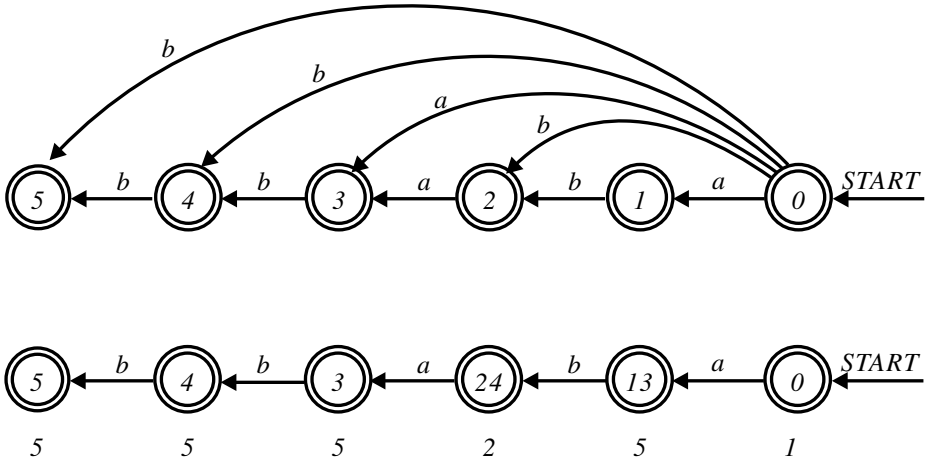

Figure 13.8: Transition diagram of nondeterministic factor automaton and the backbone of factor automaton for pattern $P^R = ababb$ from Exercise 13.5

Suffix repetition table has the following form:

d-subset	Suffix	Repetitions
13	a	$(1,F),(3,G)$
24	ba	$(2,F),(4,O)$

The GSS table has the following form:

state	0	13	24	3	4	5
Suffix	ε	a	ba	aba	$baba$	$bbaba$
GSS(state)	1	2	2	5	5	5
GSS_{opt}(state)	1	5	2	5	5	5

There is a possible optimisation of this GSS table for state 13. The suffix a is repeating two positions to the left but with the same symbol in front of it. Therefore we can enlarge the GSS shift to 5.

□

Exercise 13.6
Let pattern be $P = babaa$. Compute GSS table. Transition diagrams of the nondeterministic factor automaton and the backbone of the deterministic factor automaton for pattern P are depicted in Fig. 13.9. Suffix repetition table has the following form:

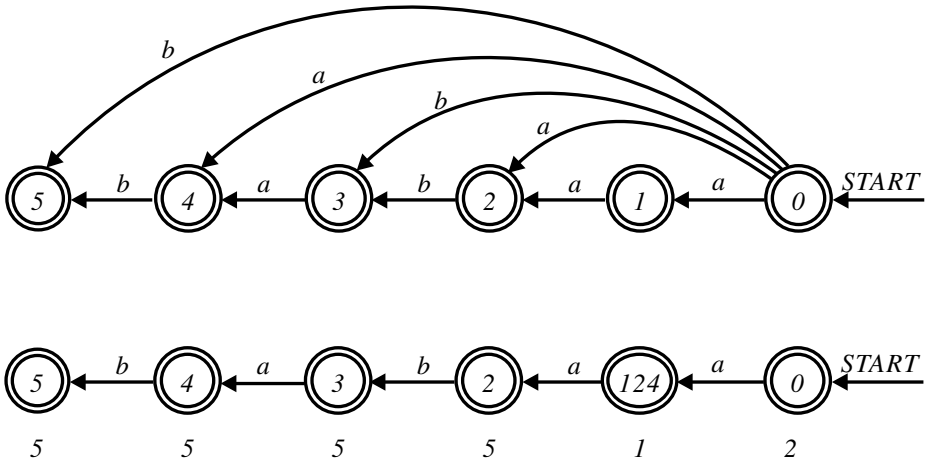

Figure 13.9: Transition diagrams of the nondeterministic factor automaton and the backbone of the deterministic factor automaton for pattern $P^R = aabab$ from Exercise 13.6

d-subset	Suffix	Repetitions
124	a	$(1, F), (2, S), (4, G)$

The GSS table has the following form:

state	0	124	2	3	4	5
Suffix	ε	a	aa	baa	$abaa$	$babaa$
GSS(state)	1	1	5	5	5	5
GSS_{opt}(state)	2	1	5	5	5	5

There is possible optimization of GSS table for state 0. The mismatch symbol at the end of pattern is a. The last but one symbol in the pattern is again a. Therefore using this shift the mismatch appears again. The shift can be enlarged to 2. □

Exercise 13.7
Let pattern be $P = aaaaa$. Compute GSS table. Transition diagrams of the nondeterministic factor automaton and the backbone of the deterministic factor automaton for pattern P are depicted in Fig. 13.10. Suffix repetition table has the following form:

d-subset	Suffix	Repetitions
12345	a	$(1, F), (2, S), (3, S), (4, S), (5, S)$
2345	aa	$(2, F), (3, O), (4, S), (5, O)$
345	aaa	$(3, F), (4, O), (5, O)$
45	$aaaa$	$(4, F), (5, O)$

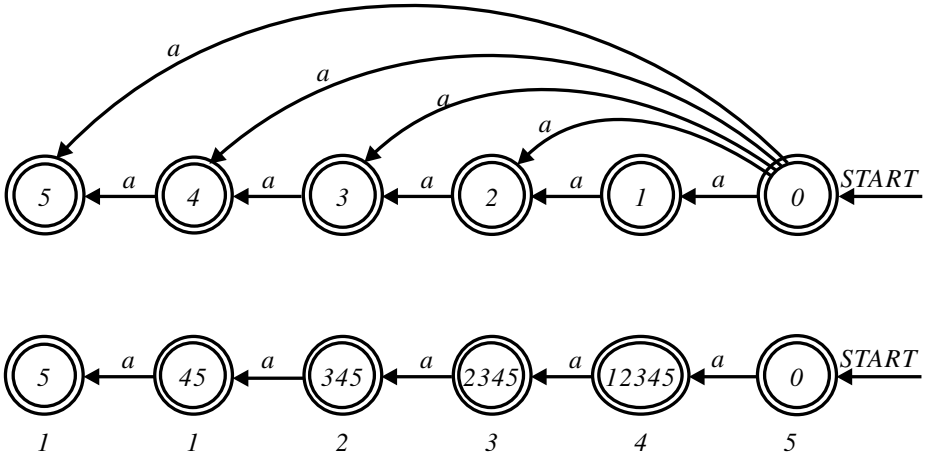

Figure 13.10: Transition diagrams of the nondeterministic factor automaton and the backbone of the deterministic factor automaton for pattern $P^R = aaaaa$ from Exercise 13.7

The GSS table has the following form:

state	0	12345	2345	345	45
Suffix	ε	a	aa	aaa	$aaaa$
GSS(state)	1	1	1	1	1
GSS_{opt}(state)	5	4	3	2	1

There are possible optimisations of GSS table. The last symbol a is repeating in all positions, therefore GSS_{opt} shift can be 5. All other suffixes (aa,aaa,aaaa) are repeating with the same symbol in front of them. Therefore GSS_{opt} can be optimised as shown in the preceding table. □

14 Backward pattern matching in text – searching for prefixes and antifactors

Basic tool for searching prefixes of the pattern is a suffix automaton constructed for reversed pattern. Suffix automaton is accepting finite languages $Suf(P^R)$ composed of all suffixes of reversed pattern P^R. It means, that it accepts all prefixes of pattern P during reading of the text from right to left.

14.1 Exact backward searching of prefixes of pattern in text

Exercise 14.1
Let pattern be *baba*. Construct suffix automaton for reversed pattern $baba^R$, which accepts the set $Suf(baba^R)$ composed of all suffixes of string *abab*. Use this automaton for backward searching of prefixes of the pattern. $Suf(baba^R) = \{\varepsilon, b, ab, bab, abab\}$. Set $Suf(baba^R)$ contains all reversed prefixes of pattern *baba*. We start by construction of finite automaton M_1, which accepts the reversed pattern. Its transition diagram is depicted in Fig. 14.1.

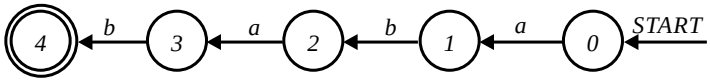

Figure 14.1: Transition diagram of automaton M_1 accepting string *abab* from Exercise 14.1

The next step is the addition of ε-transitions from initial state to all other states. The result is the finite automaton M_2 having transition diagram depicted in Fig. 14.2.

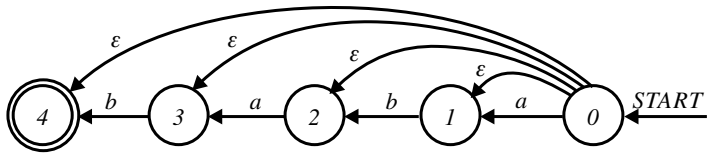

Figure 14.2: Transition diagram of finite automaton M_2 accepting all suffixes of string $abab = (baba)^R$ from Exercise 14.1

To obtain deterministic suffix automaton the two standard operations will be used - removal ε-transitions and determinisation. Suffix automaton M_3 after removal ε-transitions is depicted in Fig. 14.3.

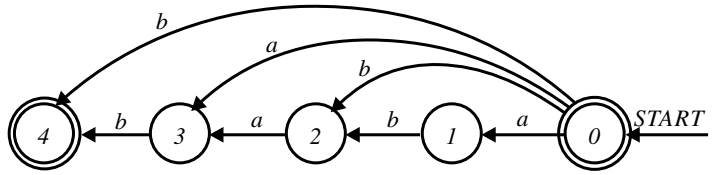

Figure 14.3: Transition diagram of the suffix automaton M_3 after removal of ε-transitions

In case, when no symbol in pattern is repeating which means that all symbols of the pattern are different, automaton M_3 is deterministic and the procedure is finished. If at least one symbol in pattern is repeating then, automaton M_3 is nondeterministic and it is

necessary to make the determinisation. In our case transition diagram of the deterministic suffix automaton M_4 is depicted in Fig. 14.4. □

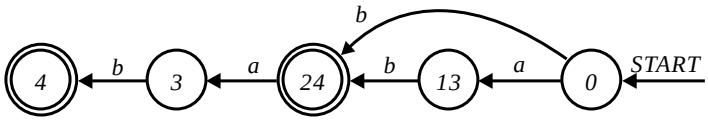

Figure 14.4: Transition diagram of the deterministic suffix automaton M_4 from Exercise 14.1

We use suffix automaton M_4 from Exercise 14.1 for searching of pattern $P = baba$ in text $T = aaaabbabab$. The pattern is placed at the beginning of the text and the longest prefix of the pattern is backward searched. If the shortest prefix is found then the shift is given by the difference of the pattern and the length of the found prefix. If no prefix is found the shift is equal to the length of the pattern. The longest prefix is then searching in all cases. The algorithm for the searching of prefixes uses suffix automaton for reversed pattern (in our case M_4). For determination of the length of shift the following variables will be used:

m – length of pattern,

$position$ – position of symbol in text, corresponding to the last symbol of the pattern,

$tcounter$ – transition counter,

$lprefix$ – length of the longest found prefix.

The configuration of the algorithm is a quadruple $(q, position - tcounter, tcounter, lprefix)$, where q is the state of suffix automaton. The initial configuration is $(0, m, 0, 0)$.

For text $T =$

1	2	3	4	5	6	7	8	9	10
a	a	a	a	b	b	a	b	a	b

and pattern $P = baba$

the searching algorithm performs this sequence of transitions:

$$(0, 4, 0, 0) \quad \vdash^a \quad (13, 3, 1, 0).$$

The next transition in the suffix automaton is not possible and the length of prefix is 0, therefore the shift will be $4 - 0$ positions in text to the right:

$$\begin{aligned} &\vdash \quad (0, 8, 0, 0) \\ &\vdash^b \quad (24, 7, 1, 1) \\ &\vdash^a \quad (3, 6, 2, 1) \\ &\vdash^b \quad (4, 5, 3, 3). \end{aligned}$$

The next transition is not possible and the length of prefix is 3, therefore the shift will be $4 - 3 = 1$ position to the right:

$$\begin{aligned} &\vdash \quad (0, 9, 0, 0) \\ &\vdash^a \quad (13, 8, 1, 0) \\ &\vdash^b \quad (24, 7, 2, 2) \\ &\vdash^a \quad (3, 6, 3, 2) \\ &\vdash^b \quad (4, 5, 4, 4). \end{aligned}$$

The pattern is found, shift has the length 2 (length of period) to the right:

$$\vdash \quad (0, 11, 0, 0).$$

Because *position* is greater the length of text, searching is finished. It can be seen from this sequence of transitions that the algorithm is not effective, because the found prefix is checked up again. The algorithm can be modified order to avoid the checking of found prefix again. An additional variable will be introduced:

ncomp –number of necessary comparisons.

Its initial value will be the length of shift and during every transition it is decreased by one. Its value is equal to zero if the pattern is found.

The configuration of the algorithm will be in this case fivetuple $(q, position - tcounter, tcounter, lprefix, ncomp)$. The initial configuration is $(0, m, 0, 0, m)$.

For text T the searching algorithm performs the sequence of transitions:

$$
\begin{array}{lll}
(0,4,0,0,4) \vdash^{a} (13,3,1,0,3) & \text{shift by 4} \\
\vdash (0,8,0,0,4) & \\
\vdash^{b} (24,7,1,1,3) & \\
\vdash^{a} (3,6,2,1,2) & \\
\vdash^{b} (4,5,3,3,1) & \text{shift by 1} \\
\vdash (0,9,0,0,1) & \\
\vdash^{a} (13,8,1,0,0) & \text{pattern found, shift by 2} \\
\vdash (0,11,0,0,2) & \text{finish.}
\end{array}
$$

In this case the automaton performed 8 transitions, while in the preceding case it performed 11 transitions. □

15 Backward pattern matching of finite set of patterns

15.1 Backward searching finite set of patterns – searching of repeating suffixes

Exercise 15.1
Let set of patterns be $S = \{bbaba, babaa\}$ over alphabet $A = \{a, b\}$. Compute shifts for it. Transition diagrams of nondeterministic suffix automaton and the backbone of the deterministic suffix automaton for set S^R are depicted in Fig. 15.1.Transition table of the backbone

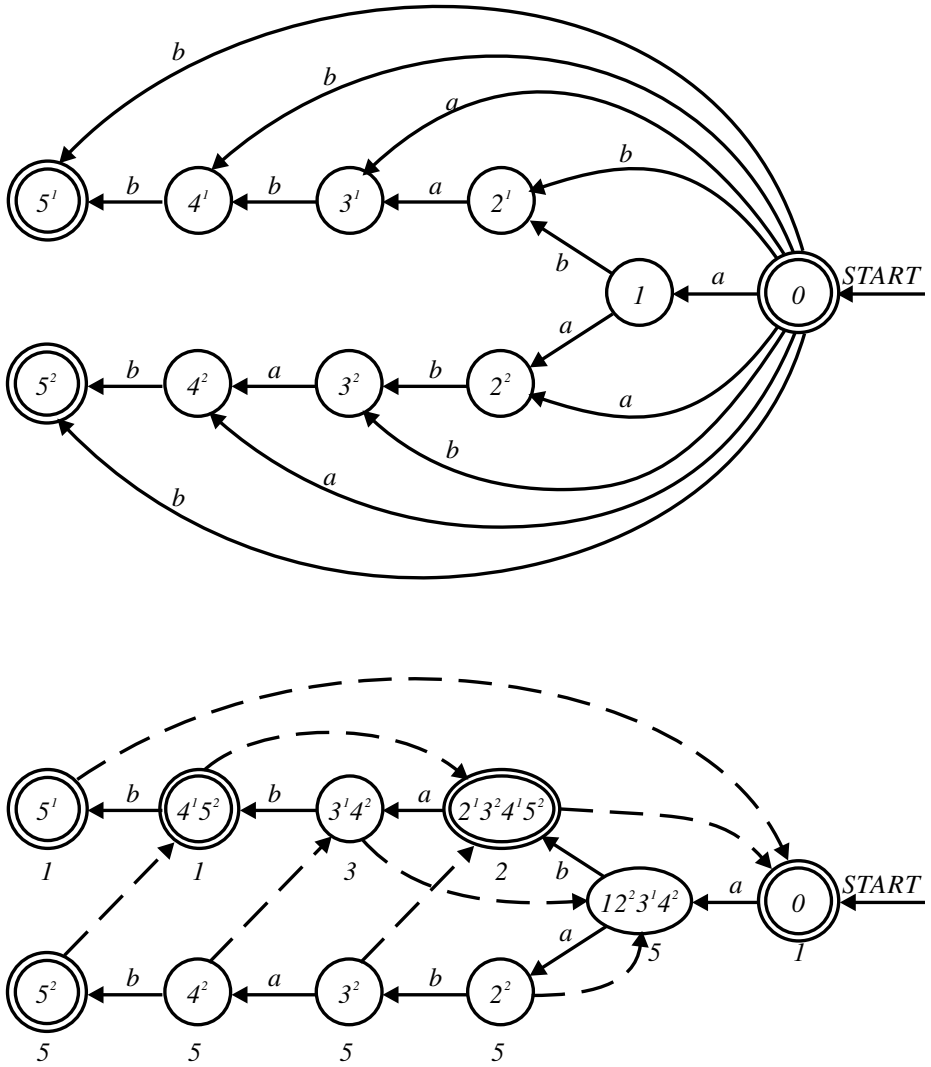

Figure 15.1: Transition diagrams of the nondeterministic suffix automaton and the backbone of the deterministic suffix automaton for set of patterns $S^R = \{bbaba^R, babaa^R\}$ from Exercise 15.1

is shown in Table 15.1. In this table are shown only transitions of the backbone of the deterministic suffix automaton. Suffix repetition table (GSS table) has the following form:

d-subset	Suffix	Repetititons
$12^2 3^1 4^2$	a	$(1,F),(2^2,S),(3^1,G),(4^2,G)$
$2^1 3^2 4^1 5^2$	ba	$(2^1,F),(3^2,F),(4^1,S),(5^2,S)$
$3^1 4^2$	aba	$(3^1,F),(4^2,F)$
$4^1 5^2$	$abab$	$(4^1,F),(5^2,F)$

The GSS and GSS_{opt} table has the following form:

state	0	$12^2 3^1 4^2$	$2^1 3^2 4^1 5^2$	$3^1 4^2$	$4^1 5^2$	5^1	2^2	3^2	4^2	5^2
Suffix	ε	a	ba	aba	$abab$	$bbaba$	aa	baa	$abaa$	$babaa$
GSS(state)	1	$*$	1	1	1	5	5	5	5	5
GSS_{opt}(state)	1	$*$	2	5	1	5	5	5	5	5

$*$ – Shift is not defined, because transitions for both a and b $\delta(12^2 3^1 4^2, a)$ and $\delta(12^2 3^1 4^2, b)$ are defined. □

	a	b
0	$12^2 3^1 4^2$	
$12^2 3^1 4^2$	2^2	$2^1 3^2 4^1 5^2$
$2^1 3^2 4^1 5^2$	$3^1 4^2$	
$3^1 4^2$		$4^1 5^2$
$4^1 5^2$		5^1
2^2		3^2
3^2	4^2	
4^2		5^2

Table 15.1: Transition table of the backbone of deterministic suffix automaton for set $S^R = \{ababb, aabab\}$ from Exercise 15.1

16 Approximate backward pattern matching

16.1 Searching for approximate prefixes

The basic tool for backward pattern matching of approximate prefixes is an approximate suffix automaton constructed for reversed pattern. The Hamming distance is used in the next exercises. The Hamming suffix automaton accepts finite language $HSuff_k(P)$ composed of all suffixes of pattern P and all strings having Hamming distance less or equal to k from this suffixes.

Exercise 16.1
Construct approximate good prefix shift ($AGPS$) table for pattern $P = baba$ for Hamming distance $k = 1$. First we construct the Hamming suffix automaton $HSuff_1(P)$ for reversed pattern $P^R = baba^R = abab$. This construction is started by construction of finite automaton M_1 accepting reversed pattern and all string having Hamming distance equal to one from it. Transition diagram of this automaton is depicted in Fig. 16.1

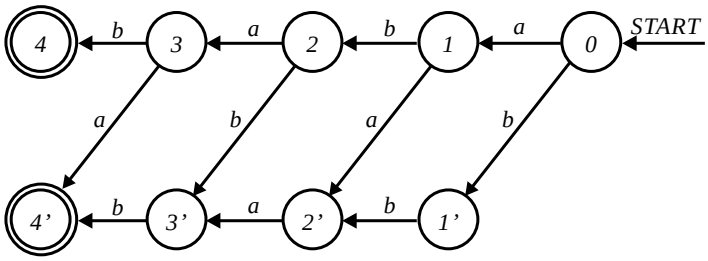

Figure 16.1: Transition diagram of automaton M_1 accepting string $P = baba^R = abab$ and strings with distance one from pattern P from Exercise 16.1

The next step is an addition of ε-transitions from the initial state to the zero level states (it corresponds to the exact suffixes). The automaton M_2 with the ε-transitions has transition diagram depicted in Fig. 16.2 The resulting automaton after replacing of ε-transitions is

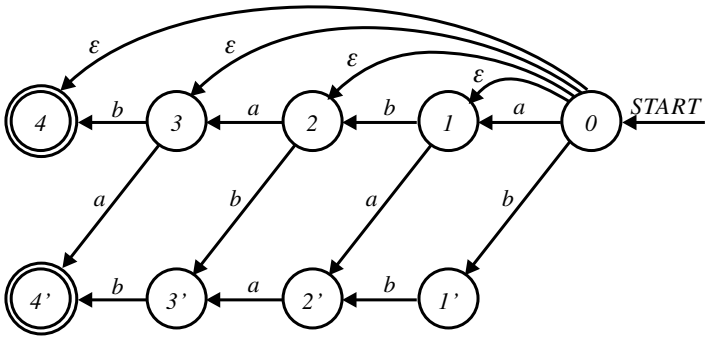

Figure 16.2: Transition diagram of suffix automaton M_2 with ε-transitions accepting language $HSuf_1(abab)$ from Exercise 16.1

automaton M_3 having transition diagram depicted in Fig. 16.3.

The last step is the construction of the equivalent deterministic suffix automaton M_4. Transition tables of the nondeterministic suffix automaton M_3 and an equivalent deterministic suffix automaton are shown in Table 16.1. Transition diagram of deterministic suffix

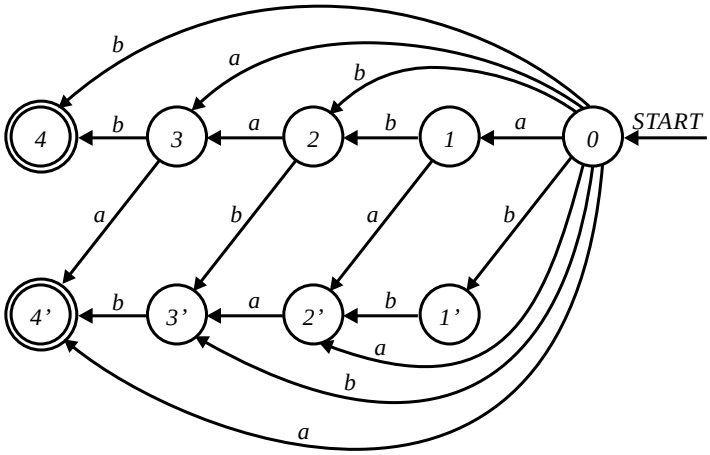

Figure 16.3: Transition diagram of nondeterministic suffix automaton M_3 accepting all suffixes of the language $HSuf_1(abab)$ from Exercise 16.1

	a	b
0	$132'4'$	$241'3'$
1	$2'$	2
2	3	$3'$
3	$4'$	4
4		
$1'$		$2'$
$2'$	$3'$	
$3'$		$4'$
$4'$		

	a	b
0	$132'4'$	$241'3'$
$132'4'$	$2'3'4'$	24
$241'3'$	3	$2'3'4'$
$2'3'4'$	$3'$	$4'$
24	3	$3'$
3	$4'$	4
4		
$3'$		$4'$
$4'$		

Table 16.1: Transition tables of finite automaton M_3 and automaton M_4 from Exercise 16.1

automaton M_4 is depicted in Fig. 16.4.

We use the suffix automaton M_4 for searching of pattern $P = baba$ in text $T = aaabbbabab$. The searching algorithm behave in the same way as for exact searching (see Exercise 14.1). The number of errors in the found pattern can be fixed according to the state reached when the pattern is found. In our case the pattern is found without errors in state 4. The pattern is found with one error in state $4'$. The configuration of searching algorithm is the same as for exact searching (see Exercise 16.1). For text

$T =$

1	2	3	4	5	6	7	8	9	10
a	a	a	b	b	b	a	b	a	b

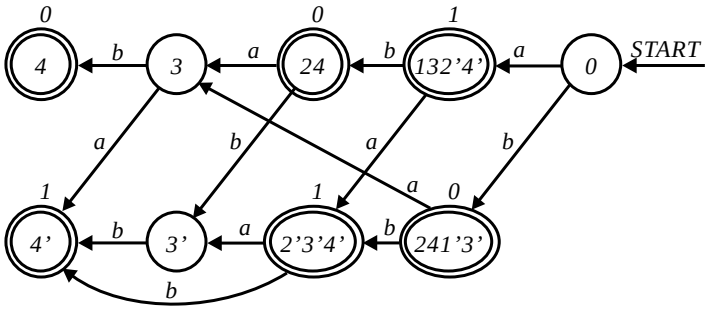

Figure 16.4: Transition diagram of deterministic suffix automaton M_4 accepting all suffixes of the language $HSuf_1(abab)$ from Exercise 16.1

performs the searching algorithm this sequence of transitions:

$$
\begin{aligned}
(0,4,0,0,4) &\vdash^b (241'3',3,1,1) \\
&\vdash^a (3,2,2,1) \qquad \text{shift by 3} \\
&\vdash\ (0,7,0,0) \\
&\vdash^a (132'4',6,1,1) \\
&\vdash^b (24,5,2,2) \\
&\vdash^b (3',4,3,2) \\
&\vdash^b (4',3,4,2) \qquad \text{pattern found with one error, shift by 2} \\
&\vdash\ (0,9,0,0) \\
&\vdash^a (132'4',8,1,1) \\
&\vdash^b (24,7,2,2) \\
&\vdash^a (3,6,3,2) \\
&\vdash^b (4,5,4,2) \qquad \text{pattern found without errors, shift by 2} \\
&\vdash\ (0,11,0,0) \qquad \text{finish}
\end{aligned}
$$

For optimisation it is needed to define approximate period and approximate border. □