

Formal Languages And Compilers  
*Exercises*

Christian Rossi

Academic Year 2023-2024

## **Abstract**

The lectures are about those topics:

- Definition of language, theory of formal languages, language operations, regular expressions, regular languages, finite deterministic and non-deterministic automata, BMC and Berry-Sethi algorithms, properties of the families of regular languages, nested lists and regular languages.
- Context-free grammars, context-free languages, syntax trees, grammar ambiguity, grammars of regular languages, properties of the families of context-free languages, main syntactic structures and limitations of the context-free languages.
- Analysis and recognition (parsing) of phrases, parsing algorithms and automata, push down automata, deterministic languages, bottom-up and recursive top-down syntactic analysis, complexity of recognition.
- Syntax-driven translation, direct and inverse translation, syntactic translation schemata, transducer automata, and syntactic analysis and translation. Definition of semantics and semantic properties. Static flow analysis of programs. Semantic translation driven by syntax, semantic functions and attribute grammars, one-pass and multiple-pass computation of the attributes.

The laboratory sessions are about those topics:

- Modellization of the lexicon and the syntax of a simple programming language (C-like).
- Design of a compiler for translation into an intermediate executable machine language (for a register-based processor).
- Use of the automated programming tools Flex and Bison for the construction of syntax-driven lexical and syntactic analyzers and translators.

# Contents

<b>1</b>	<b>Exercise session I</b>	<b>2</b>
1.1	Regular expression's equality . . . . .	2
1.2	Regular expression's ambiguity . . . . .	3
1.3	Operations on languages . . . . .	4
<b>2</b>	<b>Exercise session II</b>	<b>5</b>
2.1	Regular expressions and FSA . . . . .	5
2.2	Regular expressions and FSA . . . . .	8

# Chapter 1

## Exercise session I

### 1.1 Regular expression's equality

Given two regular expression:

$$R_1 = ((2b)^*c)^* \quad R_2 = (c^*(2b)^*)^*$$

Check if they are equal. If they are not give a counterexample.

#### Solution

It is possible to see that  $R_1$  and  $R_2$  are not equivalent because the character  $c$  is in a different position and can be found multiple times in  $R_2$ , while in  $R_1$  is found exactly one time. An example can be  $ab$ . This string is included in the second language, but not in the first one.

## 1.2 Regular expression's ambiguity

Given the regular expression:

$$R_1 = (a|\varepsilon)^+(ba|bab)^*$$

check if it is ambiguous.

### Solution

First, we enumerate all the characters in the regular expression, obtaining:

$$R_1 = (a_1|\varepsilon)^+(b_2a_3|b_4a_5b_6)^*$$

and now we try to come up with an ambiguous string to prove that the regular expression is ambiguous. A regular expression is considered ambiguous if there is a string which can be matched by more than one way from the regular expression. For instance, we can have the string  $a_1$  can be generated multiple times selecting the  $\varepsilon$   $n - 1$  times. This proves that the regular expression is ambiguous.

## 1.3 Operations on languages

Given two regular expressions:

$$R_1 = a((b|bb)a)^+ \quad R_2 = (ab)^*ba$$

Define the quotient language  $L = R_1 - R_2$ .

1. Write the three shortest strings of the language  $L$ .
2. Write a regular expression that defines the language.

### Solution

First, we enumerate all the characters in the regular expressions, obtaining:

$$R_1 = a_1((b_2|b_3b_4)a_5)^+$$

$$R_2 = (a_1b_2)^*b_3a_4$$

The  $a_1$  is surely a prefix for every string in the language, and all the strings have  $a_5$  as a suffix. The shortest strings of this language are: *aba*, *ababa* and *abababa*.

For the second regular expression we have that every string generated starts with *ab* and have a single *ba* as a suffix. The shortest strings of this language are: *ba*, *abba* and *abababa*.

We can see that all the strings that have the suffix *aba* or have at least two *bb* are certainly in  $L$ .

1. Now, we can see that the three shortest strings are: *aba*, *ababa*, and *abbaba*.
2. The regular expression is:

$$L = \{(a(b|bb))^*aba\} \cup \{(a(b|bb))^*abba(a(b|bb))^+abba\}$$

# Chapter 2

## Exercise session II

### 2.1 Regular expressions and FSA

Consider the regular expression  $R$  below, over the three-letter alphabet  $\Sigma = \{a, b, c\}$

$$R = a(b|c^+a)^*$$

Answer the following questions:

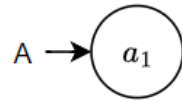
1. Write all the strings  $x$  of the language of  $R$  that have a length less than or equal to four, i.e.,  $x \in L(R)$  with  $|x| \leq 4$ , in lexicographic order (with  $a < b < c$ ).
2. By means of the Berry-Sethi method, find a deterministic automaton  $A$  equivalent to the regular expression  $R$ .
3. Is the deterministic automaton  $A$  found before minimal? Justify your answer.
4. By means of the Brzozowski method (node elimination), starting from the automaton  $A$  found before, obtain a regular expression  $R'$  equivalent to  $A$ .
5. Is the language  $L(R)$  locally testable? Formally prove your answer.

### Solution

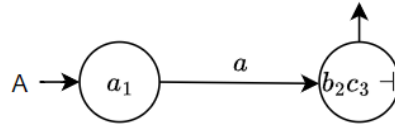
1. The possible strings are:  $a, ab, abb, aca, abba, abca, acab, acca$ .
2. First, we enumerate the symbols  $R_{\#} = a_1(b_2|c_3^+a_4)^*$ . We construct the following support table:

Initials	$a_1$
Terminals	Followers
$a_1$	$b_2c_3 \dashv$
$b_2$	$b_2c_3 \dashv$
$c_3$	$c_3a_4$
$a_4$	$b_2c_3 \dashv$

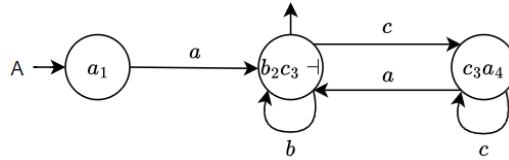
We start with the initial, and we have:



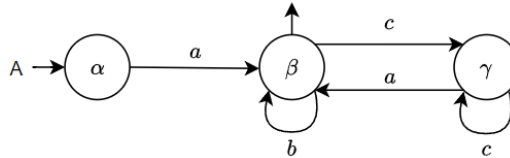
Now we create the state reachable from  $a_1$ , and we obtain:



After doing this steps for all the states we obtain the following automaton:



3. We can reduce an automaton if we can reduce the number of states. We have to use some criterions to do the check on automaton  $A$ . To simplify the automaton we start by renaming the states:

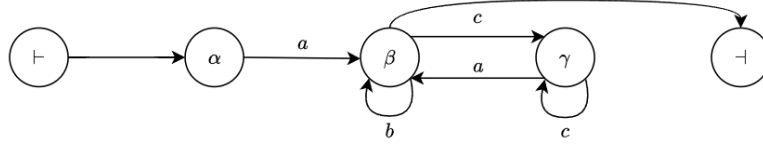


We can see that  $\alpha$  cannot be merged with  $\beta$  because one is not final and the other one is final. Same reasoning holds for  $\beta$  and  $\gamma$ . States



$\alpha$  and  $\gamma$  cannot be merged because they have different transitions. So, the three states are distinguishable, and the automaton is the minimal.

4. We start by creating a virtual initial node and final node connected to the automata:



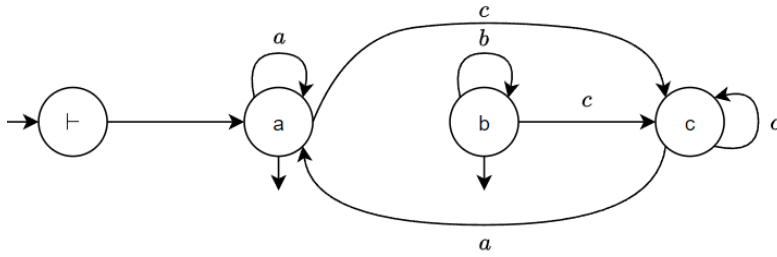
Now we can remove the states one by one until we reach the final state directly from the initial one. So, we remove  $\gamma$  with the loop  $c^+a$ . We have two cycles on  $\beta$  that can be substituted by the expression  $(b|c^+a)^*$ . We have only  $\alpha$  that can be easily removed, and we finally have that:

$$R' = a(b|c^+a)^*$$

5. We have the following sets:

- Initials:  $\{a\}$
- Finals:  $\{a, b\}$
- Digrams:  $\{aa, ac, bb, bc, ca, cc\}$

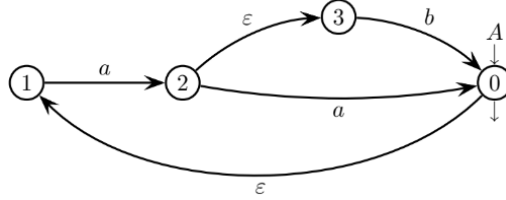
Using these sets we can build a particular automaton  $A'$ , that have the initial set connected to the set of initials, the final states are the ones in the finals set, and the transitions are the one belonging to the digrams set.



The automaton is local if it recognizes the initial language and if the edge reached by an arc has the same name of the transition. In this case we constructed the nodes such that the second property is satisfied, so we simply need to check the first property. We can not that the states  $a$  and  $b$  are not distinguishable, so we can reduce the automaton to the  $A$  one. So, the language is locally testable.

## 2.2 Regular expressions and FSA

Take a two-letter alphabet  $\Sigma = \{a, b\}$ . Consider the nondeterministic automaton  $A$  over  $\Sigma$  below, which has spontaneous transitions ( $\varepsilon$ -transitions):



And consider also the regular expression  $R$  over  $\Sigma$  below:

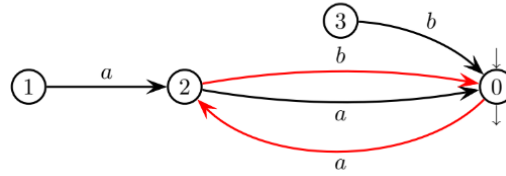
$$R = (aa|ab|ba)^*$$

Answer the following questions:

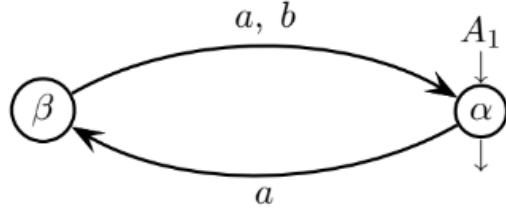
1. Find all the valid strings of language  $L(A)$  that have a length less or equal than four. Do the same for language  $L(R)$ , too.
2. By using the back propagation method, eliminate the spontaneous transitions of automaton  $A$  and obtain an equivalent deterministic automaton  $A_1$  without spontaneous transitions. Clean and minimize automaton  $A_1$ , if necessary. Test automaton  $A_1$  with the valid short strings found at point one for automaton  $A$ .
3. By using the Berry-Sethi method, obtain a deterministic automaton  $A_2$  equivalent to the regular expression  $R$ . Minimize automaton  $A_2$ , if necessary. Test automaton  $A_2$  with the valid short strings found at point one for expression  $R$ .
4. Say if language  $L(R)$  is local or not, and shortly justify your answer.
5. By using a systematic method of your choice, obtain a deterministic automaton  $A_3$  for the difference language  $L_D = L(R) - L(A)$ , i.e.,  $L(R) \cap \overline{L(A)}$ , which contains the strings generated by  $R$  that are not recognized by  $A$ . Then test automaton  $A_3$ : by examining (only) the strings of length 4 found at point one, find in alphabetical order the first two strings  $x$  and  $y$  such that  $x \in L_D$  and  $y \notin L_D$ , and show that  $x \in L(A_3)$  and  $y \notin L(A_3)$ .

## Solution

1. The strings of the language  $L(A)$  are  $\varepsilon$ ,  $aa$ ,  $ab$ ,  $aaaa$ ,  $aaab$ ,  $abaa$ , and  $abab$ . The strings of the language  $L(R)$  are:  $\varepsilon$ ,  $aa$ ,  $ab$ ,  $ba$ ,  $aaaa$ ,  $aaab$ ,  $aaba$ ,  $abaa$ ,  $abab$ ,  $abba$ ,  $baaa$ ,  $baab$ , and  $baba$ .
2. The idea of the back propagation method is to start from the arriving state of an  $\varepsilon$  transition, eliminate the spontaneous transition and back propagate the transition exiting from the state. In this case we obtain the following automaton:



After that we can remove the useless states and rename the remaining ones.

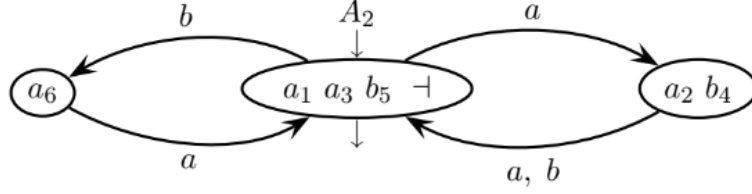


The states are un-distinguishable, so the automaton is the minimal one, and it is the same as the initial one.

3. First, we enumerate the symbols  $R_{\#} = (a_1a_2|a_3b_4|b_5a_6)^* \dashv$ . We construct the following support table:

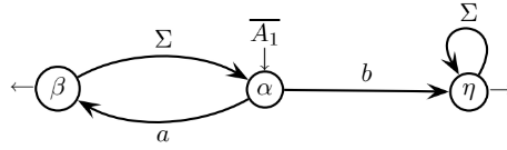
Initials	$a_1a_3a_5 \dashv$
Terminals	Followers
$a_1$	$a_2$
$a_2$	$a_1a_3a_5 \dashv$
$a_3$	$b_4$
$b_4$	$a_1a_3a_5 \dashv$
$b_5$	$a_6$
$a_6$	$a_1a_3a_5 \dashv$

From which we construct the following automaton:

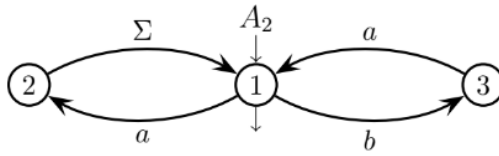


That is already minimal.

4. The language is not local. The minimal automaton of a local language is necessarily local. Automaton  $A_2$  is minimal, but it is not local. Thus, language  $L(R)$  is not local.
5. The language  $L_D = L(R) - L(A) = L(R) \cup \overline{L(A)}$ . We need to find the complement of the automaton  $A$ , by switching final and non-final states with the complement, and by adding an error state  $\eta$ . In this case we obtain:



We rewrite the automaton  $A_2$



We can write all the states and then try to connect with both transitions, and so we obtain:

