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

Chapter 1

Exercise session I

1.1 Regular expression's equality

Given two regular expression:

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

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 ε $n-1$ times. This proves that the regular expression is ambiguous.

1.3 Operations on languages

Given two regular expressions:

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

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 $\Sigma = \{a, b, c\}$

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

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.

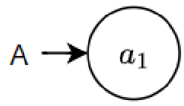
Solution

1. The possible strings are: a, ab, abb, aca, $abba$, $abca$, $acab$, $acca$.

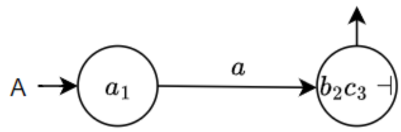
2. First, we enumerate the symbols $R_\# = a_1\left(b_2|c_3^+a_4\right)^* \dashv$. We construct the following support table:

Initials	a_1
Terminals	**Followers**
a_1	$b_2 c_3 \dashv$
b_2	$b_2 c_3 \dashv$
c_3	$c_3 a_4$
a_4	$b_2 c_3 \dashv$

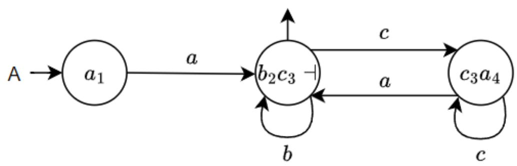
We start wit the initial, and we have:



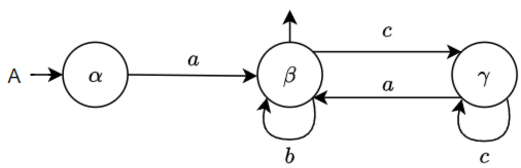
Now we create the state reachable from a_1, and we obtain:



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



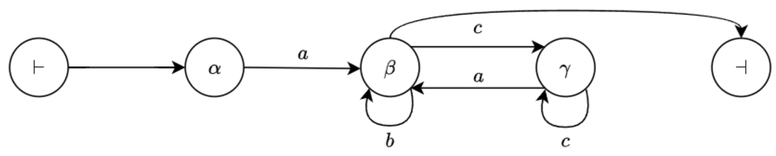
3. We can reduce an automaton if we can reduce the number of states. We have to use some criterions to do the check on automaton A. To simplify the automaton we start by renaming the states:



We can see that α cannot be merged with β because one is not final and the other one is final. Same reasoning holds for β and γ. States

α and γ 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:



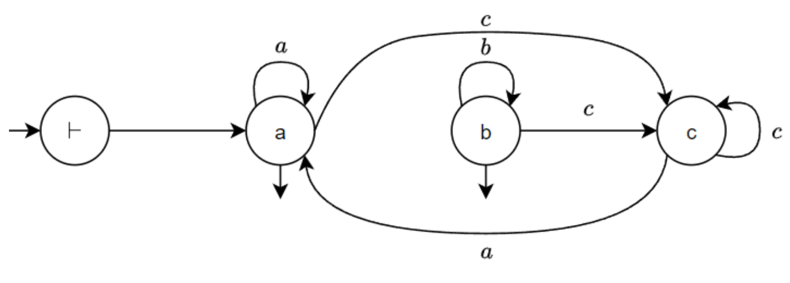
Now we can remove the states one by one until we reach the final state directly from the initial one. So, we remove γ with the loop c^+a. We have two cycles on β that can be substituted by the expression $(b|c^+a)^*$. We have only α that can be easily removed, and we finally have that:

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

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 Σ below, which has spontaneous transitions (ε-transitions):