1DV517 - Assignment 1

April 2, 2021

- Your solutions should include (i) a .pdf file in addition to (ii) the well-commented code, its executable files, and instructions for running the code.
- Submit each file separately in uncompressed format.
- The deadline for submissions is 16 April 2021.
- **Only typed solutions will be accepted.** You can however submit scans or images of the automata.
- *Reports that do not comply with the aforementioned requirements will not be considered.*

Excercise 1. Consider the alphabet $\{a, b, c\}$.

- Write a regular expression that matches all strings where if the number of a's is even in the string, then b's are not followed by c.
- Write a regular expression that matches all the strings whose last symbol has appeared before in the string at least once, e.g., it accepts bcabacba, because a has appeared twice before its last occurrence.
- Design a deterministic finite state automaton **with a minimum number of states** that accepts the language of

$$\epsilon + (a + b)(b + a + \epsilon)^*(\epsilon + a + b) + (c + a)^*(b + \epsilon)$$

Excercise 2. Let s be a string that is defined over the alphabet Σ. The projection of s on Σ' is obtained by removing all symbols that are not in Σ'. For instance, consider the string $s = abbbbabca$ defined over $\Sigma = \{a, b, c\}$. The projection of s over $\{a, c\}$ is $s = aaca$.

Let L be a regular language defined over Σ and L' be a language whose strings are the projection of L's strings over Σ'. Is L' regular ? Prove/justify your answer.

Excercise 3. For this exercise, show the steps.

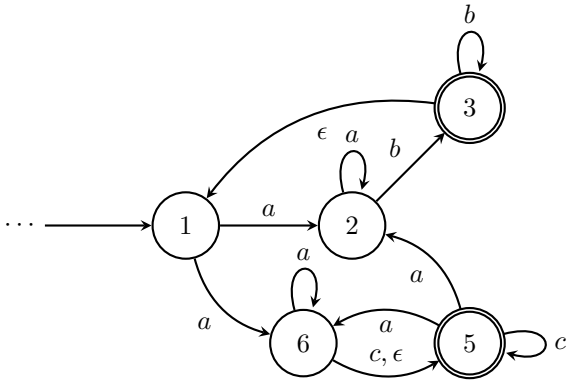

Figure 1: NFA 1

I. Show that the automaton in Figure 1 accepts the language $L = (a^+b^+|a^+c^*)^+$.

II. Give the language of the automaton in Figure 2.

Excercise 4. For each of the following languages defined over $\Sigma = \{a, b, c\}$. Prove/show its regularity or irregularity.

1. $\{wtw \mid t, w \in \Sigma^*\}$
2. All strings of form $(a+b)^n(c+a^j)^m$ where n is odd, m is even and $j \geq 0$.

Excercise 5.

Practical Problem Implementing a simple lexical analyzer in a Java-like language.

Consider a C-like language in which a program consists of a few functions. This language

- supports four types: `int`, `long`, `char` and arrays of these types (there is no objects or classes), a function can also have a `void` return type.
- supports assignments, method calls, conditionals (`if-then-else`), `for` loops, and `while` loops.

You can assume that parentheses following anything other than `if, while` and `for` identifiers correspond to a function definition at the top level or a function call inside a function block. For all blocks match greedily in your implementation. Write a program that takes a code written in this language as input, and uses regular expressions to identify and report the followings:

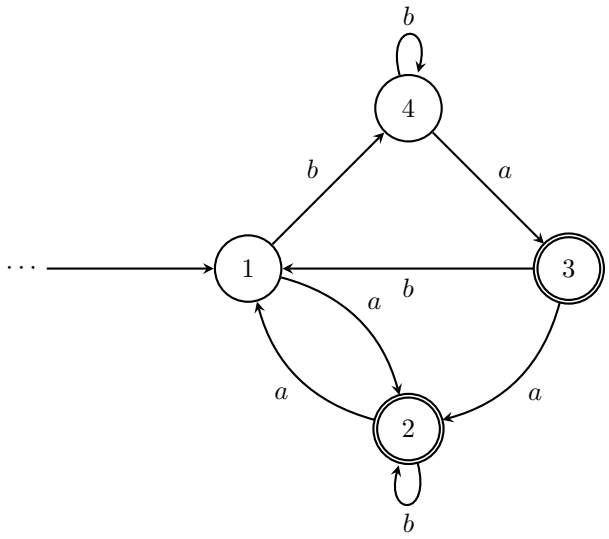

Figure 2: NFA 2

- The list of function definitions including their names, their arguments and types.
- For each function: its body including '{' and '}', the names and types of local variables used in the functions, and the names of possible function calls inside the function.
- For `if` and `while` structures, the condition upon which the structure is entered, and their bodies (all statements inside the structure's block),
- For `for` statements, report the initialization. condition, iteration and its body.

A sample input and output is provided in the supplementary materials. Your program output should follow the same pattern.