TD 3
Finite Automata

Version du September 20, 2020

Exercice 1 – Matching lists

Let U be a finite alphabet. We define a *list* of U to mean a sequence beginning by the symbol '(', ending by the symbol ')', and containing elements of U separated by the symbol ':'. As an example: $(1:2:3:2:1)$ is a list of elements of $U = \{1, 2, 3\}$. In the remainder of this exercise, we consider that a list always contains at least one element: *there are no empty lists*.

1. Is it possible, by using a finite automaton, to recognize the set of lists of elements of $U = \{1, 2, 3\}$? If you think so, give a graphical representation of an automaton recognizing this language, indicate the alphabet, the initial state, and final states. Simpler automatons are preferred.
2. In the case where U is totally ordered, we are interested in increasing lists: every element of the list should be greater or equal to its predecessor. Is it possible to recognize such lists with a finite automaton? Justify a positive answer by representing such an automaton in the case where $U = \{1, 2\}$.
3. A perverse computer scientist is interested in the case where U contains the special characters '(', ')', and ':'. What happens in this case? When these three symbols are added to U, should we modify the automaton from question 1? If yes, how?
4. The notion of list is recursively extended in order to include lists of lists, lists of lists of lists, ... such as $((1:3):3:(2:1):((1:2)))$. Is it possible to recognize these lists with a finite automaton? Justify a positive answer by representing such an automaton in the case where $U = \{1, 2\}$.

Exercice 2 – Beverage vending machine

We model a beverage vending machine by a finite automaton. The alphabet is $\Sigma = \{d, v, c\}$ corresponding to 10, 20, and 50 cent coins accepted by the machine. A drink costs 50 cents.

1. Build the *deterministic* automaton, A, recognizing every sequences of coins with a total sum of 50 cent. You can begin by proposing non-deterministic automatons recognizing all these sequences.
2. This vending machine is primitive: it does not give back change. In order to make the machine capable to do so, we need to complete the automaton. It should be able to "read" (gather) coins, but also to "write" (return) them. We are talking here of *finite-state transducer*. Concretely, every transition of our new machine will be labeled by an input/output pair a/b. Taking a labeled transition a/b is interpreted by: *read a, and write b*.

   Starting from the automaton from the last question, build a finite-state transducer, T, that simulate a vending machine that can give back change: for all input sequences where the total is greater than 50 cent, your transducer should produce an output sequence where the sum is the difference between the input amount and 50 cent.

   Illustrate its operation on the input sequence: *dvcv*.

Exercice 3 – Regular expression translation

In this exercise, we suppose that $\Sigma = \{a, b, c\}$.

For each of the following regular expressions, use the Thompson algorithm to build the finite non-deterministic automaton with ϵ-transitions, then apply the method presented in the course to delete the ϵ-transitions.

1. $c(ab + c)$
2. $(a + (cc)^\star)(b + c)$
3. $((ab + \varepsilon)^\star c)^\star$