TD 4
Pumping lemma and determinization

Version du September 20, 2020

Exercice 1 – Lists of lists of lists . . .

We take back a question from the last TD:

> 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?

1. Use the pumping lemma for the regular languages in order to demonstrate that the language $L_p = \{(^n 1)^n \mid n \geq 0\}$ is not regular.

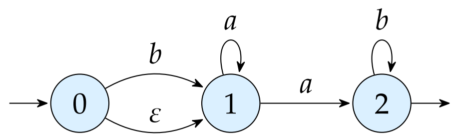
   Recall the **Pumping Lemma:** For any rational language L, there exists an integer k, called the *pumping length*, for which for all $x \in L$, with $|x| \geq k$, there exists a factorization $x = u \cdot v \cdot w$ (with $u, v, w \in \Sigma^\star$) such that:

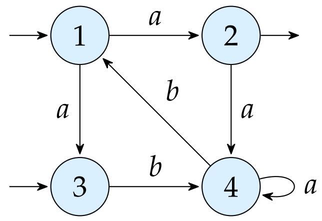
   1. $|uv| \leq k$
   2. $|v| \geq 1$
   3. for all $i \geq 0$ it holds that $u \cdot v^i \cdot w \in L$.

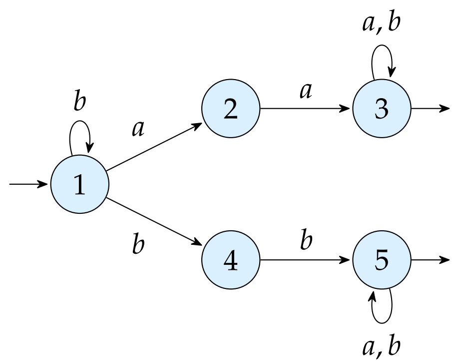
2. Deduce that it is not possible to recognize the language L_l, composed of lists, lists of lists, lists of lists of lists, . . . with a finite automaton.

Exercice 2

We suppose $\Sigma = \{a, b\}$. Using the method presented in the course, build a deterministic automaton equivalent to each of the following automatons:







Exercice 3 – Pattern search

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

1. Let $m = abab$ be a word, and L the language of words that have m as suffix. L contains the words of the form $v = um$, with $u \in \Sigma^\star$; for example, the words *aaabab* and *babab*. On the other hand, *caabc* does not belong to L.

   Prove that L is rational. Propose a finite *non-deterministic* automaton A_n for L. You should justify the building of the automaton.

2. Using the method presented in the course, transform A_n in to an equivalent complete deterministic automaton A_d. You should explain the different steps of the progress of the algorithm.

3. Why is it obvious that A_d is complete and trim?

4. We modify the alphabet with $\Sigma = \{a, b, c, d, e\}$ for this question only. How should we reverberate this modification on A_d ?

5. We consider the following algorithm:

   // $u = u_1 \dots u_n$ is the word in which we are looking for.
   // $A_d = (\Sigma, Q, \{q_0\}, F, \delta)$ is a deterministic automaton for L.
   $q \leftarrow q_0$
   $i \leftarrow 1$
   $c \leftarrow 0$
   while $(i \leq n)$ **do**
   $\quad q \leftarrow \delta(q, u_i)$
   $\quad i \leftarrow i + 1$
   $\quad$**if** $(q \in F)$ **then** $c \leftarrow c + 1$ **end if**
   end while
   return c

   a. Illustrate the behaviour of this algorithm when $u = bcababcabbababac$, and the automaton is A_d the one from question 2. You should give for each step of the main loop the value of c.

   b. What is computed by this algorithm? Justify your affirmation.

   c. What is the complexity of this algorithm?

   d. What is the value of c at the end of the execution of the algorithm for $u = cabbabababababc$? What do you observe?

   e. How should we modify the automaton A_d in order to count the maximal number of disjoint occurrences of the pattern in the input string. For example, the response should be 2 for the last example.