université PARIS-SACLAY | FACULTÉ DES SCIENCES D'ORSAY

semaine du 6 février 2023

Automates et Applications
M1–MPRI Orsay

TP nº 5
Automata in practice

Motivation

The goal of this practical sheet is to prepare for the (practical part of the) home assignments and revisit some aspects of automata theory. It can be done in any programming language that supports strings and a notion of dictionary or hash-table. Guidance and answers are given in Python and OCaml, but an easily be translated in other languages.

If you do not have a programming environment readily available on your machine, you can use :

`https://jupyterhub.ijclab.in2p3.fr`

With your `universite-paris-saclay` credentials and choose Start My Server or VSCode.

1 Automaton data-structure

1. In your programming language give a data structure for a non-deterministic word automaton. Be very precise. What choices can be made and what are the trade-offs (efficiency, space requirements, kinds of texts that can be processed).

2 Naive text recognition

1. Write a naïve accept function that takes an automaton and a string and returns a Boolean indicating whether the string is accepted.
2. Give an automaton and a word such that the run function exhibits and exponential behavior.

3 On the fly algorithms

1. Write a function `next` that takes as input an automaton, a set of states S, a letter a and returns the set of all states that can be reached from a state in S reading a.
2. What is the complexity of this function?
3. Use `next` to write a function test if a string is recognized by an automaton.
4. Retry the experiment of part one.
5. Using `next` write two functions that take a string and a list of automata and test whether the string is in the union/intersection.

4 Memoization

1. The function `next` written in the previous part is sometimes doing te same computations over and over. Improve its efficiency in practice by making it take a cache as an extra argument.
2. Retry the experiment of part 1.

5 ϵ-moves

1. explain how to add ϵ moves to your automata data structure
2. extend the naive run and efficient run to support ϵ-moves
3. give an abstract syntax for regexps and implement Thompson's procedure
4. evaluate the efficiency of regexp matching with naive, efficient and efficient + cache procedure.