Formal Methods
Fall 2017

S06

Professor : Ultes-Nitsche Ulrich
Assistant : Christophe Stammet

Submitted by Sylvain Julmy

Exercise 1

Note : we use a tree representation for all the posibilities of runs for a given word. The *nth* level of a tree corespond to the *nth* character of a word.

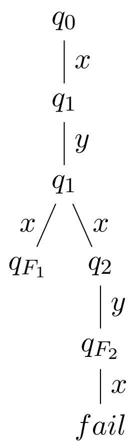
(w_1)

$$\begin{tikzcd}
 & q_0 \arrow[d, dash, "x"] & \\
 & q_1 \arrow[d, dash, "y"] & \\
 & q_1 \arrow[d, dash, "y"] & \\
 & q_1 \arrow[d, dash, "y"] & \\
 & q_1 \arrow[dl, dash, "x"'] \arrow[dr, dash, "x"] & \\
q_{F_1} & & q_2 \arrow[d, dash, "y"] \\
 & & q_{F_2}
\end{tikzcd}$$

Possible runs with tapes :

- $(q_0, q_1, q_1, q_1, q_1, q_{F_1})$, $tape = (x, y, y, y, x, B, B, \dots)$
- $(q_0, q_1, q_1, q_1, q_1, q_2, q_{F_2})$, $tape = (x, y, y, B, B, B, \dots)$

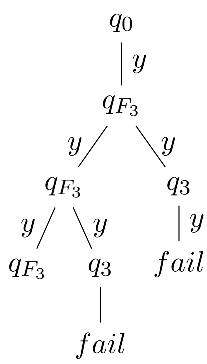
(w_2)



Possible runs with tapes :

- (q_0, q_1, q_1, q_{F_1}) , $tape = (x, y, x, B, B, \dots)$
- $(q_0, q_1, q_1, q_2, q_{F_2})$, $tape = (x, y, B, B, B, \dots)$, end in an non-accepting state and no possible transition.

(w_3)



Possible runs with tapes :

- $(q_0, q_{F_3}, q_{F_3}, q_{F_3})$, $tape = (x, x, x, B, B, \dots)$
- $(q_0, q_{F_3}, q_{F_3}, q_3)$, $tape = (x, x, B, B, B, \dots)$, end in an non-accepting state.
- (q_0, q_{F_3}, q_3) , $tape = (x, B, B, B, \dots)$, end in an non-accepting state and no possible transition.

We can also assume that the Turing Machine immediatly accept the word if an accepting state is reached. So, for the word $w_3 = yyy$, the only possible run would be (q_1, q_{F_3}) with the tape $tape = (x, \dots)$.

(w_4)

We can assume that the Turing Machine immediatly accept the word if an accepting state is reached. So, for the word $w_4 = yx$, the only possible run would be (q_1, q_{F_3}) with the tape $tape = (x, \dots)$.
If the Turing machine don't stop when reaching q_{F_3}, then the computation tree would be :

$$
\begin{tikzcd}
q_0 \arrow[d, dash, "y"] \\
q_{F_3} \arrow[d, dash, "x"] \\
fail
\end{tikzcd}
$$

and the Turing machine will be non-accepting.

(w_5)

$$
\begin{tikzcd}
& q_0 \arrow[d, dash, "z"] & \\
& q_4 \arrow[dl, dash, "x"'] \arrow[dr, dash, "x"] & \\
q_4 \arrow[d, dash, "y"'] & & q_5 \arrow[d, dash, "y"] \\
q_{F_4} & & q_5 \arrow[d, dash, "z"] \\
& & q_{F_4}
\end{tikzcd}
$$

Possible runs with tapes :

- (q_0, q_4, q_4, q_{F_4}) , $tape = (B, x, B, B, B, \dots)$
- (q_0, q_5, q_5, q_{F_4}) , $tape = (B, x, y, B, B, \dots)$

Exercise 2

(1)

If a problem is in $\mathcal{P}$, it means that the problem could be solve in a polynomial time. In other word, the problem is solved by a deterministic Turing machine in a polynomial complexity.

(2)

If a probkem is in $\mathcal{NP}$, it means that the problem needs a super-polynomial time to be solve. In other word, the problem is solved by a non-deterministic Turing Machine in a polynomial complexity.

(3)

The $\mathcal{NP}$-complete class is the problem from $\mathcal{NP}$ for which we don't know if it exist a polynomial algorithm on a deterministic Turing Machine which solve them. This class also holds an interresing property : if we can solve 1 problem on the $\mathcal{NP}$-complete class in a polynomial time on a deterministic Turing machine, then we can solve any $\mathcal{NP}$-complete problem in a polynomial time on a deterministic Turing machine. Example of $\mathcal{NP}$-complete problems :

- SAT
- Hamiltonian path
- Knapsack
- ...

(4)

$\mathcal{P} \subseteq \mathcal{NP}$

Correct, the complexity class $\mathcal{P}$ is contained in $\mathcal{NP}$, because a deterministic Turing Machine is just a special case of a non-deterministic Turing Machine.

$\mathcal{NP} \neq \emptyset$

Wrong, $\mathcal{NP}$ contains the problems of $\mathcal{P}$ and from the $\mathcal{NP}$-complete classes.

$\mathcal{NP} \subseteq \mathcal{NPC}$

If we prove that $\mathcal{P} = \mathcal{NP}$, then we would have $\mathcal{P} = \mathcal{NP} = \mathcal{NPC}$ so $\mathcal{NP} \subseteq$ would be true.
Otherwise it is false, because we have $\mathcal{P} \subseteq \mathcal{NP}$, $\mathcal{NPC} \subseteq \mathcal{NP}$ and $\mathcal{P} \neq \mathcal{NPC}$ so $\mathcal{NP} \subseteq \mathcal{NPC}$ is false.

$\mathcal{P} \cap \mathcal{NPC} \neq \emptyset$

It is the $\mathcal{P} = \mathcal{NP}$ problem, if $\mathcal{P} = \mathcal{NP}$, so $\mathcal{P} \cap \mathcal{NPC} = \mathcal{P} = \mathcal{NPC} = \mathcal{NP}$. Else, if $\mathcal{P} \neq \mathcal{NP}$, then $\mathcal{P} \cap \mathcal{NPC} = \emptyset$. Or maybe, it is not a decidable problem to prove that $\mathcal{P} = \mathcal{NP}$, but we don't know yet.

(5)

In order to prove that $\mathcal{P} = \mathcal{NP}$, we could find an algorithm in a polynomial time for a $\mathcal{NP}$-complete problem. For example, finding an algorithm in $O(n^{O(1)})$ for the SAT problem is enough to prove $\mathcal{P} = \mathcal{NP}$.