Lecturer : Michel Abdalla TD-man : Sébastien Samain Final exam : 23/01/2021
Project reports: 05/01/2021 (cf Moodle)

General Methodology

- Choose a computational model
- Which problems can be solved?
- Understand the limits of the model

Regular languages

Σ a finite, non empty set (alphabet) (e.g. $\Sigma = \{0, 1\}$)

A **string** is a finite sequence of symbols in Σ. Special case : the **empty string** ε. We note $|\omega|$ the length of the string ω.

We note $\Sigma^k = \{\omega,\ |\omega| \text{ and } \omega \text{is a string over } \Sigma\}$, $\Sigma^* = \bigcup_{k \geq 0} \Sigma^k$, $\Sigma^+ = \bigcup_{k \geq 1} \Sigma^k$.

A language over Σ is a subset of Σ^*.

An automata (cf fig 1). This automata recognizes $1\{0, 1\}^*\{1\}^*$.

In order to define more formally an automata, we need t define what is allowed.

Deterministic finite automata (DFA)

A DFA is a 5-tuple $D = (Q, \Sigma, \delta, q_0, F)$, with Q the finite set of states, Σ the alphabet, δ the transition function, q_0 the starting state and F the set of all accept states. Precisions about δ : it is of the form : $\delta : Q \times \Sigma \to Q$.

There is only one start state, one state per transition, and potentially several final states.

Extended transition function

$\hat{\delta}(q, \varepsilon) = q$, $\hat{\delta}(q, xa) = \delta(\hat{\delta}(q, x), a)$. It accepts strings as an input.

Language

We define the language of an automata $L(D) := \{\omega \in \Sigma^*, \hat{\delta}(q_0, \omega) \in F\}$. It is the set of strings that make the automata end in a final state.

We say that D recognizes L iff $L = L(D)$.

Example

$Q = \{q_0, q_1, q_2, q_3\}$

$F = \{q_2\}$

$\Sigma = \{0,1\}$

δ	0	1
q_0	q_1	q_2
q_1	q_3	q_2
q_2	q_2	q_2

$L = \{\omega | \omega \equiv 0[5]\}$ $xa = 2x + a$ (binary representation) cf fig 2

Regular languages

Def : a language L is regular iff there exists a DFA D such that D recognizes L.

Questions: if L_1 and L_2 are regular,

- Is $L_1 \cup L_2$ regular ?
- Is $\bar{L_1}$ regular ?
- Is $L_1 \cap L_2$ regular ?

Solution: Yes!

- $\bar{L_1}$: $F \leftarrow Q \setminus F$.
- Union and intersection : make the Cartesian product of the two automata to make them run at the same time and tune the accepting states accordingly.

Non deterministic finite automata (NFA)

ex : cf fig 3

Definition: A NFA is a 5-tuple $D = (Q, \Sigma, \delta, q_0, F)$, with Q the finite set of states, Σ the alphabet, δ the transition function, q_0 the starting state and F the set of all accept states. Precisions about δ : it is of the form : $\delta : Q \times \Sigma \to \mathcal{P}(Q)$.

Extended transition function

$\hat{\delta}(q, \varepsilon) = \{q\}$, $\hat{\delta}(q, xa) = \bigcup\limits_{p \in \hat{\delta}(q,x)} \delta(p, a)$. Read the string x => union over all possible results

Language recognized

$L(N) = \{\omega \in \Sigma^* | \hat{\delta}(q_0, \omega) \cap F \neq \emptyset\}$. N recognizes L iff $L(N) = L$.

Equivalence between NFA and DFA

DFA are NFA.

We can build a DFA that recognizes the same language as a given NFA N by making the following :

$N = (Q, \Sigma, \delta_N, q_0, F_N)$ $D = (\mathcal{P}(Q), \Sigma, \delta_D, \{q_0\}, F_D)$ with $F_D = \{P \subset Q, F_N \cap P \neq \emptyset\}$, $\delta_D = \bigcup_{p \in S} \delta_N(p, a)$.

We now just have to check that D is a DFA and that $L(D) = L(N)$, which is trivial given this expression (proof by induction).

NB: this works because Q is finite.

Example:

Let $L := \Sigma^* 1 \Sigma^{n-1}$. We can build a NFA with $n+1$ states that recognizes it. But we cannot build a DFA with less than 2^n states that recognizes L.

Proof: if it has less than 2^n states, two strings $a = a_1 \dots a_N$ and $b = b_1 \dots b_N$ that are different will end up in the same accepting state p (pigeon hole). There exists i such that $a_i \neq b_i$.

- Case $i = 1$: let's suppose that $a_1 = 0$ and $b_1 = 1$. Because of a, $p \in F$ and because of b, $p \notin F$.
- Case $i = 2$: $a_1 = b_1$, at state 1, they are both at the same state p'. We conclude with case $i = 1$ on the substring.
- ...

ε-transitions

$\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$. To make an ε-NFA, we modify the transition function such that they take Σ_ε in input instead of Σ.

We note $ECLOSE(q)$ = the set of states that can be reached from q with ε-transitions. We define $ECLOSE$ on sets by taking the direct image of the set.

We define $\hat{\delta}(q, \varepsilon) = ECLOSE(q)$ and $\hat{\delta}(q, xa) = \bigcup_{r \in \bigcup_{p \in \hat{\delta}(q,x)}} r$

The equivalence is not that much difficult to show with this definition.