

EITQ : Homework

To be handed back on November 27, 9:00

1 Finite automata with ϵ -transitions

In this exercise, you will have to manipulate automata with ϵ -transitions. In nondeterministic automata as in the lecture, each transition triggers the reading of one letter of the word, and a transition can be used only when the corresponding letter must be read in the word. In automata with ϵ -transitions, some transitions are labelled with one letter, and some are with the empty-word ϵ . ϵ -transitions can always be used from the state from which they are outgoing, and do not trigger the reading of a letter of the word.

Definition 1 (Automaton with ϵ -transitions). An **automaton with ϵ -transitions** is a tuple $\mathcal{A} = (Q, \Sigma, I, F, \delta)$ where:

- Q is a finite set, whose elements are called **states** of \mathcal{A} .
- Σ is an alphabet.
- I is a subset of Q , whose elements are called **initial states** of \mathcal{A} .
- F is a subset of Q , whose elements are called **final states** of \mathcal{A} .
- $\delta : Q \times (\Sigma \cup \{\epsilon\}) \rightarrow \mathcal{P}(Q)$ is a partial function, called the **transition function** of \mathcal{A} .

In this definition, the only difference with nondeterministic automata is that the transitions can also be labelled with the empty-word ϵ . The following definition of ϵ -closure and extended transition function describes what "paths" are allowed when reading a word, and are used to define the accepted language of an automaton.

Definition 2 (ϵ -closure of a set of states). Let $\mathcal{A} = (Q, \Sigma, I, F, \delta)$ an automaton with ϵ -transitions, and $q \in Q$. The **ϵ -closure of q** , denoted as $E(q)$, is the set of states that are reachable from q by following ϵ -transitions in the automaton. In other words, $p \in E(q)$ if and only if there exist a sequence of states q_1, \dots, q_k such that:

- $q_1 = q$
- $\forall 1 \leq i \leq k, q_{i+1} \in \delta(q_i, \epsilon)$
- $q_k \in p$

Definition 3 (Extended transition function of an automaton with ϵ -transitions). Let $\mathcal{A} = (Q, \Sigma, I, F, \delta)$ an automaton with ϵ -transitions. The **extended transition function** of \mathcal{A} is the function δ^* defined as follow:

$$\begin{aligned} \forall q \in Q, \quad \delta^*(q, \epsilon) &\stackrel{\text{def}}{=} E(q) \\ \forall q \in Q, \forall l \in \Sigma, \forall w \in \Sigma^* \quad \delta^*(q, lw) &\stackrel{\text{def}}{=} \bigcup_{q' \in \delta(E(q), l)} \delta^*(q', w) \end{aligned}$$

The language of an automaton with ϵ -transitions is the set of words that can reach a final state when read from an initial state.

Definition 4 (Language of an automaton with ϵ -transition). Let $\mathcal{A} = (Q, \Sigma, I, F, \delta)$ an automaton with ϵ -transitions. The **language of \mathcal{A}** , noted $\mathcal{L}(\mathcal{A})$, is defined as follow:

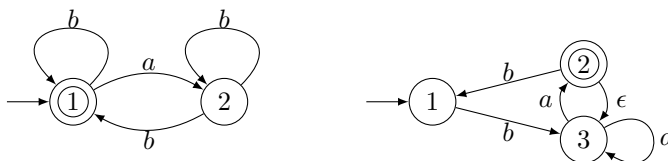
$$\mathcal{L}(\mathcal{A}) \stackrel{\text{def}}{=} \{w \in \Sigma^* \mid \delta^*(I, w) \cap F \neq \emptyset\}$$

where $\delta^*(I, w) \stackrel{\text{def}}{=} \bigcup_{q \in I} \delta^*(q, w)$.

In the following questions, unless explicitly mentioned, we consider nondeterministic finite automata with ϵ -transitions.

Question 1. For the following automata

- Give the language they recognize.
- Say if they are deterministic or not.
- If they are not deterministic, give a deterministic automaton **without ϵ -transitions** that recognize the same language.



Question 2. For any language L , we write L^2 the languages of words that are the concatenation of two words of L , that is:

$$L^2 = \{uv \mid u \in L, v \in L\}$$

- Writing L_1 and L_2 for the languages recognized by the automata of Question 1, build an automata recognizing L_1^2 and an automata recognizing L_2^2 .
- Describe a general procedure to build an automaton recognizing L^2 from an automaton recognizing L .

Question 3. For the following languages, write an automata that recognize them.

- L_3 is the set of words over $\{a, b\}$ such that the parity of the number of a is the same as the parity of the number of b (so both even or both odd).
- L_4 is the set of words over $\{a, b, c\}$ such that the parity of the number of each letter is the same (so all even or all odd).

2 Reduction from SAT to 3-SAT

We consider a finite set of Boolean variables x_1, \dots, x_n . Recall that a SAT formula is obtained through the following grammar:




$$f_1, f_2 ::= x_i \mid \neg f_1 \mid f_1 \wedge f_2 \mid f_1 \vee f_2$$

A **literal** ℓ is a formula of the form x_i or $\neg x_i$, where \neg stands for the negation. A **clause** is a formula of the form $\bigvee_i \ell_i$ where the ℓ_i are literals and \vee stands for the Boolean “or”. A formula in Conjunctive Normal Form (CNF) is a **conjunction** of clauses, of the form $\bigwedge_j c_j$ where the c_j are clauses and \wedge stands for the Boolean “and”. Recall that the problem SAT is: given an arbitrary formula, find if there is an assignment

to true or false of the variables such that the formula evaluates to true. For example, the formula $x_1 \wedge \neg x_1$ is not satisfiable, while $x_1 \wedge \neg x_2$ is satisfiable by taking x_1 true and x_2 false.

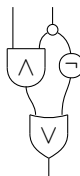
A 3-SAT formula is a formula in CNF where each clause is of size ≤ 3 . The aim of this exercise is to show that 3-SAT, the satisfiability problem on 3-SAT formulas, is **NP-complete**.

A boolean circuit is a graphical representation of a boolean function where we can copy variables

(as many times as required), perform their negation , their conjunction  or their disjunction .

The outputs of gates can be plugged in the inputs of other gates, but no feedback loop is allowed. A boolean circuit has as many input edges as the function has variables, and has one output edge (representing the function itself).

For instance, the following circuit represents the formula $(x_1 \wedge x_2) \vee \neg x_2$ (assuming variables are ordered x_1, x_2):



Question 1. Draw a boolean circuit for the formula $x_1 \wedge x_2 \wedge x_3$ with only binary gates (i.e. gates with exactly two inputs and one output).

Question 2. Draw a boolean circuit for the formula $\varphi := (x_1 \vee x_3) \wedge (\neg(x_2 \vee x_3) \vee \neg x_1)$.

Let's represent any output of the gates \neg , \vee and \wedge in a circuit as a fresh variable (i.e. a variable that was never used before).

Question 3. Suppose x_2 is the variable representing the output of the negation of x_1 . Give a formula in CNF that describes the behaviour of x_2 w.r.t. x_1 (i.e. this formula should be true if and only if $x_2 = \neg x_1$).

Same question when x_3 is the output of the conjunction of x_1 and x_2 ; and when x_3 is the output of the disjunction of x_1 and x_2 .

Question 4. Given a boolean circuit, explain how to build a formula in CNF that describes it (i.e this formula should be satisfiable if and only if the boolean circuit is). This new formula might have more variables than the original one.

Question 5. Apply this construction to the boolean circuit built in Question 2.

Question 6. Using the answers to the previous questions, together with the assumption that SAT is **NP-complete**, show that 3-SAT is **NP-complete**.