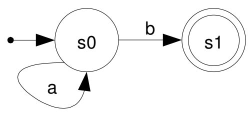
# Advanced Human-Machine Interaction
# TD01 : Finite-state automata
## ASI5 - INSA Rouen
CORRECTION

## Exercise 1




1. Is $A$ deterministic?
2. Are the following words accepted by $A$: $a$, $b$, $a*$, $b*$, $a*b$, $abab$, $a*b*$?
3. What language does this automaton support?

***Correction***

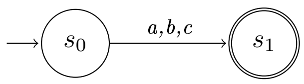
1. *Yes.*
2. *No, yes, no, no, yes, no, no.*
3. *L={a*b}.*

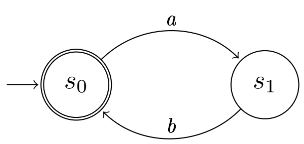
## Exercise 2

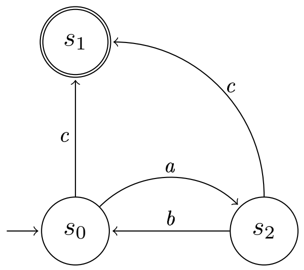
For each of the following languages, build a deterministic automaton which accepts it.

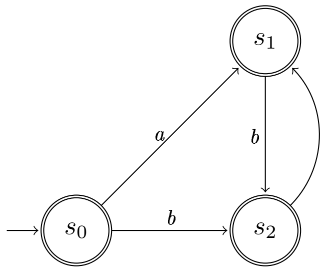
1. $L(A_1) = \{a, b, c\}$
2. $L(A_3) = \{(ab)*\}$
3. $L(A_2) = \{c, ac, (ab)*c, (ab)*ac\}$
4. $L(A_4) = \{(ab)*, (ba)*, (ab)*a, (ba)*b\}$

***Correction***

*1.*



*2.*



*3.*





4.

## Exercise 3

Given the alphabet $E = \{q, r, c\}$ with $q$: "*send a question*", $r$: "*send an answer*" et $c$: "*cancel the question*".

1. Build the automaton supporting the following defined dialogues:
   - A dialogue always begins with a question ($q$).
   - A dialogue either ends with an answer ($r$) or with a cancellation ($c$).
   - If the question is not precise enough, the user sends a request for precision ($q$), that will be followed by a reformulation of the asked question ($r$) or a cancellation of the initial question ($c$).
2. Let $x_1, ..., x_n \in E$ and let $z = x_1...x_nq$ be the execution. Is $z$ supported by this automaton? Explain with you own words. Does that seem plausible to you?

## Exercise 4

Let be an information system whose operation is described by an automaton $A$ based on the following alphabet: $E = \{q, r, b\}$ with $q$: "*reception of a request*", $r$: "*sending an answer*" and $b$: "system error".

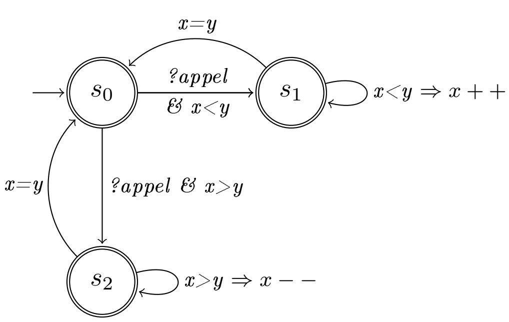
1. Build $A$ given that $\{(qr)*, (qr)*bq*\} \subseteq L(A)$ but $(qr)*b(qr)* \notin L(A)$.
2. Prove that such a system cannot answer 2 questions at the same time.
3. Change $A$ for it to describe a system managing 2 servers.

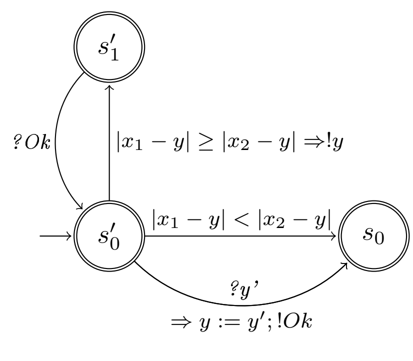
## Exercise 5: Communicating hybrid automata

The aim of this exercise is to model the behaviour of 2 called lifts. No matter which lift is called, the one that will move is always the nearest to the calling floor.

1. A lift can be modelled by 2 variables ($x$: the current lift floor and $y$: the calling one) and 3 states ($s_0$: stopped, $s_1$: going up and $s_2$: going down). *Build such an automaton.*
2. The lifts collaboration can be modelled by a communicating automaton between the processes managing each one of them (*cf.* previous question), using 2 different messages ($y'$: the calling floor, $OK$: reception confirmation). *Build such a communicating automaton.*

***Correction***



# Exercise 6: Timed automata

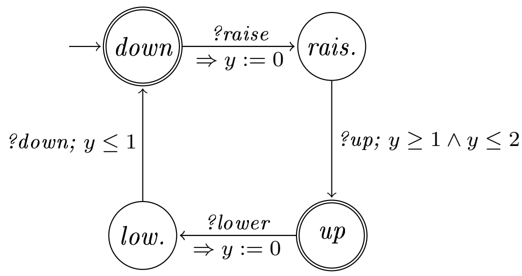
The aim of this exercise is to model a level crossing gate operation.

- When the gate receives the "?*lower*" signal, it reaches the down position in under a minute.
- When the gate receives the "?*raise*" signal, it takes between 1 and 2 minutes to reach the up position.
- If it does not receive any signal, the gate stays indefinitely in its current position.
- The positions are known thanks to sensors placed on the gate.

We consider that the only moment a pedestrian can safely walk over the level crossing is when its gate is totally lowered.

1. Build the automaton describing the gate's behaviour.
2. What is the shortest time for the gate to go up from a safe position for pedestrians, then down to the same exact state?

***Correction***




# Practical session

1. Create 3 classes `automaton`, `state` and `transition` in order to represent an automaton. A `state` is associated to an identifier and a list of `transitions`. A `transition` is described by a label (an alphabet item) along with input and output states.
2. Optional: add to your `automaton` class a constructor allowing to initialise an automaton from a file.
3. Add to `automaton` the required methods to evaluate the current state and perform transitions (an event at a time, or by full execution).
4. Modify the coffee machine example so that the solution is deterministic and code it.
5. Optional: Define a class to describe a hybrid automaton.
6. Optional: Create a class to describe a communicating automaton.
7. Optional: Code the lifts problem thanks to the classes you created earlier.
8. Optional: Simulate the gate operation with a timed automaton. You'll play the role of the gate keeper, sending the 3 possible signals (*lower*, *raise* et *exit*). The imprecise duration will be modelled randomly. Show the gate states and durations.