ARC: Computer Architecture

tanguy.risset@insa-lyon.fr
Lab CITI, INSA de Lyon
Version du March 16, 2023

Tanguy Risset

March 16, 2023

Table of Contents

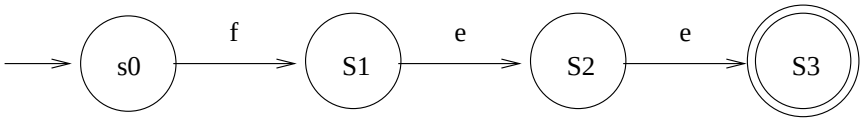
Automata

- Definition (Wikipedia): An automaton is a self-operating machine, or a machine or control mechanism designed to automatically follow a predetermined sequence of operations, or respond to predetermined instructions.
- In computer science:
  - Used in language theory to build compilers
  - Used in any technical domain: to describe predetermined behaviour.
  - Used in computer architecture: to design dedicated circuit.
  - A computer is a specific automaton.

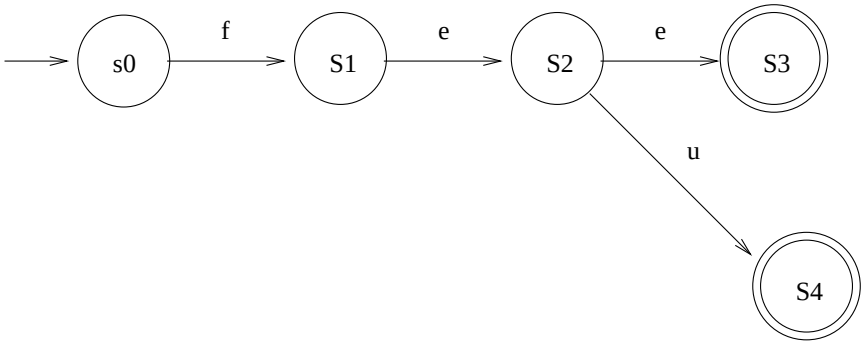
Table of Contents

Langage régulier et automate

- Formellement, un *langage* (au sens *syntaxe*) est simplement un ensemble de mots formés à partir d'un alphabet fini.
- Exemples de langage
  - l'ensemble des mots du dictionnaire (alphabet de `a` à `z`)
  - l'ensemble des mots constitués de deux `'a'` suivis d'un nombre quelconque de `'b'`: $\mathcal{L}=\{aa, aab, aabb, aabbb, \ldots\}$
- Un langage est dit *régulier* s'il est reconnu par un automate.
- Automate reconnaissant le mot `fee`:

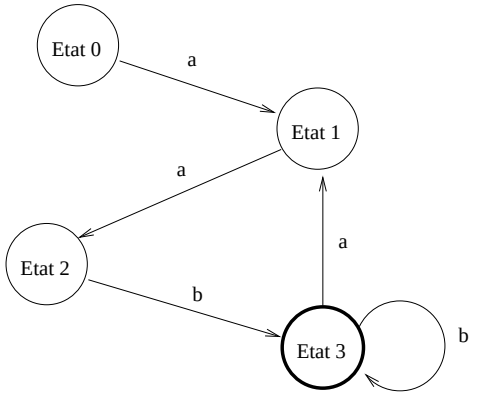


- Automate reconnaissant le langage $\{$`fee`, `feu`$\}$

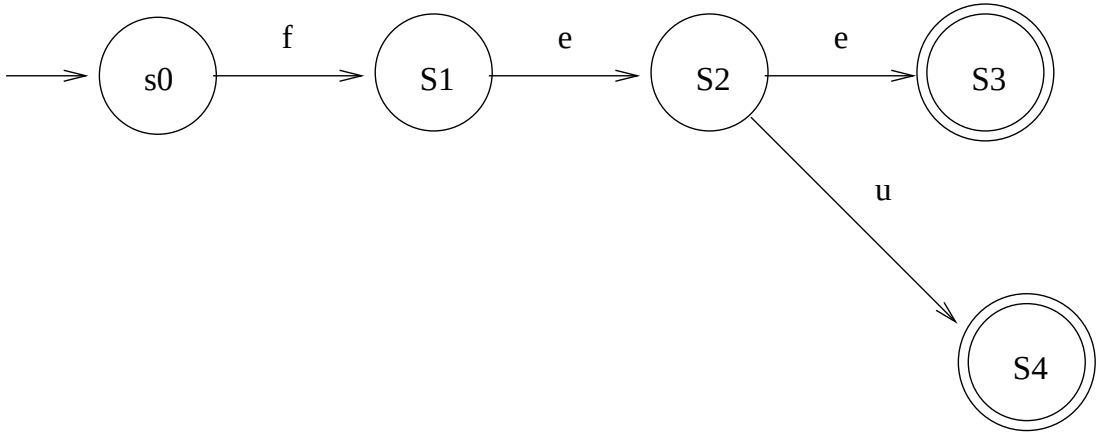


Notion d'automate

- Un automate est une collection de K états numérotés de 0 à K-1, ainsi qu'une collection de transitions
- Un état particulier est l'état initial.
- Tous les états sont soit des états d'acceptation et soit des états de refus
- Les transitions, sont étiquetées
  1. soit par des actions (par exemple, je lis la lettre `x`)
  2. soit par des condition (par exemple, la lettre `x` est présente)
- le triplets (état 1, lettre `x`, état 2) signifie: si je suis dans l'état 1 et que je lis la lettre `x`, alors je vais dans l'état 2.



Notion de mot reconnu



- `fee` → reconnu
- `feu` → reconnu
- `fei` → non reconnu (impossible de lire `'i'`)
- `fe` → non reconnu (arrêt dans un état non final)

Exemple d'application: Expressions régulières

- Les expressions régulières sont couramment employées en ligne de commande, par exemple: `ls *.c`
- Une *expression régulière* `X` basée sur un alphabet décrit un *langage*, c'est à dire un ensemble de mots sur cet alphabet, on note ce langage `E(X)`.
- Exemples:

Expression régulière	langage reconnu
`a*.b`	mots constitués d'un nombre quelconque de a suivi d'un b
a.(a+b)*.a + b.(a+b)*.b	mots constitués des lettre a et b, commençant et finissant par la même lettre

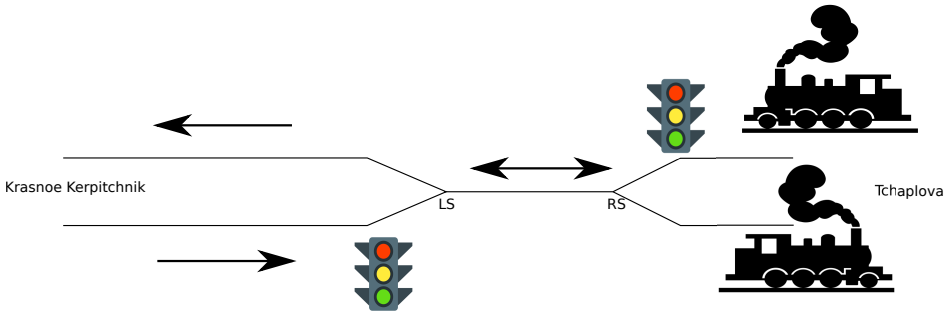
- Les mots décrits par des expressions régulières peuvent être reconnus par des automates

Link with architecture: Computers are automata

- Every computing machine is an automata
- Computer are *universal* in the sense that the program gives much flexibility in the action performed.
- In fact the basic action of a computer is very repetitive:
  - Read the instruction at $PC in memory
  - decode the instruction
  - send the decoding to the ALU (or to memory if it is a load)
  - increment $PC
- Dedicated circuits (ASICs) are automata designed for specific tasks.

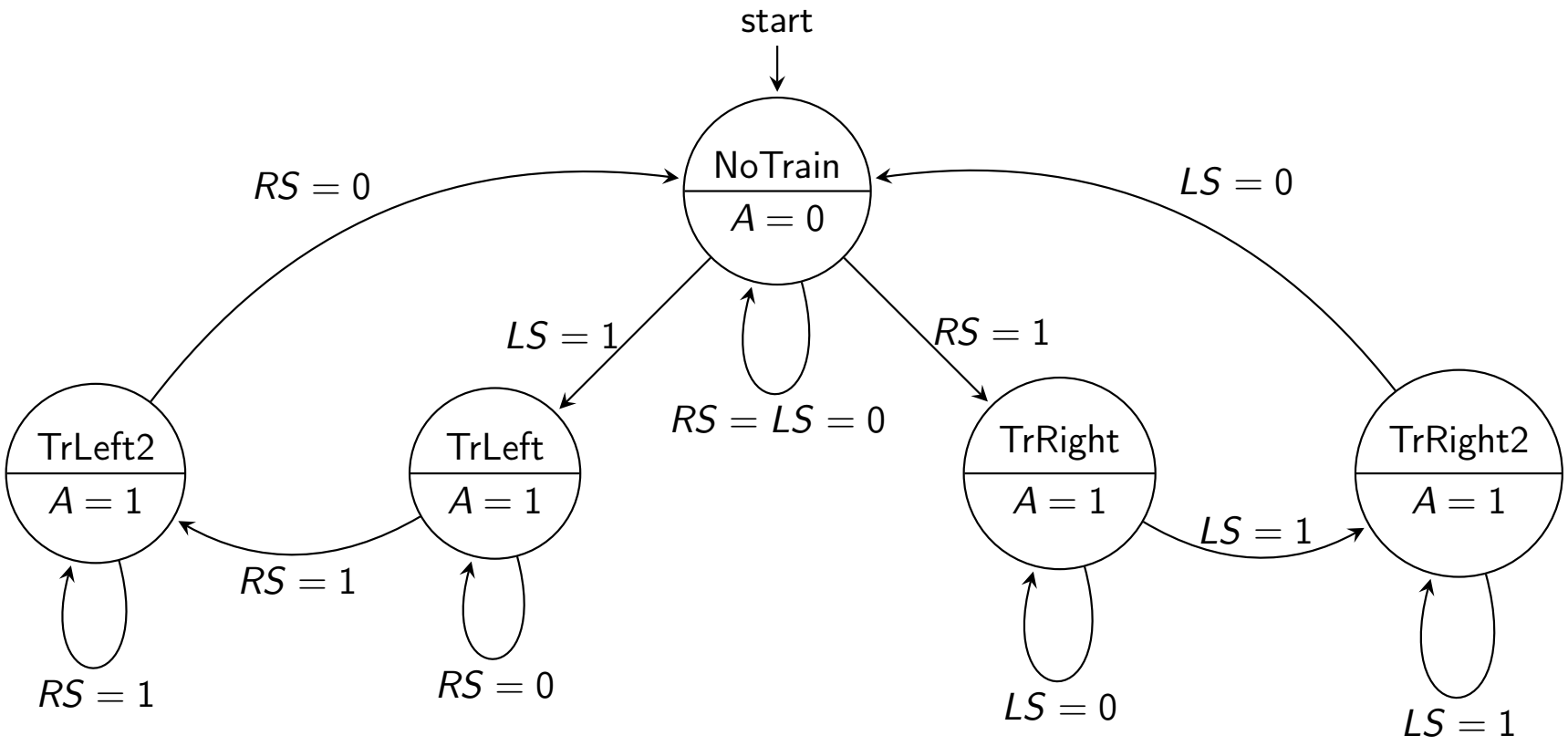
Table of Contents

Example from the poly

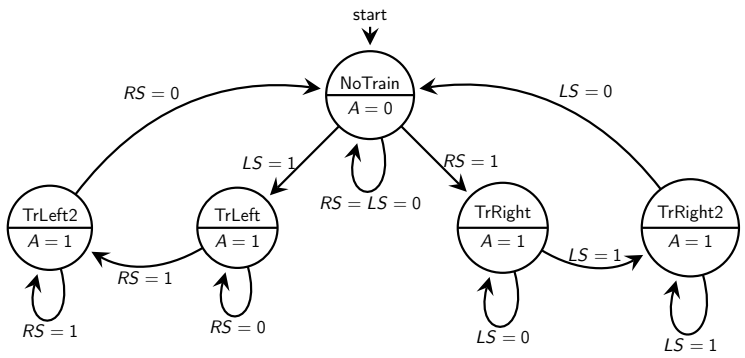


- A piece of unique train track for both train directions between the cities `T.` et `K.`
- Sensors triggered by train weight on rallways will command red lights when the track is used by a train.
- Modeling:
  - A booleen `A` (for 'Ampoule') indicating the state of the red light
  - Two booleans (LS for Left Sensor and RS for Rigth sensor) indicating the states of the sensors
  - An automaton to command the red lights

The Russian train automaton



The Russian train automaton



- Circles are *states* of the automaton (e.g. `NoTrain` state models the cases where no train stand on the track).
- States specifies output Values (here only one: `A`)
- Arrows are *transitions*, labeled by event that triggered them.

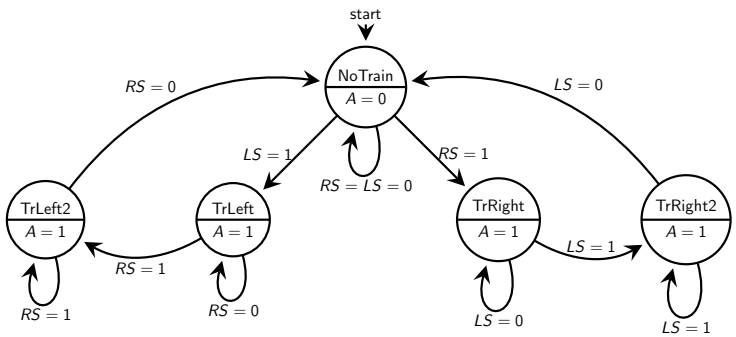
Formal Definition of Synchronous Automata

Formaly, a synchronous automaton (as use in ARC course) is a quintuplet (I, O, S, T, F, s_0) where:

- I is a set of Inputs (binary values usually),
- O is a set of Outputs (binary values usually)
- S is a finite set of states.
- T is the *transition function* from $S \times I \rightarrow S$
- F is the *output function* from $S \rightarrow O$
- $s_0 \in S$ is a special state called *initial state*, where we start from.

The *synchronous* specifier indicates that there is a kind of *clock* that triggeres the evaluation of the transition and output fonction.

Back to the Russian train example

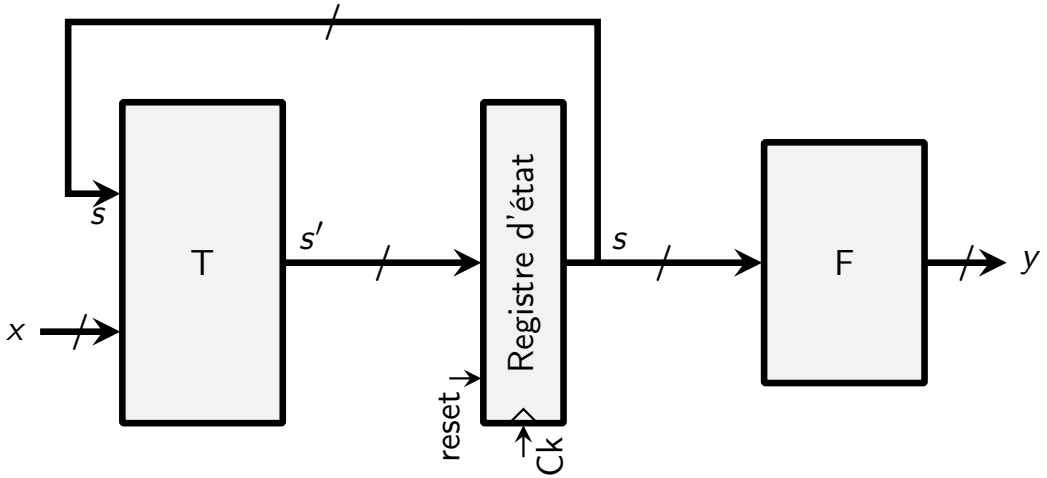


- The Inputs are `RS` and `LS` sensors Boolean values
- The Output is the value of Boolean `A`
- The functions (Transition and Output) can be defined by tables ⇒
- `X` means 'don't care'

s	x=(LS, RS)	s'=T(s,x)
NoTrain	00	NoTrain
NoTrain	01	TrRight
NoTrain	10	TrLeft
NoTrain	11	XXX
TrRight	0X	TrRight
TrRight	1X	TrRight2
TrRight2	1X	TrRight2
TrRight2	0X	NoTrain

s	y=F(s)
NoTrain	0
TrRight	1
TrRight2	1

Implementation of a synchronous automaton as a circuit

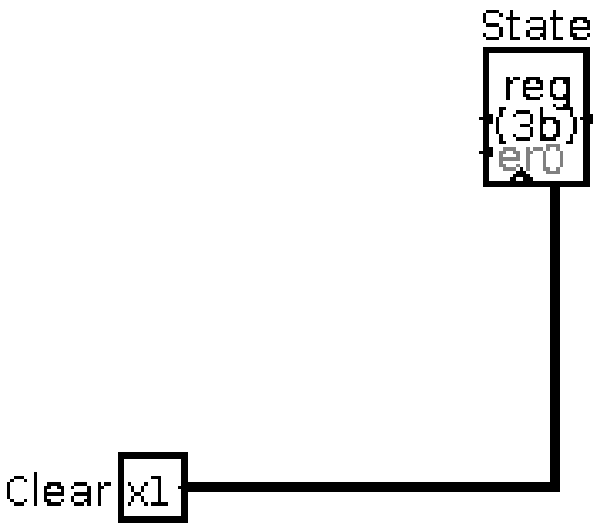


- `s` is current state, `s'` is next state, `x` are input bits, `y` are output bits.
- `Ck` and `reset` are not considered as inputs
- State change will occur on each rising edge of the Clock.

Implementation in Logisim

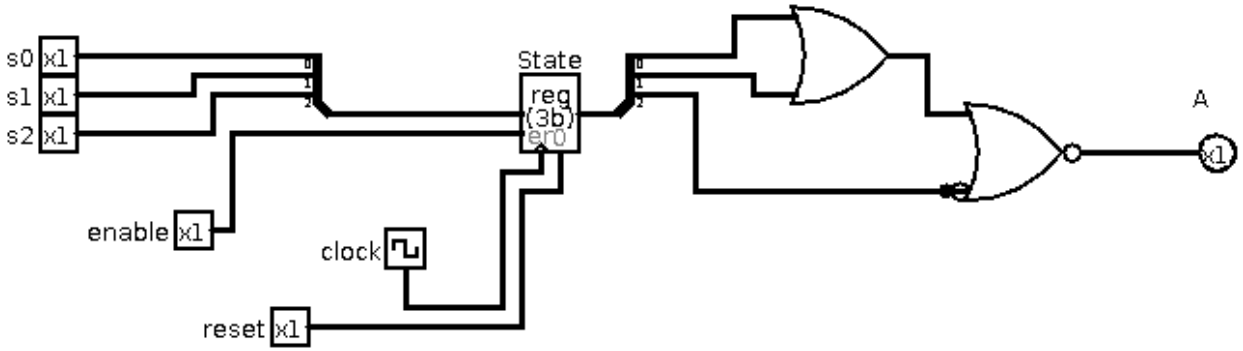
- We need to store 5 States, hence we need at least 3 bits:

value (binary)	state
100	NoTrain
000	TrRight1
001	TrRight2
010	TrLeft
011	TrLeft2



Russian train output function

- The output function is easy: A is on iff state is ''NoTrain'



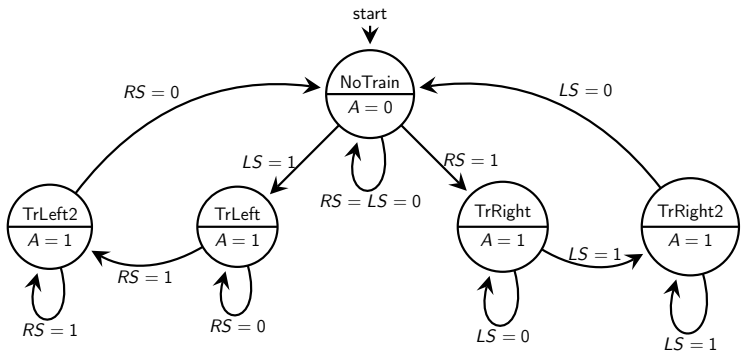
Russian train Transition function

s	x=(LS, RS)	s'=T(s,x)
100 (NoTrain)	00	NoTrain
100 (NoTrain)	01	TrRight
100 (NoTrain)	10	TrLeft
100 (NoTrain)	11	XXX
000 (TrRight)	0X	TrRight
000 (TrRight)	1X	TrRight2
001 (TrRight2)	1X	TrRight2
001 (TrRight2)	0X	NoTrain
010 (TrLeft)	X0	TrLeft
010 (TrLeft)	X1	TrLeft2
011 (TrLeft2)	X1	TrLeft2
011 (TrLeft2)	X0	NoTrain

Russian train Transition function

How do we build a transition function

- The general method is to start from the truth tables (previous slide) and to apply a systematic method to buid a logic function that corresponds to this table (see course 1)
- Or we can use a property of the transition table.
- For instance here, we remark that there is only one event that produces a change of state, in any state

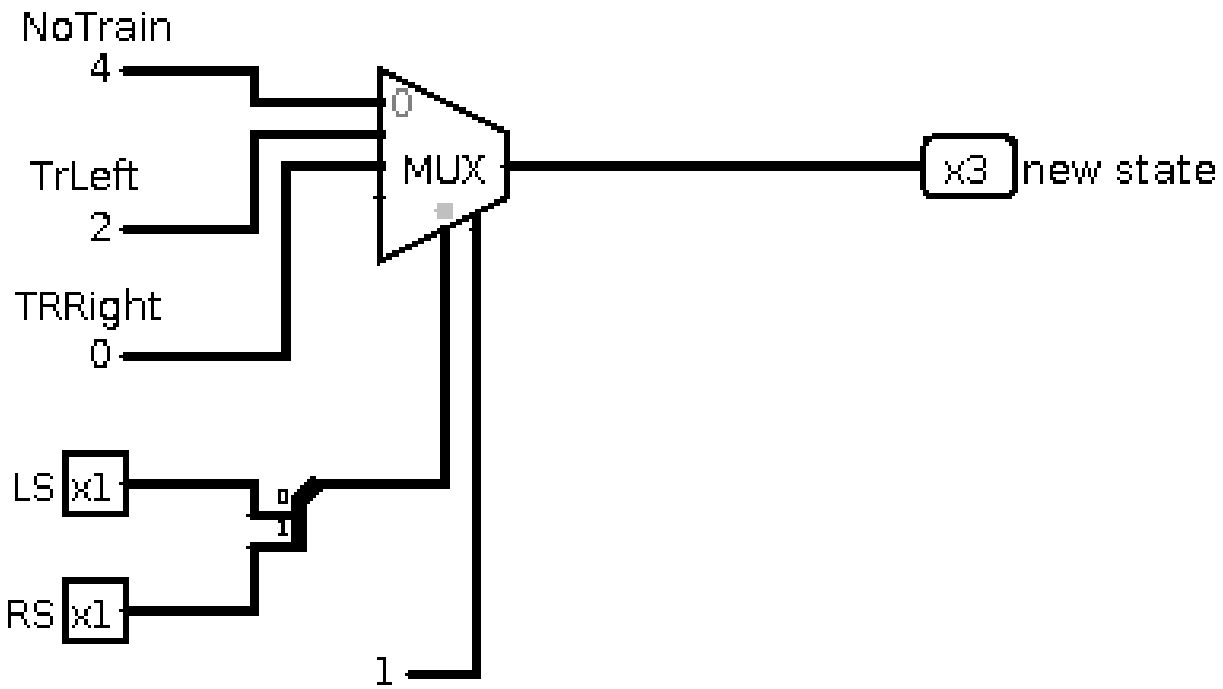


A possible Transition function

- Select a "next state" function for each state
- compute each of them
- Select the output depending on the value of the state

Example of the transition from the NoTrain state

s	x=(LS, RS)	s'=T(s,x)
100 (NoTrain)	00	NoTrain
100 (NoTrain)	01	TrRight
100 (NoTrain)	10	TrLeft
100 (NoTrain)	11	XXX



The complete Transition Automaton

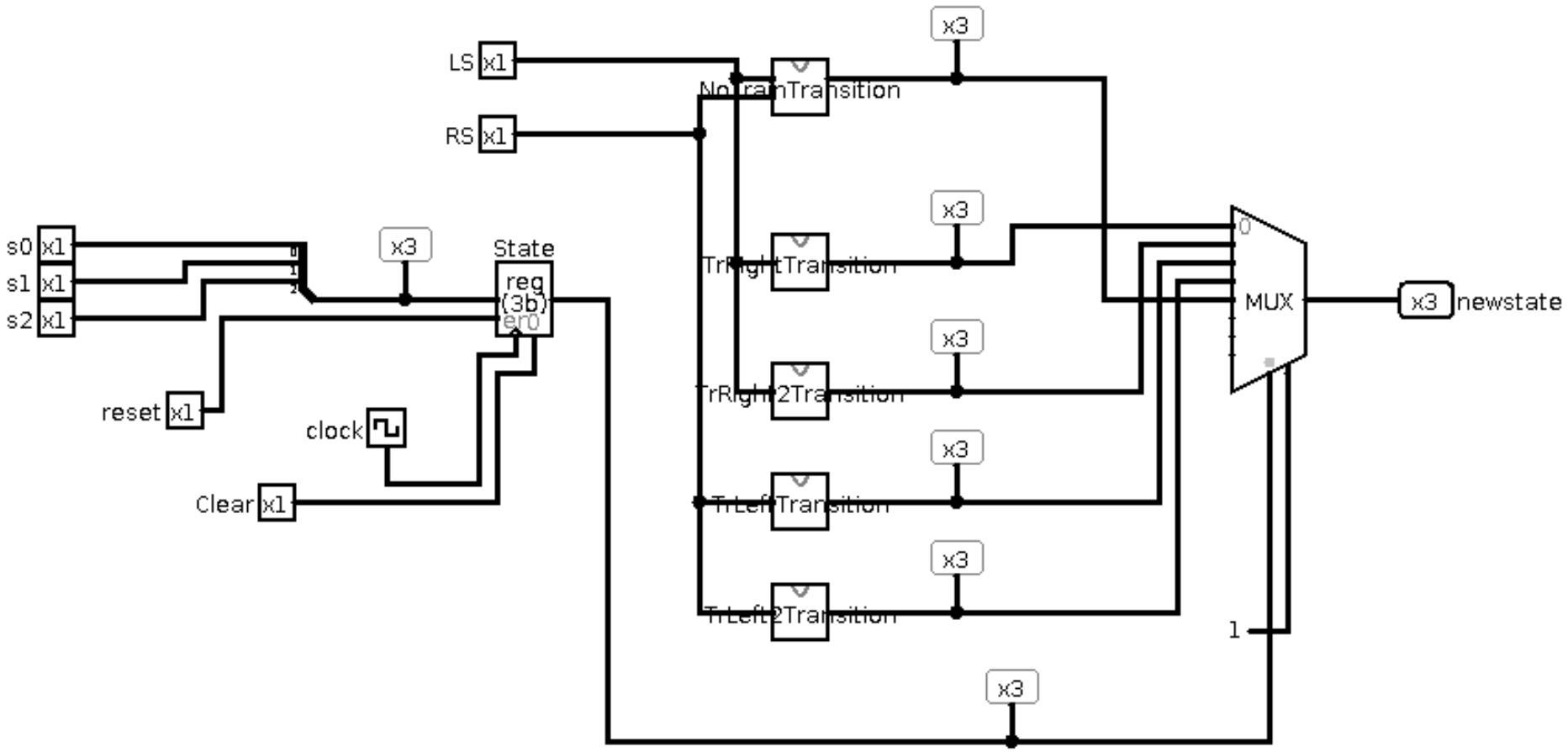


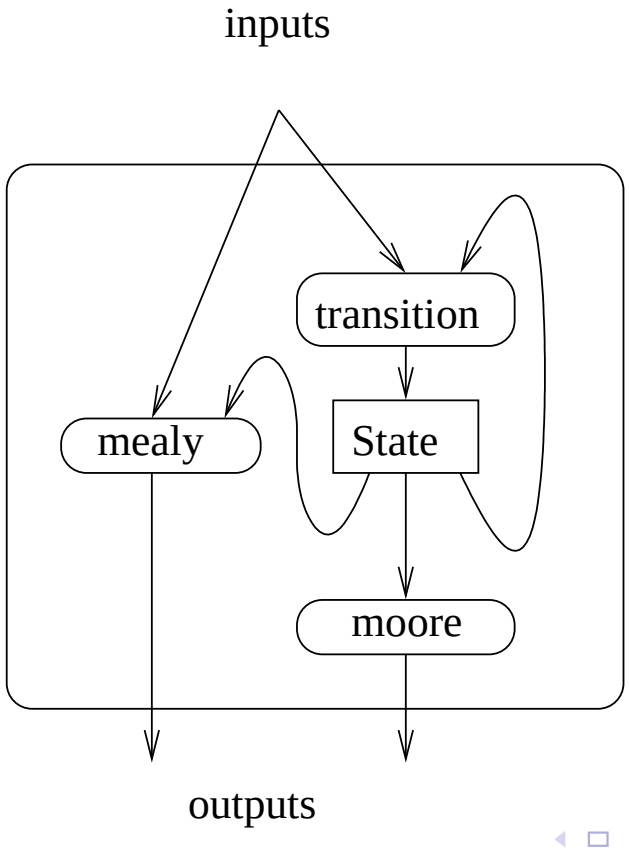
Table of Contents

Comming back to automata

- Automata are very widely used in computer science in different domains.
- In ARC we use them to *control the execution of dedicated synchronous circuits*
- As soon as a dedicated circuit is designed, there is an automaton designed.

Mealy and Moore automata

- We have seen a *Moore automaton*: output only depend on the state (not on the input), usually simpler to handle.
- The most
  general form of an automaton has a moore output and a mealy output



Summery: from Algorithm to Circuit

- From algorithm to automata (states and input/output)
- From automata to synchronous automata
- From synchronous automata to digital design

Lab topic: circuit for integer division

```
n := entrée N
p := entrée P
x := 0
q := 0
tant que x+p ⩽ n
    x := x+p
    q := q+1
fin tant que
sortie Q := q
```

Lab topic: proposed circuit to realize it