

Informatique Fondamentale IMA S8

Cours 3: Counter automata, Turing Machines and decidable problems

Laure Gonnord

<http://laure.gonnord.org/pro/teaching/>
Laure.Gonnord@polytech-lille.fr

Université Lille 1 - Polytech Lille

March/April 2011



Université
Lille1
Sciences et Technologies



POLYTECH[®]
LILLE

III - Counter automata, Turing Machines and decidable problems

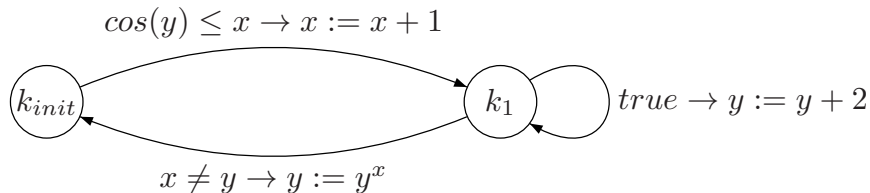
- 1 Counter automata
- 2 Programs
- 3 Turing Machines
- 4 Decidability - Complexity

What for ?

Express more than context-free languages !

- ▶ Give a way for automata to “**count** more” : include **variables**.

Example



General definition

Counter Automata

- A finite number of **counters** (variables)
 - A finite number of **control points**
 - Transitions between them that operate on counters.
- ▶ The transition function is of the form $g \rightarrow a$ (g is the **guard**, a is the **action**)

How it works ? 1/3

State

A **state** is (q, σ)

- q is the current control point
- $\sigma : Var \rightarrow Val$ is a function that assigns a value (a real one or \perp) to all counters.

► Non deterministic/No notion of acceptance/Notion of **reachability**.

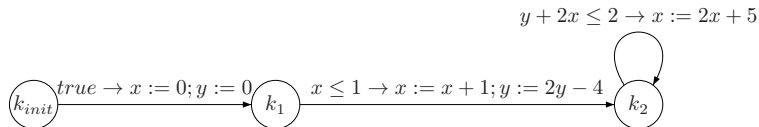
How it works ? 2/3

Given a state (q, σ) , the next one is computed by the help of the transition function :

- **enabled transitions** are transition of the current control points where the current valuations of variables satisfy the guard.
- Pick one enabled transition, and compute the next state : (q', σ') . The new values for the variables are computed w.r.t. the action.

How it works ? 3/3

Example of an affine automata :

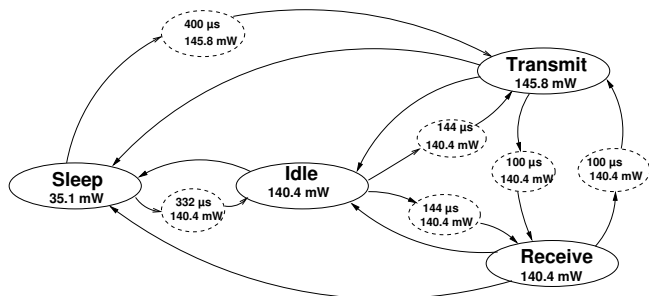


- ▶ Compute successive states.
- ▶ Exercises.

What kind of systems ?

These automata are used to encode, for example :

- Simple systems (coffee machine, ...) **specifications**
- Energy consumption of sensors :



What for ?

Some classical problems :

- Automatically finding invariants
- **Reachability** analysis
- Formula proving.
- (deterministic) Code generation.

- 1 Counter automata
- 2 Programs
- 3 Turing Machines
- 4 Decidability - Complexity

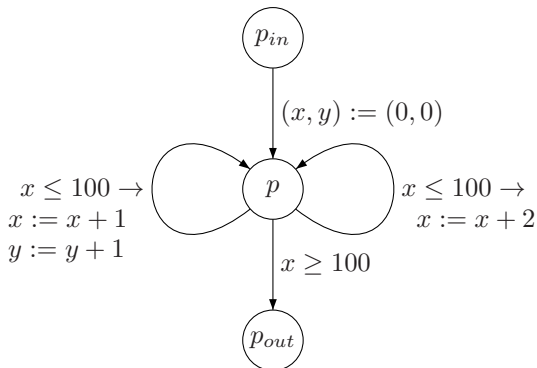
Expressing programs as counter automata

An **example** :

```

x:=0;y:=0
while (x<=100) do
  read(b);
  if b then
    x:=x+2
  else begin
    x:=x+1;
    y:=y+1;
  end;
endif
endwhile

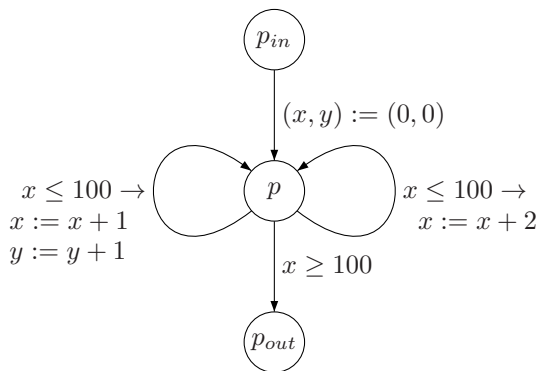
```



► Some approximations are made (arrays, Boolean, ...)

Expressing properties of programs

Some (**safety**) properties of programs can be expressed inside the counter automaton :



- ▶ Encode the fact that $state = p_{out} \wedge y > 100$ is “bad”
- ▶ Some modifications of the automaton can be automatically done.

Expressing properties of programs - 2

Automatically deriving bad states :

```
int j;
char user[USERSZ];

for(j = 0; line[j] != EOS; ++j)
    if (!strchr("-", line[j]))
        break;

if(j == J && line[j] == ' ') { /* long list */
    /* BUG! No bounds check. */
    assert(USERSZ >= N - j, "badstate");

    r_strcpy (user, line + j);
}
```

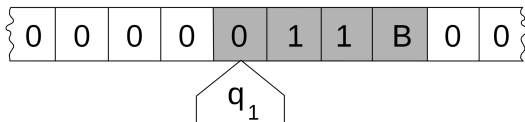
► “badstate” is reachable.

Expressing programs as counter automata

- **Pros** : finding and proving program properties
 - **Cons** : how do we define the operations ? the fact that integers are encoded on bits ?
- ▶ There is a need for a more accurate (not specialised) model !

- 1 Counter automata
- 2 Programs
- 3 Turing Machines**
- 4 Decidability - Complexity

Turing Machine, def - 1



Turing Machine, Turing, 1935

- A **tape** (infinite in both sides) : the memory. (it has a working alphabet which contains a **blank** symbol).
- A **head** : read/write symbols on the tape.
- A finite transition table (or function)
- ▶ A PushDown automaton which is more flexible.

image credits : Wikipedia.org

Turing Machine, def - 2

Turing Machine elements

- States (Q), initial state (q_0), final (accepting) states (F).
 - One alphabet Γ for the tape.
 - $b \in \Gamma$ the blank letter.
 - Transitions are finitely many : read the letter under the head, and then :
 - Erase a symbol or write one under the head
 - Move the head (read, write, or stays)
 - The “state” can be modified.
- The transition function is :

$$Q \times \Gamma \times \rightarrow Q \times \Gamma \times \{L, R, S\}$$

An Example

TM that decides if x is even : (final State : q_4)

State/char	0	1	B
q_1	$(q_1, 0, R)$	$(q_1, 1, R)$	(q_2, B, L)
q_2	(q_3, B, L)	(q_4, B, L)	—

Play on $BBBBBBB11BBBBB$ and $BBBBBBB10BBBBB$

► A **configuration** is a tuple (word on the tape, position of the head, state).

Adapted from : http://www.madchat.fr/coding/algo/algo_epfl.pdf slide 4

Another Example

Demo of a TM recognising a language : $a^n b^n c^n$

Program found here :

<http://www.cs.columbia.edu/~zeph/software/BJDweck/>

General results 1/2

There exists Turing machines for the following languages :

- palindromes
 - $a^n b^n c^n$ (non algebraic language)
 - a^i with i prime (non algebraic)
 - a^{n^2} , with $n \geq 0$
- Turing machines are more powerful than all other models (we have seen yet)

Decidable Languages

A language that is recognised by a Turing Machine is said to be **decidable**.

Accepting or computing

A TM can also compute functions

TM that writes 1 if x is even, 0 else (q_6 is final state) :

State/char	0	1	B
q_1	$(q_1, 0, R)$	$(q_1, 1, R)$	(q_2, B, L)
q_2	(q_3, B, L)	(q_4, B, L)	—
q_3	(q_3, B, L)	(q_3, B, L)	$(q_5, 1, R)$
q_4	(q_4, B, L)	(q_4, B, L)	$(q_5, 0, R)$
q_5	—	—	(q_6, B, R)

Play on $BBBBBBB11BBBBB$ and $BBBBBBB10BBBBB$

Another Example

Demo of a TM computing the subtraction.

General results 2/2

There exists Turing machines for the computation of :

- $x \mapsto x + 1$
- $(x, y) \mapsto x + y$
- $x \mapsto x^2$
- all the functions you are able to write on computers

Computable functions

A function (defined for all its input) that is computable by a Turing Machine is said to be **TM computable**

▶ A Model of what can be computed with machines. (**Church Thesis**)

- 1 Counter automata
- 2 Programs
- 3 Turing Machines
- 4 Decidability - Complexity

Decidable - Semi-decidable languages

If L is a language, L is **decidable** if there exists a Turing Machine (or an algorithm) that outputs for all w :

- 1 if $w \in L$
- else 0.

Semi-decidable :

- 1 if $w \in L$
 - else does not terminate
- ▶ equivalent definition for problems.

Complexity

Link with computational complexity :

- The number of steps in the execution of a TM gives the **complexity in time**,
- The number of seen squares in the execution of a TM gives the **complexity in space**.

Examples of undecidable problems

- The halting problem for TM or counter automata (for more than 2 counters)
- Given a program, does it loop ?
- Is a given algebraic expression (with log, *, exp, sin, abs) equal to 0 ? (Richardson, 1968)
- The 10th Hilbert Problem (Diophantine equations)

Summary

Turing Machines :

- A model closed to **programming languages**.
- Non deterministic.
- Acceptors for decidable languages. But some languages are **not decidable** !
- (equivalently) Computes solutions to problems. But ...

Important fact

Some common problems are undecidable !

Counter automata :

- are a more simpler model
- have the same power of expression as Turing Machines.
- Reachability is **undecidable** too. But we can do approximations.