Advanced Human-Machine Interaction

Interaction Data Analysis

Automata / State Machine

Alexandre Pauchet
BO.B.RC.18, alexandre.pauchet@insa-rouen.fr

INSA Rouen Normandie - ASI departement

Finite-state automata (1/8)

Formal definition

Finite-state automata

A *finite-state automaton* (or a *finite state machine*) is a 5-tuple $A = (S, E, T, S_0, F)$ with

- S a finite set of states,
- E an alphabet,
- T a state-transition function $T : SxE \rightarrow S$,
- $S_0 \subseteq S$ the set of initial states,
- $F \subseteq S$ the set of final states.

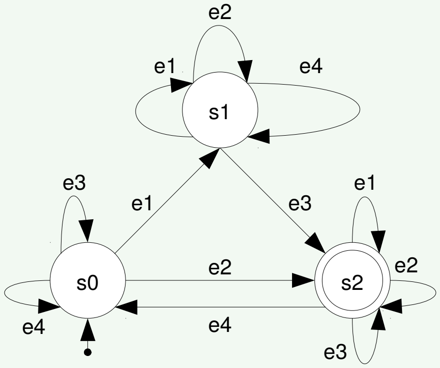
Representation

A finite-state automaton can be represented as an oriented graph whose vertices symbolise states and annotated edges describe transitions.

Finite-state automata (2/8)

Example

Example



- $S = \{s_0, s_1, s_2\}$,
- $E = \{e_1, e_2, e_3, e_4\}$,
- for T, see the figure,
- $s_0 \in S_0$ is the initial state,
- $s_2 \in F$ is the final state.

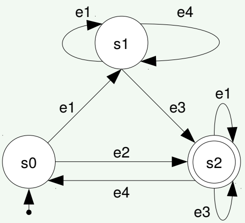
Finite-state automata (3/8)

Receptiveness

Definition

An automaton in a given state may only be *receptive* to a subset of E. The domain of definition of T is no longer the cartesian product (SxE) but a subset of this set.

Example



Finite-state automata (4/8)

Determinism and execution

Deterministic automaton

An automaton is called *deterministic* iff

- $card(S_0) = 1$,
- $\forall s \in S, \forall a \in E, card(T(s,a)) \leqslant 1$.

Execution

An *execution* is a sequence $s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \ldots$ in such a way that $\forall i \geqslant 0, s_{i+1} \in T(s_i, a_i)$ and $s_0 \in S_0$.

Finite-state automata (5/8)

Words and language

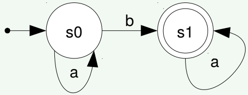
Word

A *word* $m \in E^\omega$ is supported/accepted/generated by an execution $c = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \dots$ iff $m = a_0 a_1 a_2 \dots$.
m can also be supported/accepted/generated by all the automata for which c is an execution.

Language

The *language* $L(A)$ is the (infinite) set of words accepted by A.

Exemple



$L(A) = \{b, ab, aab, \dots ba, baa, \dots, aba, \dots\}$

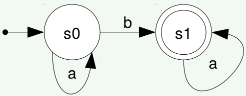
Finite-state automata (6/8)

Transitions tree

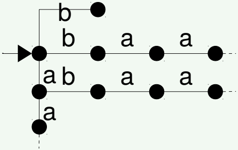
Language representation

- The language $L(A)$ of an automaton may be represented as a(n often infinite) tree.
- The language $L(A)$ definition of an automaton can be compacted.

Exemple



$L(A) = \{a * ba*\}$



Finite-state automata (7/8)

Particular notations

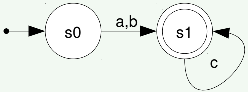
Word repetition

Let r be an execution such that $r = ...x * ...$ with $x \in E$ means that x is repeated an infinite number of times.

Multiple transitions

For $T(e_1, a) = e_2$ and $T(e_1, b) = e_2$ with $a \neq b$ represented as "a, b" for a unique transition in the corresponding automaton.
In the language, the alternative '|' is therefore used.

Exemple



$$L(A) = \{(a|b)c*\}$$

Finite state automata (8/8)

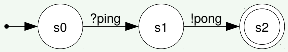
Transitions

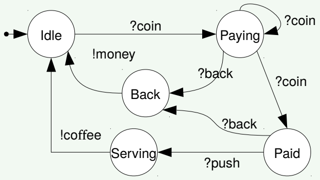
Complex systems

When an automaton models an interacting system element, a specific notation can be used for transitions:

- ?: message reception or external action
- !: message sending or internal action

Exemples





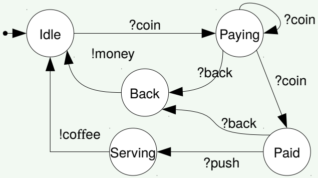
Hybrid automata(1/4)
Problem

Goal

- Maintain determinism
- Maintain a limited number of states
- Configure automata / upgrade automata with variables

Exemples

Non-deterministic automaton:



Hybrid automata (2/4)
Definition

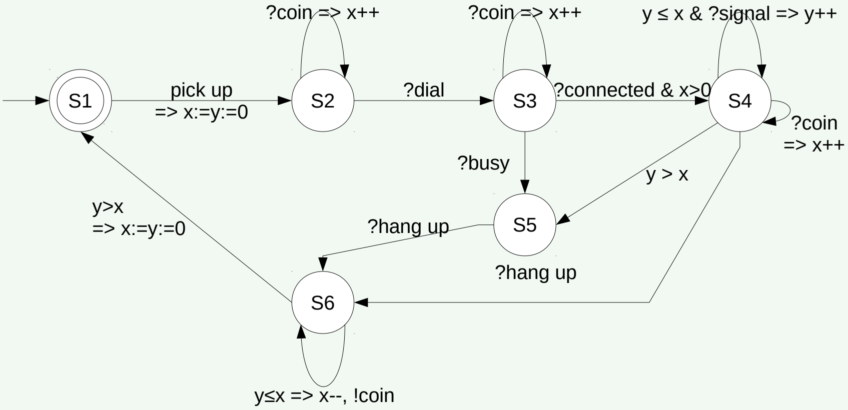
Hybrid automata

- A *hybrid automaton* is an automaton to which we add internal variables (mainly counters) that the automaton can read or modify when enabling a transition. A transition
  - can be enabled if its *preconditions* (or *guards*) are verified,
  - can trigger a set of internal actions in order to modify its variables.
- Usually, a value assignment is noted '$:=$' and an internal action can be specified as a '$\Rightarrow$'.
- Often, *semantics* are provided to clarify the transitions.

Hybrid automata (3/4)

Example: phone box modelling

Exemple



Hybrid automata (4/4)

Transition semantics

Reduction rule

$$\{Preconditions\}\frac{s_i \xrightarrow{(!|?)message_a} s_f}{a_1;\ \ldots;\ a_n}$$

with:

- s_i the initial state,
- s_f the final state,
- *message* the message sent ! (resp. received ?) to (resp. from) the automaton,
- $a_1;\ \ldots;\ a_n$ the internal actions to apply.

Exemples

$\{x > 0\}\frac{s_3 \xrightarrow{?connected} s_4}{\emptyset}$ $\{y \leqslant x\}\frac{s_4 \xrightarrow{?signal} s_4}{y++}$ $\{\}\frac{s_4 \xrightarrow{?coin} s_4}{x++}$

Timed automaton (1/8)

Motivation

Timed automata goal

- Ensure the proper functioning of a system taking into account time constraints.
  Examples: a toaster, a server to synchronise, *etc...*
- Use proven models, enhancing them with time constraints
- Be limited to a model allowing the application of verification methods

Timed automata (2/8)

Description

What is a timed automaton?

- Timed automaton = Finite-state automaton + a set of variables + a set of real-valued clocks
  - which increase simultaneously,
  - that can be reset independently
- A clock measures time elapsed since its last initialisation
  $\Rightarrow$ Interval measurement between two events
- In the basic model, there is no clock drift

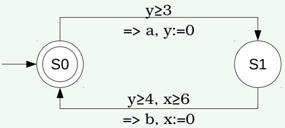
Run

We call *run* a timed automaton execution.

Timed automata (3/8)

Example

Timed automata examples



Running example

- $(s_0, (0,0)) \xrightarrow{a,3(+3)} (s_1, (3,0)) \xrightarrow{b,7(+4)} (s_0, (0,4)) \ldots$
- $(s_0, (0,0)) \xrightarrow{a,3.2(+3.2)} (s_1, (3.2,0)) \xrightarrow{b,7.5(+4.3)} (s_0, (0,4.3)) \ldots$

Timed automata (4/8)

Formal definition

Definition

A *timed automaton* is a 7-tuple $A = (S, E, S_I, S_F, X, I, T)$ with

- S a final states set,
- E a finite alphabet,
- S_I the initial states set,
- S_F the final states set,
- X a finite set of clocks,
- $I : S \rightarrow C(X)$ associates each state with a time constraint,
- $T \subset SxExC(X)x2^X xS$ is a set of action transitions.

Exemple

$t_{i \rightarrow f} =< s_i, e, \alpha, \lambda, s_f >\in T$ is a transition from s_i to s_f, guarded by a constraint α, tagged with an event e which modifies clocks values following $\lambda : X \rightarrow \mathbb{N}$.

Timed automata (5/8)

Timed automaton semantics

Tagged transitions

1. State: (s, v)
   - $s \in S$ is a state,
   - v is a *clock valuation*, i.e. a function associating a value with each clock of X.
2. Initial state: (s_0, v_0)
   - $s_0 \in S_0$
   - $v_0(x) = 0, \forall x \in X$
3. A relation of transitions between states

Timed automata (6/8)

Transitions between states

2 types of state change

1. Caused by time flow: v evolves, s doe snot

$$(s, v) \xrightarrow{\delta} (s, v + \delta)$$

2. Caused by a change of location: v does not evolve, s does

$$(s, v) \xrightarrow{a} (s', v')$$

Observation: only the clocks reset by the transition does not have the same value in v and v'

Timed automata (7/8)

Transition relation

Property

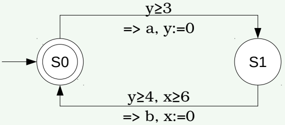
$$(s_{i-1}, v_{i-1}) \xrightarrow{e_i,(\tau_i - \tau_{i-1})} (s_i, v_i)$$

iff there is an edge $< s_{i-1}, e_i, \phi_i, \lambda_i, s_i >$ such that

- $v_{i-1} + (\tau_i - \tau_{i-1})$ satisfies ϕ_i
- $v_i = \lambda_i(v_{i-1})$

Timed automata (8/8)
Example

Exemple



$$(s_0, (0,0)) \xrightarrow{a,3.2} (s_1, (3.2, 0)) \xrightarrow{b,T} (s_0, (X, Y)) \dots$$

$$v_{i-1} + (\tau_i - \tau_{i-1}) = \begin{cases} x : 3.2 + (T - 3.2) = T \\ y : 0 + (T - 3.2) = T - 3.2 \end{cases}$$

$$\phi_{s_1 \to s_0} = \begin{cases} x \geq 6 \\ y \geq 4 \end{cases} \text{ so } \phi_{s_1 \to s_0} \, True \Rightarrow \begin{cases} T \geq 6 \\ T - 3.2 \geq 4 \end{cases} \text{ and so } T \geq 7.2$$

e.g.: $T = 7.2$, $v_{i-1} + (\tau_i - \tau_{i-1}) = (7.2, 4)$, $\Rightarrow v_i = (0, 4)$

Conclusion (1/3)

Limitations of finite-states automata

Limitations and solutions

1 automaton = 1 entity of the system

- How to coordinate 2 entities?
  $\Rightarrow$ Communication between automata (*synchronous*/*asynchronous*)
- Is it possible to introduce some kind of memory?
  $\Rightarrow$ Hybrid automata
- How to represent time-related transitions?
  $\Rightarrow$ Timed automata
- How to represent a complex system?
  $\Rightarrow$ Automata network, *Petri* network
- Automatised construction of automata?
  $\Rightarrow$ Supervised/unsupervised learning

Conclusion (2/3)

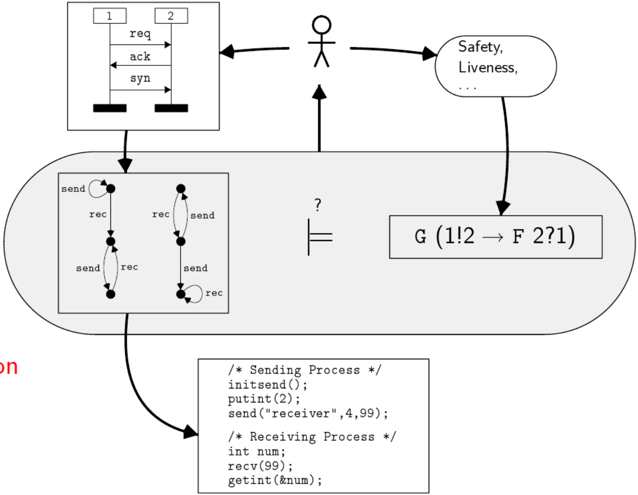
Modelling and verification

specification

synthesis

modeling & verification

code generation



From Benedikt Bollig's lesson http://www.lsv.ens-cachan.fr/~bollig/mpri/2007/decembre10.pdf.

Conclusion (3/3)

References

Links

- http://www.pps.jussieu.fr/~rifflet/enseignements/AF3/
- http://www-igm.univ-mlv.fr/~desar/Cours/automates/ch1.pdf
- http://mpri-wiki.inria.fr/bin/view/WebSite/2007-2008-C-2-8
- http://lifc.univ-fcomte.fr/~julliand/CoursAutomates.pdf
- http://www-master.ufr-info-p6.jussieu.fr/2005/IMG/pdf/Automates_tempo.pdf
- http://www.lamsade.dauphine.fr/~berard/PDF/MCT.pdf
- http://www.loria.fr/~bournez/cours/CoursDEAModelChecking/10-N-AutomatesTemporises.pdf

Article

- "*A theory of timed automata*", R. Alur and D. L. Dill, in Theoretical computer science, volume 126, pp. 183-235, 1994.