Modeling and analysis of hybrid systems

Modeling

Prof. Dr. Erika Ábrahám

Informatik 2 - Theory of Hybrid Systems
RWTH Aachen

SS 2010

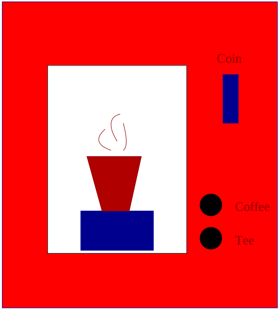
Contents

Motivation

- Dynamical system: evolution of the state over time
- Classification based on state type:
  continuous $\rightsquigarrow$ states from $\mathbb{R}^n$ (subclasses: linear/nonlinear)
  discrete $\rightsquigarrow$ states from a countable set
  hybrid $\rightsquigarrow$ both
- Classification based on time:
  continuous time $\rightsquigarrow$ $t \in \mathbb{R}$, evolution of the state is described by *ordinary differential equations (ODEs)* $\dot{x} = f(x, u)$
  discrete time $\rightsquigarrow$ $k \in \mathbb{Z}$, evolution of the state is described by *difference equations* $x_{k+1} = f(x_k, u_k)$
  hybrid time $\rightsquigarrow$ the evolution is over continuous time, but there are also discrete "instants" where something "special" happens

Example: Vending machine

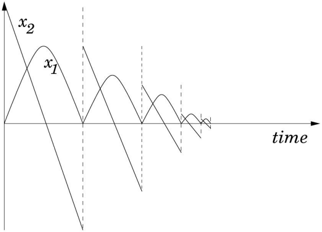
- insert coin
- choose beverage (coffee/tee)
- wait for cup
- take cup



$\rightsquigarrow$ Discrete space, discrete time

Example: Bouncing Ball

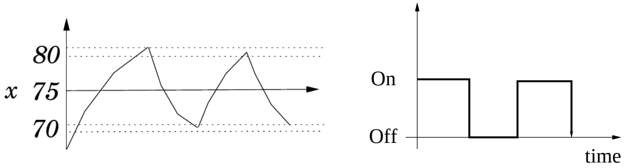
- vertical position of the ball x_1
- velocity x_2
- continuous changes of position between bounces
- discrete changes at bounce time



$\rightsquigarrow$ Continuous space, hybrid time

Example: Thermostat

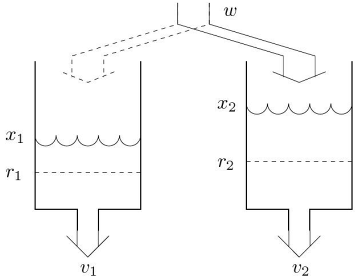
- temperature x controlled by switching a heater
- x regulated by thermostat:
  - $68° \leq x \leq 70°$ $\rightsquigarrow$ "heater on"
  - $80° \leq x \leq 82°$ $\rightsquigarrow$ "heater off"



$\rightsquigarrow$ Hybrid space, hybrid time

Example: Water Tank System

- two constantly leaking tanks v_1 and v_2
- hose w refills exactly one tank at one point in time
- w can switch between tanks instantaneously



$\rightsquigarrow$ Hybrid space, hybrid time

There are much more complex examples of hybrid systems...

Quelle: `www.digi-help.com`

- Automobils, trains, etc.
- Automated highway systems
- Collision-avoidance and free flight for aircrafts
- Biological cell growth and division

Aims:

- modeling
- analysis
- synthesis

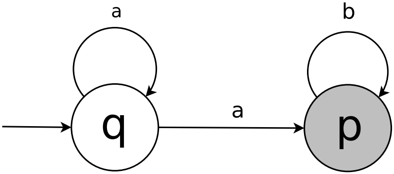
Contents

Nondeterministic finite automaton

Definition

A *nondeterministic finite automaton* (NFA) $\mathcal{A}$ is a tuple $\mathcal{A} = (Q, \Sigma, \Delta, q_0, F)$ with

- finite set of states Q,
- finite alphabet Σ,
- transition relation $\Delta \subseteq Q \times \Sigma \times Q$,
- initital state $q_0 \in Q$,
- set of final (accepting) states $F \subseteq Q$.



Nondeterministic finite automaton

- An execution $q_0\ \sigma_0\ q_1\ \sigma_1 \ldots$ of the NFA A is a (finite or infinite) sequence of states and inputs with q_0 the initial state and $(q_i, \sigma_i, q_{i+1}) \in \Delta$.
- The NFA A accepts a word $\sigma_0\ \sigma_1\ \ldots \in \Sigma^*$ iff there is an execution $q_0\ \sigma_0\ q_1\ \sigma_1\ \ldots$ visiting some finite states infinitely often.
- The language $L(A) \subseteq \Sigma^*$ accepted by A is the set of accepted words.
- Two NFA are equivalent iff they accept the same language.
- An NFA may be blocking.
- An NFA may be non-deterministic.

Deterministic finite automaton

Definition

A *deterministic finite automaton* (DFA) $\mathcal{A}$ is a tuple $\mathcal{A} = (Q, \Sigma, \Delta, q_0, F)$ with

- finite set of states Q,
- finite alphabet Σ,
- transition function $\Delta : Q \times \Sigma \to Q$,
- inititial state $q_0 \in Q$,
- set of final (accepting) states $F \subseteq Q$.

- For every NFA there is a DFA that accepts the same language.
- Language type: regular

Parallel composition

Definition

For two DFAs $\mathcal{A}_1 = (Q_1, \Sigma, \Delta_1, q_0^1, F_1)$ and $\mathcal{A}_2 = (Q_2, \Sigma, \Delta_2, q_0^2, F_2)$ we define $\mathcal{A}_1 || \mathcal{A}_2 = (Q, \Sigma, \Delta, q_0, F)$ with

- $Q = Q_1 \times Q_2$,
- $\Delta((q_1, q_2), a) = (\Delta_1(q_1, a), \Delta_2(q_2, a))$ for all $(q_1, q_2) \in Q_1 \times Q_2$ and $a \in \Sigma$,
- $q_0 = (q_0^1, q_0^2)$,
- $F = F_1 \times F_2$.

Contents

Labeled state transition system

Definition

A *labeled state transition system* (LSTS) is a tuple $\mathcal{LSTS} = (\Sigma, Lab, Edge, Init)$ with

- a (probably infinite) state set Σ,
- a label set Lab,
- a transition relation $Edge \subseteq \Sigma \times Lab \times \Sigma$,
- non-empty set of initial states $Init \subseteq \Sigma$.

Labeled (state) transition system

Definition

A *labeled transition system* (LTS) is a tuple
$\mathcal{LTS} = (Loc, Var, Lab, Edge, Init)$ with

- finite set of locations Loc,
- finite set of (typed) variables Var,
- finite set of synchronization labels Lab, $\tau \in Lab$ (stutter label)
- finite set of edges $Edge \subseteq Loc \times Lab \times 2^{V^2} \times Loc$ (including stutter transitions (l, τ, Id, l) for each location $l \in Loc$),
- initial states $Init \subseteq \Sigma$.

with

- valuations $\nu : Var \to Domain$, V is the set of valuations
- state $\sigma = (l, \nu) \in Loc \times V$, Σ is the set of states

Semantics of LTS

Operational semantics has a single rule:

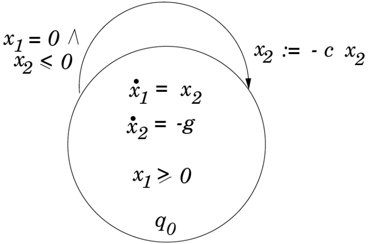
$$\frac{e = (l, a, \mu, l') \in Edge \quad (\nu, \nu') \in \mu}{(l, \nu) \xrightarrow{a} (l', \nu')}$$

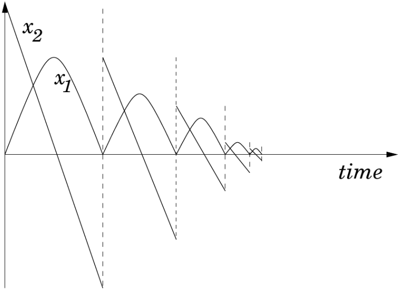
- system *run* (execution): $\sigma_0 \xrightarrow{a_0} \sigma_1 \xrightarrow{a_1} \sigma_2 \ldots$ with $\sigma_0 \in Init$
- a state is called *reachable* iff there is a run leading to it

Example: modeling a simple while-program

Example revisited: Bouncing Ball

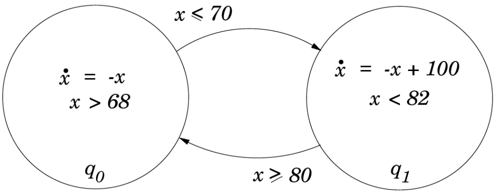
- vertical position of the ball x_1
- velocity x_2
- continuous changes of position between bounces
- discrete changes of bounce time

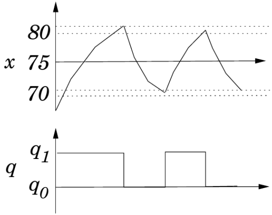




Example revisited: Thermostat

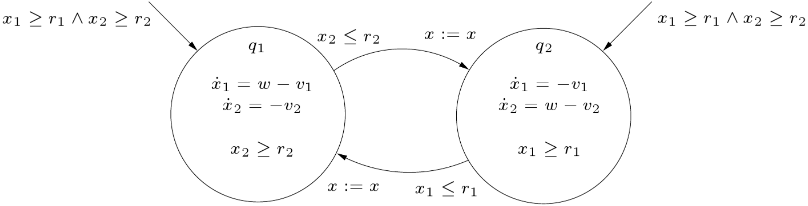
- $68° \leq x \leq 70° \rightsquigarrow$ "heater on"
- $80° \leq x \leq 82° \rightsquigarrow$ "heater off"





Example revisited: Water Tank System

- two constantly leaking tanks v_1 and v_2
- hose w refills exactly one tank at one point in time
- w can switch between tanks instantaneously



Hybrid automaton

Definition

A *hybrid automaton* $\mathcal{H}$ is a tuple $\mathcal{H} = (Loc, Var, Lab, Edge, Act, Inv, Init)$ with

- finite set of locations Loc,
- finite set of real-valued variables Var,
- finite set of synchronization labels Lab, $\tau \in Lab$ (stutter label)
- finite set of edges $Edge \subseteq Loc \times Lab \times 2^{V^2} \times Loc$ (including stutter transitions $(l, \tau, \mathtt{Id}, l)$ for each location $l \in Loc$),
- Act is a function assigning a set of activities $f : \mathbb{R}^+ \to V$ to each location; the activity sets are time-invariant, i.e., $f \in Act(l)$ implies $(f + t) \in Act(l)$, where $(f + t)(t') = f(t + t')$ f.a. $t' \in \mathbb{R}^+$,
- a function Inv assigning an invariant $Inv(l) \subseteq V$ to each location $l \in Loc$,
- initial states $Init \subseteq \Sigma$.

with

- valuations $\nu : Var \to \mathbb{R}$, V is the set of valuations
- state $(l, \nu) \in Loc \times V$, Σ is the set of states
- transitions: discrete and time

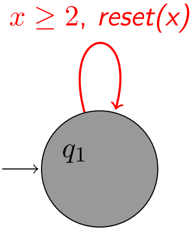
Semantics of hybrid automata

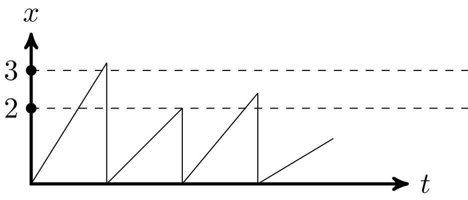
$$\frac{(l, a, \mu, l') \in Edge \quad (\nu, \nu') \in \mu \quad \nu' \in Inv(l')}{(l, \nu) \stackrel{a}{\rightarrow} (l', \nu')} \quad \texttt{Rule}_{\texttt{Discrete}}$$

$$\frac{\begin{array}{c} f \in Act(l) \quad f(0) = \nu \quad f(t) = \nu' \\ t \geq 0 \quad \forall 0 \leq t' \leq t. f(t') \in Inv(l) \end{array}}{(l, \nu) \stackrel{t}{\rightarrow} (l, \nu')} \quad \texttt{Rule}_{\texttt{Time}}$$

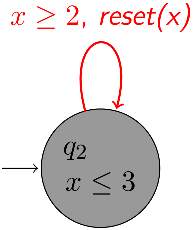
- execution step: $\rightarrow = \stackrel{a}{\rightarrow} \cup \stackrel{t}{\rightarrow}$
- run: $\sigma_0 \rightarrow \sigma_1 \rightarrow \sigma_2 \ldots$ with $\sigma_0 = (l_0, \nu_0) \in Init$ and $\nu_0 \in Inv(l_0)$
- reachability of a state: exists run leading to the state
- activities are represented in form of differential equations

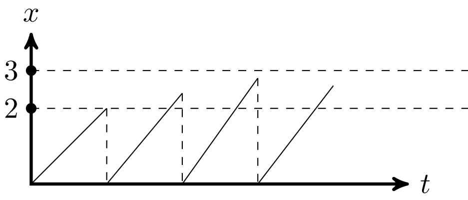
Example: Timed Automaton



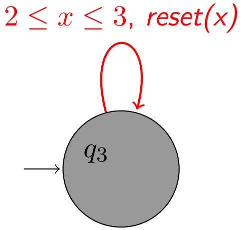


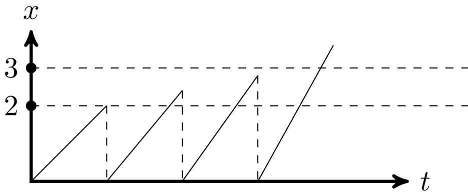
Example: Timed Automaton





Example: Timed Automaton





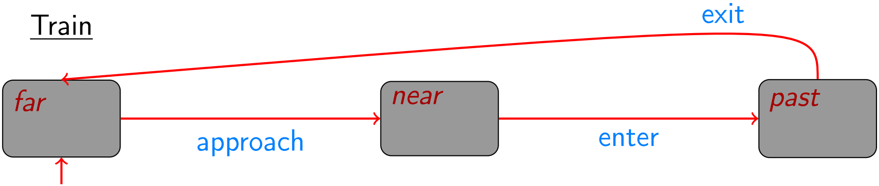
Parallel composition

Definition

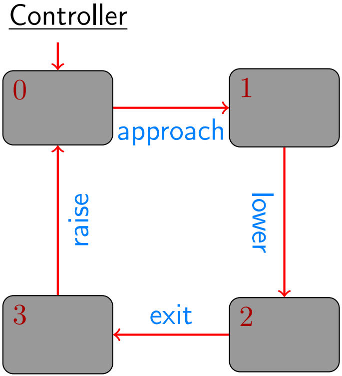
Let $\mathcal{H}_1 = (Loc_1, Var, Lab_1, Edge_1, Act_1, Inv_1, Init_1)$ and
$\mathcal{H}_2 = (Loc_2, Var, Lab_2, Edge_2, Act_2, Inv_2, Init_2)$
be two hybrid automata. The *product*
$\mathcal{H}_1 || \mathcal{H}_2 = (Loc_1 \times Loc_2, Var, Lab_1 \cup Lab_2, Edge, Act, Inv, Init)$ is the hybrid automaton with

- $Act(l_1, l_2) = Act_1(l_1) \cap Act_2(l_2)$ for all $(l_1, l_2) \in Loc$,
- $Inv(l_1, l_2) = Inv_1(l_1) \cap Inv_2(l_2)$ for all $(l_1, l_2) \in Loc$,
- $Init = \{((l_1, l_2), \nu) | (l_1, \nu) \in Init_1,\ (l_2, \nu) \in Init_2\}$, and
- $((l_1, l_2), a, \mu, (l_1', l_2')) \in Edge$ iff
  - $(l_1, a_1, \mu_1, l_1') \in Edge_1$ and $(l_2, a_2, \mu_2, l_2') \in Edge_2$, and
  - either $a_1 = a_2 = a$, or $a_1 = a \notin Lab_2$ and $a_2 = \tau$, or $a_1 = \tau$ and $a_2 = a \notin Lab_1$, and
  - $\mu = \mu_1 \cap \mu_2$.

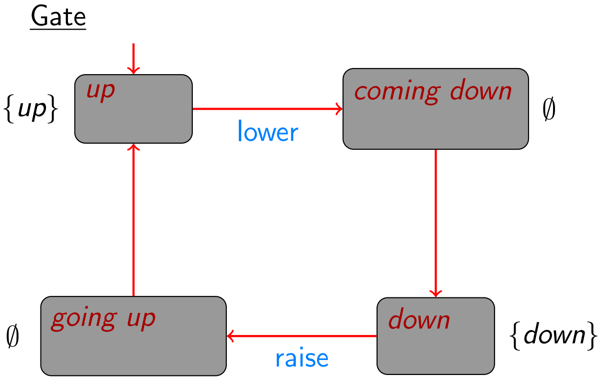
Simplified railroad crossing



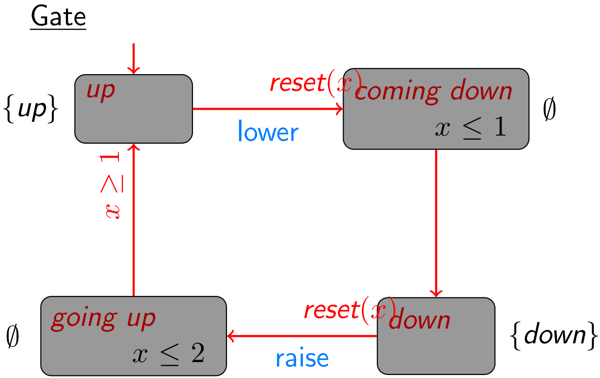
Simplified railroad crossing



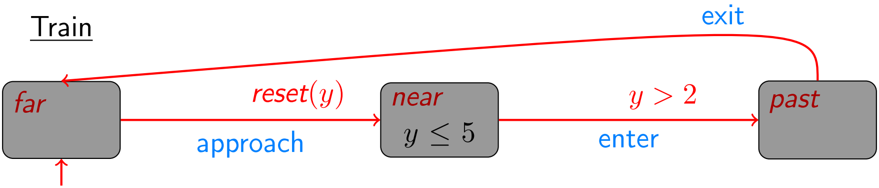
Simplified railroad crossing



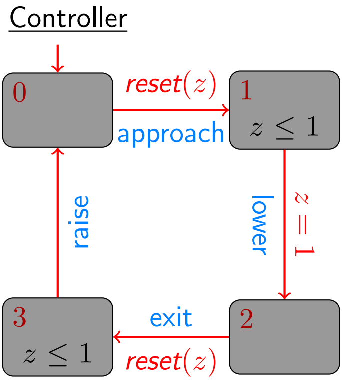
Simplified railroad crossing with time component

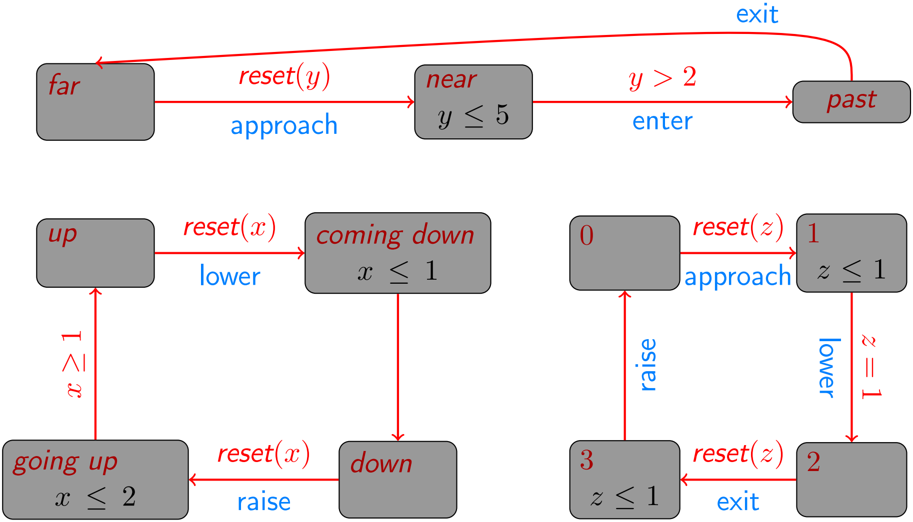


Simplified railroad crossing with time component



Simplified railroad crossing with time component




exit
far
reset(y)
approach
near
$y \leq 5$
$y > 2$
enter
past
up
reset(x)
lower
coming down
$x \leq 1$
$x \geq 1$
going up
$x \leq 2$
reset(x)
raise
down
0
reset(z)
approach
1
$z \leq 1$
raise
lower
$z = 1$
3
$z \leq 1$
reset(z)
exit
2