Systems and Automata

Stefan Ratschan

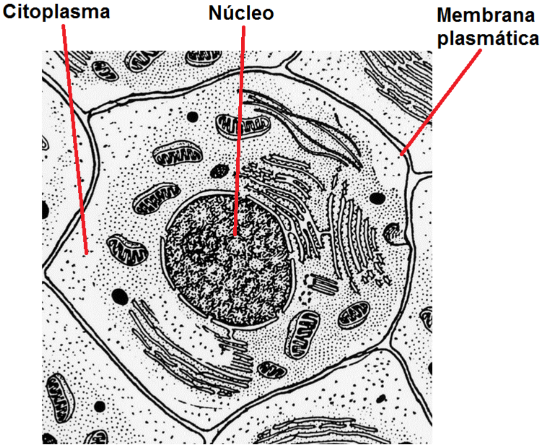
Katedra číslicového návrhu
Fakulta informačních technologií
České vysoké učení technické v Praze

Evropský sociální fond Praha & EU: Investujeme do vaší budoucnosti

Big Question

How can we describe and computationally analyze complex systems?

How do Complex Systems Look Like?



Observation:

Systems usually consist of components that influence each other.

(each component is again a system).

Modeling Communication of Systems

Example: printer with
input Postscript
output in $\{ok, problem\}$

Example: quadrocopter with
input set $\{nil, start\}$ and output set $\{free, scheduled, mission\}$.

Common characteristics?

- Certain set of symbols
- Discrete steps, continuing without end

How to model this?
Fix an arbitrary set S whose elements we call *symbols*.

A *(discrete time) signal* over S is
an infinite sequence of symbols (i.e., $\mathbb{N}_0 \to S$).

time

We write Σ_S for the set of signals over S
(e.g., $\Sigma_{\{nil, start\}}$, $\Sigma_{\{free, scheduled, misssion\}}$)

How Can We Describe the Behavior of Systems?

Example:

Quadrocopter (one possibility):

- Input signal: arbitrary signal from $\Sigma_{\{nil, start\}}$
- Output signal: Signal from $\Sigma_{\{free, scheduled, mission\}}$, s.t.
  input *start* results in output *mission* within at most 10 steps

Example:

Input: $(nil, nil, start, nil, nil, nil, nil, \dots)$

Output: $(free, free, free, free, free, scheduled, mission, \dots)$

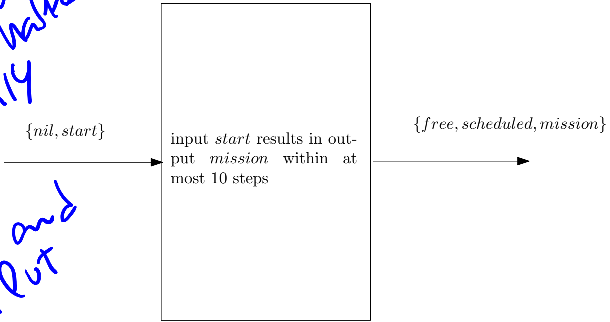
Further output for same input:
$(free, free, free, free, free, free, mission, \dots)$

Several outputs corresponding to one input!

Description of behavior from outside (black-box)

Black-box Description of Systems: Graphically

Nobody knows what happes inside only



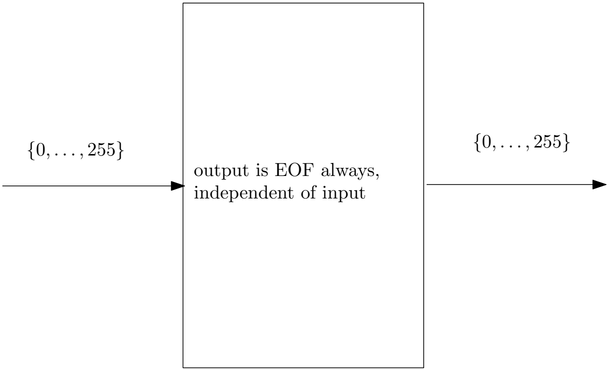
input and output

Example:

/dev/null

- Input signal: arbitrary string
- Output signal: EOF EOF EOF ...

Example:



$\{0, \ldots, 255\}$
output is EOF always, independent of input
$\{0, \ldots, 255\}$

Comparison: Classical Algorithms

For classical algorithm, I/O specifications relate input to output

Example:

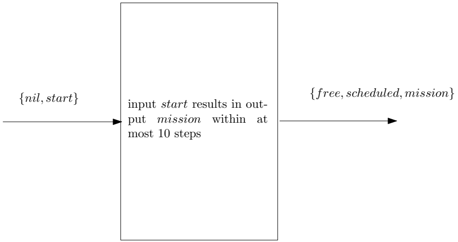
- Input: array a
- Output: array b that contains the same elements as a, but is sorted

Difference:

Traditional algorithms have input and output of finite length

Here we have *reactive systems*, and hence
input and output of infinite length (i.e., signals)

How to Formalize This?



Input: $(nil, nil, start, nil, nil, nil, nil, \dots)$

Output: $(free, free, free, free, free, scheduled, mission, \dots)$

Further output for same input:

$(free, free, free, free, free, free, mission, \dots)$

For each input signal, we have certain possible output signals ...

Certain input/output pairs are possible/allowed, others not ...

Black-box Description of Systems: Formalization

General principle: relation between input signals and output signals, that is:

A *(discrete time) system* with input set I and output set O is
a relation between signals over I and signals over O, that is
a subset of $\Sigma_I \times \Sigma_O$.

If (i, o) is an element of this subset,
then we also say that (i, o) is a *behavior* of the system.

Formalization of Examples

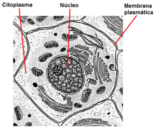
Quadrocopter:

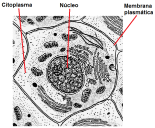
$$\{(i,o) \in \Sigma_{\{nil,start\}} \times \Sigma_{\{free,scheduled,mission\}} \mid \\ \forall t \in \mathbb{N}_0 \,.\, i(t) = start \Rightarrow \exists d \in \{0,\dots,10\} \,.\, o(t+d) = mission\}$$

/dev/null

$$\{(i,o) \in \Sigma_{\{0,\dots,255\}} \times \Sigma_{\{0,\dots,255\}} \mid \forall t \in \mathbb{N}_0 \,.\, o(t) = EOF\}$$

System vs. System





Attention: Such a “system” is really a model!

To avoid confusion, we will also use the term “real-world system”.

System Refinement

$$\{(i, o) \in \Sigma_{\{nil,start\}} \times \Sigma_{\{free,scheduled,mission\}} \mid \forall t \in \mathbb{N}_0 \,.\, i(t) = start \Rightarrow \exists d \in \{0, \dots, 10\} \,.\, o(t + d) = mission\}$$

Upper bound! make it stricter?

$$\{(i, o) \in \Sigma_{\{nil,start\}} \times \Sigma_{\{free,scheduled,mission\}} \mid \forall t \in \mathbb{N}_0 \,.\, i(t) = start \Rightarrow \exists d \in \{1, 2\} \,.\, o(t + d) = mission\}$$

System $\mathcal{S}_1$ is a *refinement* of system $\mathcal{S}_2$ iff $\mathcal{S}_1 \subseteq \mathcal{S}_2$

One system is a refinement of another if it allows fewer behaviours

Further Examples from Computer Science

Stream I/O of lazy functional programming languages: $\{0, \ldots, 255\}$

Exact real arithmetic: $\{0, \ldots, 9\}$

Communication of computational elements over network

Halting problem as a system: $\{(i, o) \in \Sigma_{\{0,\ldots,255\}} \times \Sigma_{\{0,1\}} \mid$

- i is a computer program of length l padded with blanks,
- $o(l+1) = 1$, if program from input always terminates, 0, otherwise. $\}$

Cannot be algorithmically implemented

$$\{(i, o) \mid o(0) = 1 \text{ iff } \mathbf{P} = \mathbf{NP}\}???$$

Can be implemented, but implementation unknown

System Properties

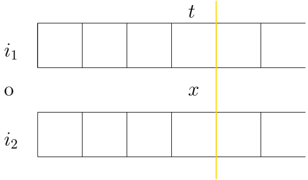
Let us assume a discrete time system $\mathcal{S}$
with input set I and output set O.

Example: $\mathcal{S} = \emptyset$?

$\mathcal{S}$ is *receptive* iff For every input, There is a output
for all $i \in \Sigma_I$ there is $o \in \Sigma_O$ s.t. $(i, o) \in \mathcal{S}$.

Example: $\mathcal{S} = \{(i, o) \mid \forall t \in \mathbb{N}_0 \,.\, o(t) = i(t+1)\}$?

$\mathcal{S}$ is *causal* iff
for all $i_1, i_2 \in \Sigma_I$, $x \in O$, $t \in \mathbb{N}_0$ such that
for all $t' \in \{0, \dots, t\} \,.\, i_1(t') = i_2(t')$,
there is $o \in \Sigma_O$ s.t. $(i_1, o) \in \mathcal{S}$, $o(t) = x$
iff
there is $o \in \Sigma_O$ s.t. $(i_2, o) \in \mathcal{S}$, $o(t) = x$



Intuition: output at given time is determined by input up to that time.

Usage of non-causal systems: audio signal processing

System Properties

$\mathcal{S}$ is *deterministic* iff
for all $i \in \Sigma_I$ there is precisely one $o \in \Sigma_O$ s.t. $(i, o) \in \mathcal{S}$.

In such a case the system can be viewed as a function
(instead of a relation)

Non-determinism: under-specification (leaving open details), uncertainty

Example: $\{(i, o) \mid \forall t \in \mathbb{N}_0 \,.\, o(t)$ is a prime divisor of $i(t)\}$

$\mathcal{S}$ is *memory-less* iff
there is $R \subseteq I \times O$ s.t. for all $(i, o) \in \Sigma_I \times \Sigma_O$,
$(i, o) \in \mathcal{S}$ iff for all t, $(i(t), o(t)) \in R$.

Intuition: output at given time is determined only by input at this time.

Every memory-less system is causal!

Philosophical Questions

Do non-causal real-world systems exist?

In other words: Can the present depend on the future?

Examples: oracle (non-causal), weather-forecast (causal)

Do non-deterministic real-world systems exist?

In other words: Does randomness exists?

In the mathematical world:
  Yes. Non-deterministic and non-causal models are very useful!

In the real world: answer unknown (experiments are not repeatable)
  Classical physics is a deterministic model,
  quantum physics a non-deterministic one

For Us

- Non-determinism very useful
- We will use certain memory-less systems
- Models of the real world are always receptive and causal,
  if not, this is a bug.

Let Us Recapitulate

How can we describe and computationally analyze complex systems?

Partial answer based on the observation that
complex system usual consist of sub-systems:

We can model systems as black-boxes that produce output from input

Model Based Design (cf. Lecture 1)

Hierarchy of System Models

- General requirements:
  "Train should follow signalling standard for Czech tracks"
- More detailed requirements:
  "Every stop signal should be acknowledged within 2 seconds"
- High level operational models,
  e.g., finite state automaton, Scilab demo
- VHDL, C program
- Circuit design, machine code
- ...

Up to now: formalization of top levels (requirements)

For next level: How to implement/simulate such (discrete time) systems on computers?

Goal of model based design: find bugs early and automatically.

Implementation in Full-Fledged Programming Languages

Example: C program that reads from stdin and writes to stdout

There are special programming languages for this
(*synchronous reactive programming*): Esterel, Lustre etc.

Part of SCADE system:
`http://www.esterel-technologies.com/`

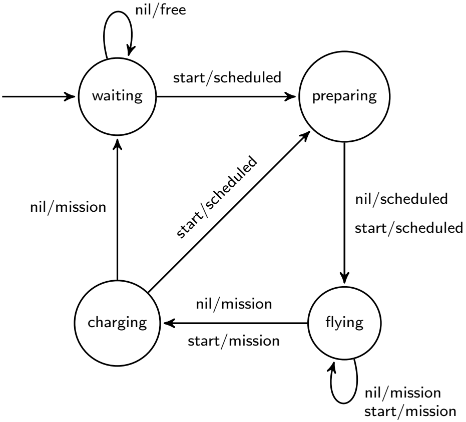
Problems?

- We have to decide on data structures, memory management, fight with complex syntax
- While we can test such implementations, more reliable automatic analysis in full generality impossible

White Box Description of Reactive Systems

Goal: as simple as possible programming language for reactive systems

Example: look inside of the system



Internal state: waiting, preparing, charging, flying.

Rule how state evolves and relates input to output

Which system described?

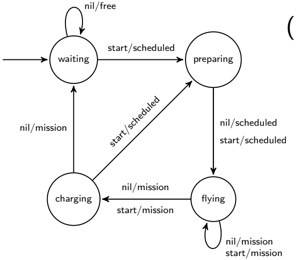
What is this?

Automaton

Discrete Time Automata

A *(discrete time) automaton* is a quintuple (S, S_0, I, O, R), where

- S is a set (*state space*) whose elements we call *states*
- $S_0 \subseteq S$ (set of *initial states*), s.t. $S_0 \neq \emptyset$
- I is a set whose elements we call *inputs*
- O is a set whose element we call *outputs*
- $R \subseteq I \times S \times S \times O$ (*transition relation*) s.t.
  for all $i \in I$, $s \in S$, there is $s' \in S$, $o \in O$ s.t. $(i, s, s', o) \in R$



$$
\begin{aligned}
(&\{waiting, preparing, flying, charging\},\\
&\{waiting\},\\
&\{nil, start\},\\
&\{free, scheduled, mission\},\\
&\{(nil, waiting, waiting, free),\\
&\quad(start, waiting, preparing, scheduled),\\
&\quad(start, preparing, flying, scheduled),\\
&\quad \ldots \qquad\qquad\qquad \}\qquad)
\end{aligned}
$$

The sets do not have to be finite, but often they are ...

Discrete Time Automata

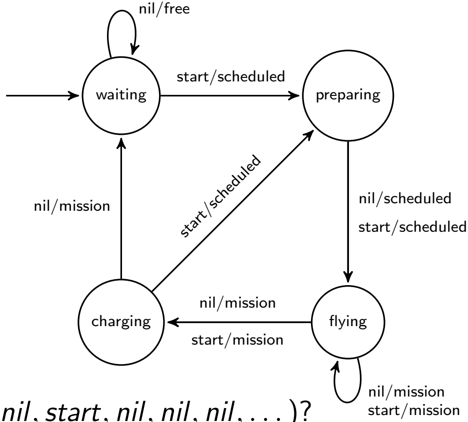
A *(discrete time) automaton* is a quintuple (S, S_0, I, O, R), where

- S is a set (*state space*) whose elements we call *states*
- $S_0 \subseteq S$ (set of *initial states*)
- I is a set whose elements we call *inputs*
- O is a set whose element we call *outputs*
- $R \subseteq I \times S \times S \times O$ (*transition relation*) s.t.
  for all $i \in I$, $s \in S$, there is $s' \in S$, $o \in O$ s.t. $(i, s, s', o) \in R$

Difference to classical automata? no terminal state

Various equivalent variants (e.g., transitions based on sets, separate output function/relation), extensions, and names (transducers, I/O automata)

Corresponding System



Output for input $(nil, start, nil, nil, nil, \ldots)$?

$$waiting \xrightarrow{nil/free} waiting \xrightarrow{start/scheduled} preparing \xrightarrow{nil/scheduled}$$
$$flying \xrightarrow{nil/mission} flying \xrightarrow{nil/mission} charging \ldots$$

$$(free, scheduled, scheduled, mission, mission, \ldots)$$

Corresponding System

Given an automaton (S, S_0, I, O, R) the pair of signals $(i, o) \in \Sigma_I \times \Sigma_O$ is a *behavior* of the automaton iff there is an $s \in \Sigma_S$ s.t.

- $s(0) \in S_0$,
- for all $t \in \{0, 1, \dots\}$, $(i(t), s(t), s(t+1), o(t)) \in R$.

In this case we also write $s(0) \xrightarrow{i(0)/o(0)} s(1) \xrightarrow{i(1)/o(1)} s(2) \dots$

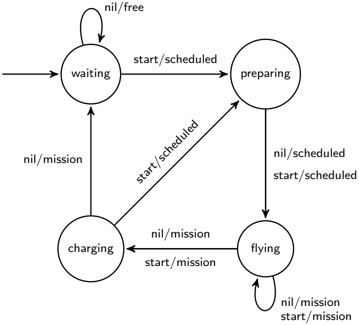
An automaton T *represents* the system

$$[\![T]\!] := \{(i, o) \in \Sigma_I \times \Sigma_O \mid (i, o) \text{ is a behavior of } T\}$$

Representation vs. Refinement

$$\{(i, o) \in \Sigma_{\{nil, start\}} \times \Sigma_{\{free, scheduled, mission\}} \mid \text{input } start \text{ results in output } mission \text{ within at most 10 steps}\}$$

$$\{(i, o) \in \Sigma_{\{nil, start\}} \times \Sigma_{\{free, scheduled, mission\}} \mid \forall t \in \mathbb{N}_0 \,.\, i(t) = start \Rightarrow \exists d \in \{0, \ldots, 10\} \,.\, o(t + d) = mission\}$$



Represents above system?

Never needs 10 steps

Does not represent the system!

But represents refinement!

Representing Systems Using Automata

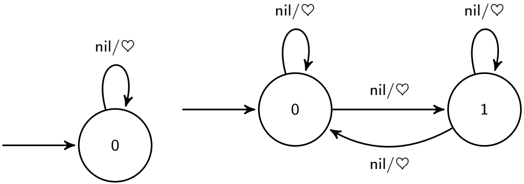
For a given system, is there an automaton representing it?

Example: $\{(i, o) \in \Sigma_{\{a,b,c\}} \times \Sigma_{\mathbb{N}_0} \mid \forall t \,.\, o(t) = |\{t' \mid t' \leq t, i(t') = a\}| \}$

no finite automaton can implement this

May there be more than one automaton representing a given system?

Example: $\{(i, o) \in \Sigma_{\{nil\}} \times \Sigma_{\{\heartsuit\}} \mid \forall t \,.\, o(t) = \heartsuit\}$



Two automata are *equivalent* iff they represent the same system.

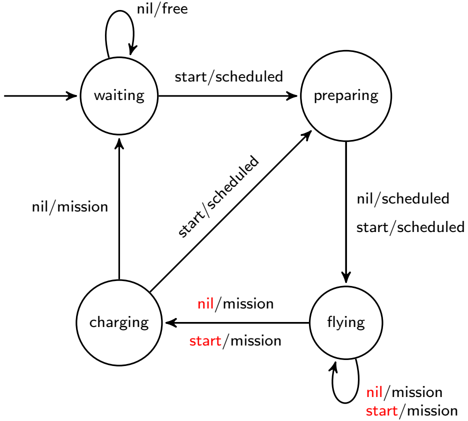
Minimal automaton? (see any textbook on automata theory, BI-AAG, with outputs [Mohri, 2000])

Properties of Automata

The system represented by an automaton is always receptive and causal!

Is it always deterministic? No

This means: For a given input, unique output?



Deterministic Automata

An automaton is *deterministic* iff

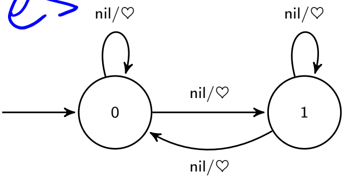
- $|S_0| = 1$
- for all $i \in I$, $s \in S$,
  there is precisely one $s' \in S$, $o \in O$ s.t. $(i, s, s', o) \in R$

For every deterministic automaton,
the system it represents is also deterministic

The other way round:
Can a non-deterministic automaton represent a deterministic system?

Example:

Yes



memory-less?

Variables: Extended State Machines

Example: Traffic lights:

after button is pressed, wait for 30 seconds, then switch to green.

- State space: $\{(s, x) \mid s \in \{\mathit{default}, \mathit{count}\}, x \in \{0, \ldots, 30\}\}$
- Initial states: $\{(\mathit{default}, 0)\}$
- Input: $\{\mathit{nil}, \mathit{button}\}$
- Output: $\{\mathit{green}, \mathit{red}\}$
- Transitions:

$$\left\{\begin{array}{l} (\mathit{nil}, (\mathit{default}, 0), (\mathit{default}, 0), \mathit{red}), \\ (\mathit{button}, (\mathit{default}, 0), (\mathit{count}, 30), \mathit{red}), \\ (\mathit{nil}, (\mathit{count}, 30), (\mathit{count}, 29), \mathit{red}), \\ (\mathit{nil}, (\mathit{count}, 29), (\mathit{count}, 28), \mathit{red}) \\ \ldots \end{array}\right\}$$

Can we write the transitions in a more compact way?

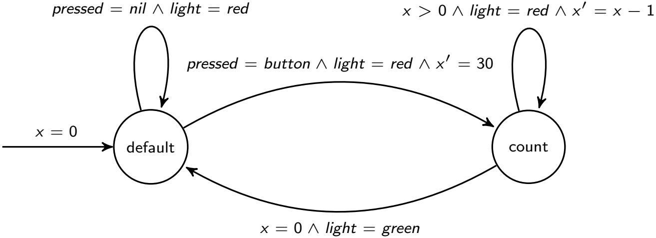
Writing Transitions

$$\left\{\begin{array}{l} (nil, (default, 0), (default, 0), red), \\ (button, (default, 0), (count, 30), red), \\ (nil, (count, 30), (count, 29), red), \\ (nil, (count, 29), (count, 28), red) \\ \ldots \end{array}\right\}$$

$$\Big\{(pressed, ((l, x), (l', x')), light) \mid$$
$$\left.\begin{array}{l} [l = default \wedge l' = default \wedge pressed = nil \wedge light = red] \vee \\ [l = default \wedge l' = count \wedge pressed = button \wedge light = red \wedge x' = 30] \vee \\ [l = count \wedge l' = count \wedge x > 0 \wedge light = red \wedge x' = x - 1] \vee \\ [l = count \wedge l' = default \wedge x = 0 \wedge light = green] \end{array}\right\}$$

Extended State Machines: Graphical Notation

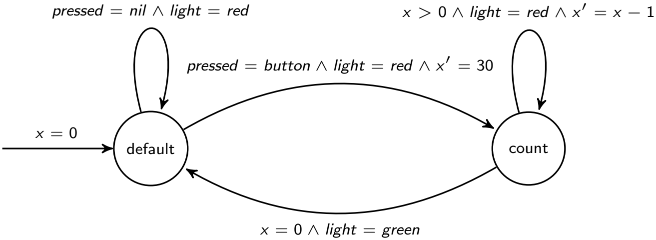
- Input variables: $pressed \in \{nil, button\}$
- Output variables: $light \in \{green, red\}$
- State variables: $x \in \{0, \ldots, 30\}$

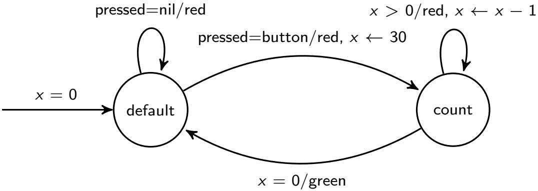


Here: only one input/output/state variable

In general: several ones, where n variables represent n-tuples

Common Notation

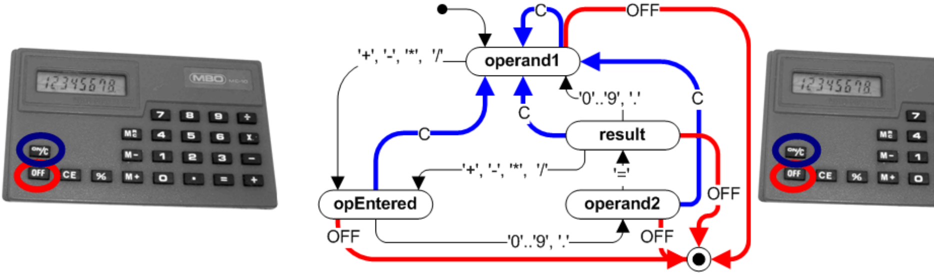




Terminology:

- *Locations*: graph vertices (only part of state space)
- *Guards*: conditions on input variables and state variables
- *Updates*: changes of state variables
  (variables without updates stay unchanged)

Hierarchical Automata



Superstate vs. sub-state

Recursive hiding versus modeling of details (cf. abstraction)

Basis for formalism Statecharts [Harel, 1987].

Various variants and extensions

Especially UML State Machines

see Wikipedia "UML state machine"

Conclusion

Systems consist of communicating sub-systems

We can describe them

- from the outside (black box):
  Which inputs result in which outputs?
- from the inside (white box):
  How are outputs produced from inputs?

Corresponding main definitions:

- discrete-time system (a model)
- automaton

Comparison: classical algorithms

- I/O specification (relation between inputs and outputs):
- Implementation in a programming language

Theory of automata with inputs of infinite length: ω-automata

Next Lecture

Forms of interconnection and interaction between systems

Literature

Rajeev Alur. *Principles of Cyber-Physical Systems*. The MIT Press, 2015.

David Harel. Statecharts: a visual formalism for complex systems. *Science of Computer Programming*, 8(3):231 – 274, 1987.

Edward A. Lee and Pravin Varaiya. *Structure and Interpretation of Signals and Systems*. http://LeeVaraiya.org, 2011.

Mehryar Mohri. Minimization algorithms for sequential transducers. *Theoretical Computer Science*, 234(1–2):177 – 201, 2000. ISSN 0304-3975. doi: http://dx.doi.org/10.1016/S0304-3975(98)00115-7. URL http://www.sciencedirect.com/science/article/pii/S0304397598001157.