CIS 4930/6930: Principles of Cyber-Physical Systems

Chapter 3: Discrete Dynamics

Hao Zheng

Department of Computer Science and Engineering
University of South Florida

What is Modeling?

- Developing insight about a system through imitation.
- A **model** is an artifact that imitates the system of interest.
- A **mathematical model** is a model in the form of a set of definitions and mathematical formulas often represented using a **modeling language**.
- *Key point:* a modeling language has *formal semantics*.

What is Model-Based Design?

- Create a mathematical model of all the parts of the embedded system:
  - Physical world
  - Sensors and actuators
  - Hardware platform
  - Software
  - Network
  - Control system
- Construct the implementation from the model:
  - Construction may be automated, like a compiler.
  - More commonly, only portions are automatically constructed.

Key Modeling Issues for Embedded Systems

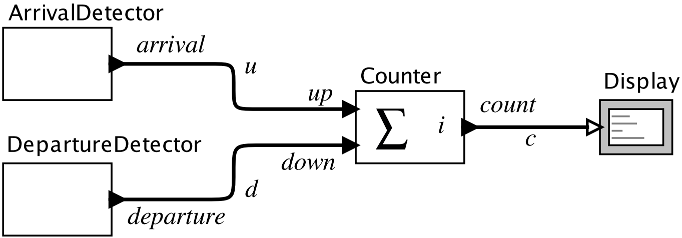
- Concurrency
- Time
- Dynamics: discrete and continuous

Modeling Techniques

- Models are abstractions of system dynamics (i.e., how things change over time):
  - Discrete dynamics - *finite-state machines* (FSMs)
  - Continuous dynamics - *ordinary differential equations* (ODEs)
  - Discrete & Continuous Dynamics - *Hybrid systems*

3.1 Discrete Systems

- Example: count the number of cars that enter and exit a parking garage:



Pure signal:

$\text{up} : \mathbb{R} \to \{\text{absent}, \text{present}\}$

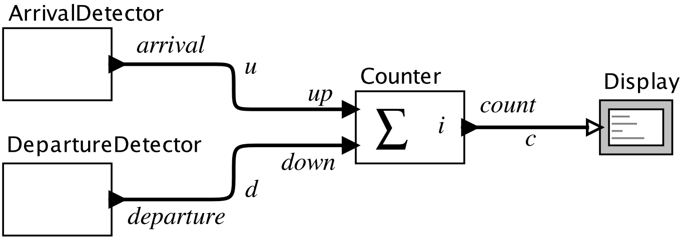
$\text{down} : \mathbb{R} \to \{\text{absent}, \text{present}\}$

Discrete actor: (def. section 2.2)

$\text{Counter} : (\mathbb{R} \to \{\text{absent}, \text{present}\})^{\{\text{up,down}\}} \to (\mathbb{R} \to \{\text{absent} \cup \mathbb{Z}\})$

Reaction

- Discrete dynamics: sequence of **reactions**.
- For any $t \in \mathbb{R}$ where $up(t) = present$ or $down(t) = present$ the Counter *reacts* by producing an output value in $\mathbb{Z}$ and changing its internal *state* - **event-triggered.**



$$\text{Counter} : (\mathbb{R} \to \{\text{absent}, \text{present}\})^{\{\text{up,down}\}} \to (\mathbb{R} \to \{\text{absent} \cup \mathbb{Z}\})$$

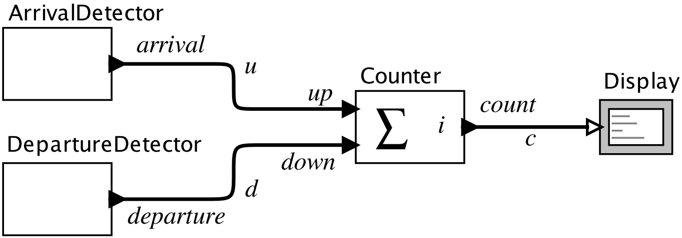
Inputs and Outputs at Reaction

- For $t \in \mathbb{R}$, the inputs are in a set:

$$\text{Inputs} \;=\; (\{\text{up}, \text{down}\} \to \{\text{absent}, \text{present}\})$$

- The outputs are in a set:

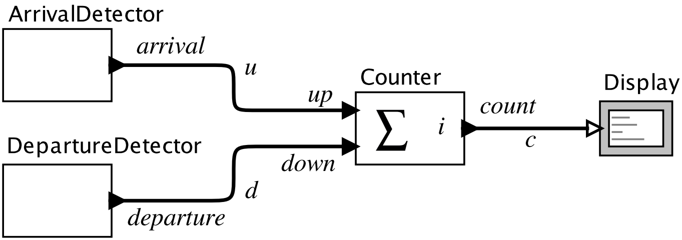
$$\text{Outputs} \;=\; (\{\text{count}\} \to \{\text{absent}\} \cup \mathbb{Z})$$



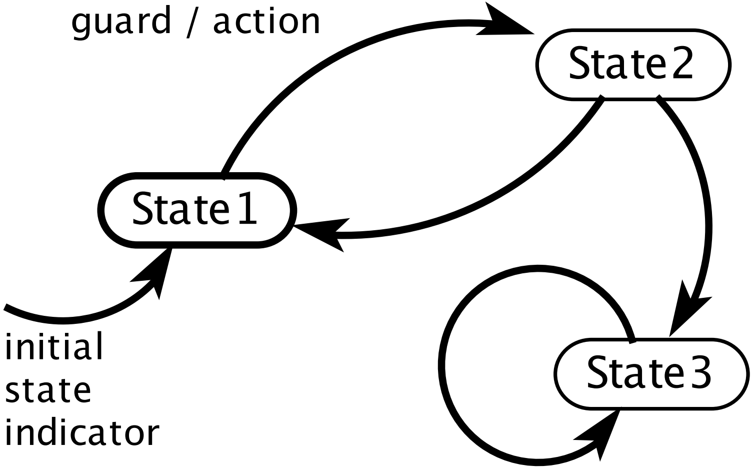
3.2 States

- A practical parking garage has a finite number, M, parking spaces, so the state space for the counter is:

$$\text{States} \quad = \quad \{0, 1, 2, \ldots, M\}$$

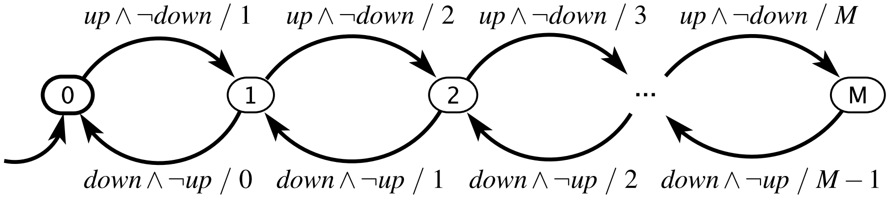


3.3 Finite State Machines (FSM): Transitions



3.3 FSM: Garage Counter

inputs: $up, down$: pure
output: $count : \{0, \cdots, M\}$



- Input is specified as *guard* using the shorthand:

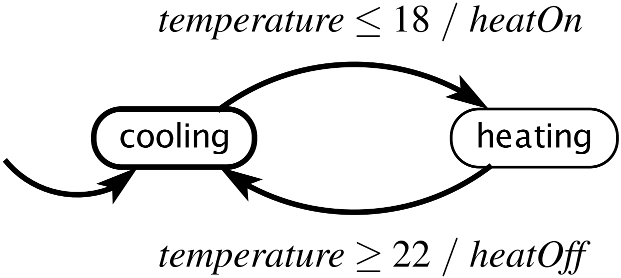
$$\text{up} \wedge \neg \text{down}$$

which means

$$(up = present \wedge down = absent)$$

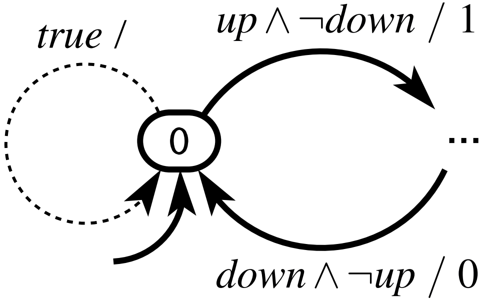
3.3 FSM: Thermostat

input: $temperature : \mathbb{R}$
outputs: $heatOn, heatOff$: pure



- **Hysteresis** is used in this example to prevent **chattering**.

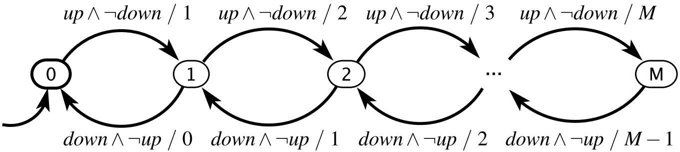
3.3.2 Default Transitions



- A default transition is enabled if no non-default transition is enabled and it either has no guard or the guard evaluates to true.
- When is the above default transition enabled?

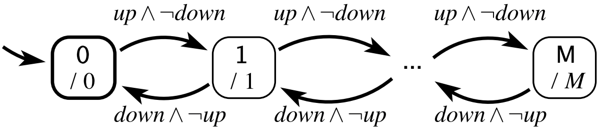
Meally Versus Moore Machines

inputs: *up*, *down* : pure
output: *count* : $\{0, \cdots, M\}$



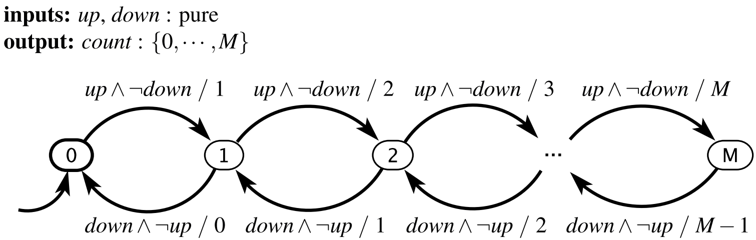
Mealy Machine

inputs: *up*, *down* : pure
output: *count*: $\{0, \cdots, M\}$



Moore Machine

Garage Counter Mathematical Model



Formally: (States, Inputs, Outputs, Update, InitialState), where:

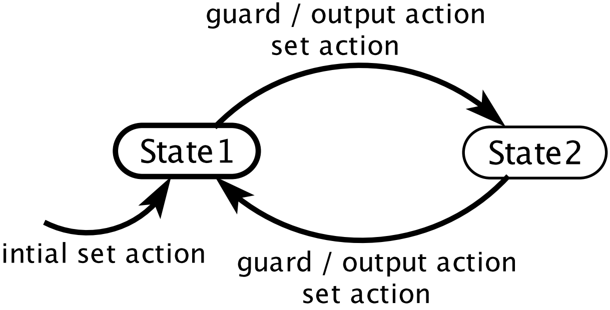
- $States = \{0, 1, 2, \dots, M\}$
- $Inputs = (\{up, down\} \rightarrow \{absent, \text{present}\})$
- $Outputs = (\{count\} \rightarrow \{0, 1, \dots, M, absent\})$
- $Update : States \times Inputs \rightarrow States \times Outputs$ (see above)
- $InitialState = 0$

3.3.4 Definitions

- **Receptiveness**: For any input values, some transition is enabled. Our structure together with the implicit default transition ensures that our FSMs are receptive.
- **Determinism**: In every state, for all input values, exactly one (possibly implicit) transition is enabled.

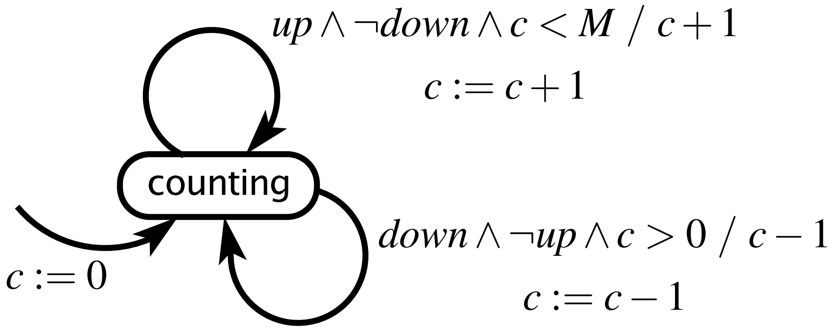
3.4 Extended State Machines

variable declaration(s)
input declaration(s)
output declaration(s)



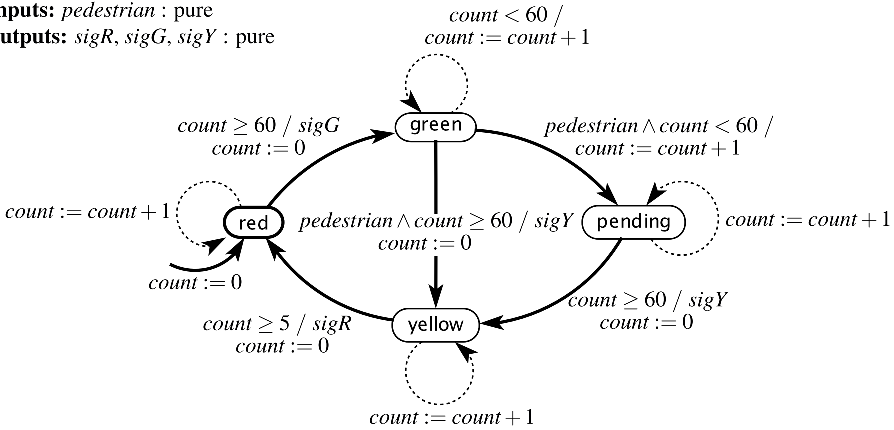
3.4 Extended FSM for the Garage Counter

variable: $c: \{0, \cdots, M\}$
inputs: $up, down$: pure
output: $count$: $\{0, \cdots, M\}$

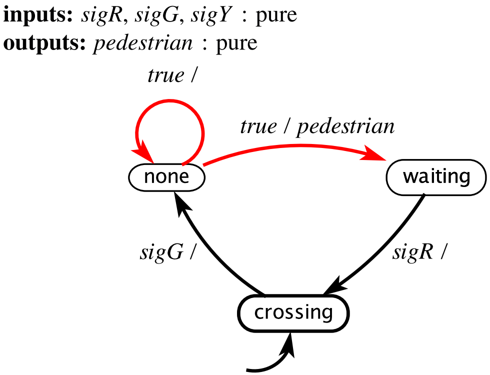


3.4 Extended FSM for Traffic Light Controller

variable: *count*: $\{0, \cdots, 60\}$
inputs: *pedestrian* : pure
outputs: *sigR*, *sigG*, *sigY* : pure

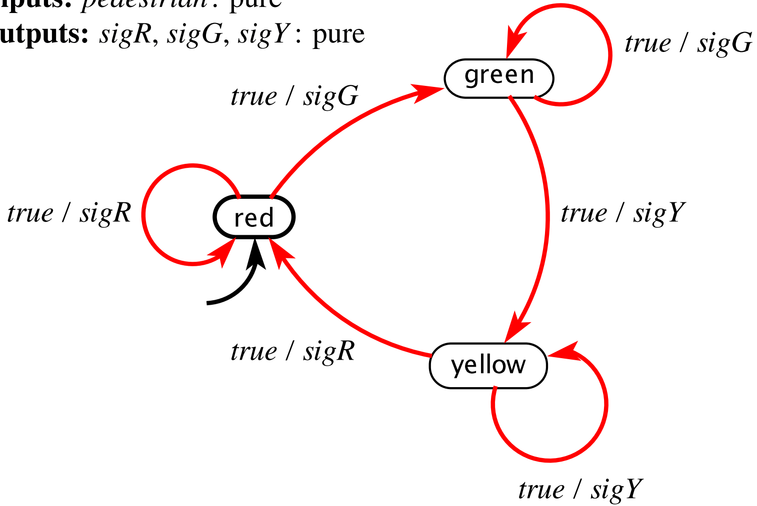


3.5 Non-deterministic FSM for Environment



- Model of the environment for the traffic light is abstracted using non-determinism.

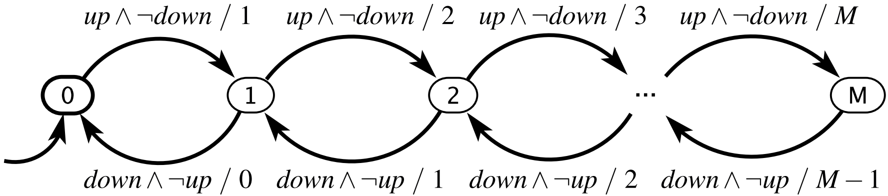
inputs: *pedestrian* : pure
outputs: *sigR*, *sigG*, *sigY* : pure



3.6 FSM Behaviors

- FSM behavior is a sequence of reactions.
- A *trace* is the record of inputs, states, and outputs in a behavior.

inputs: $up, down$: pure
output: $count : \{0, \cdots, M\}$

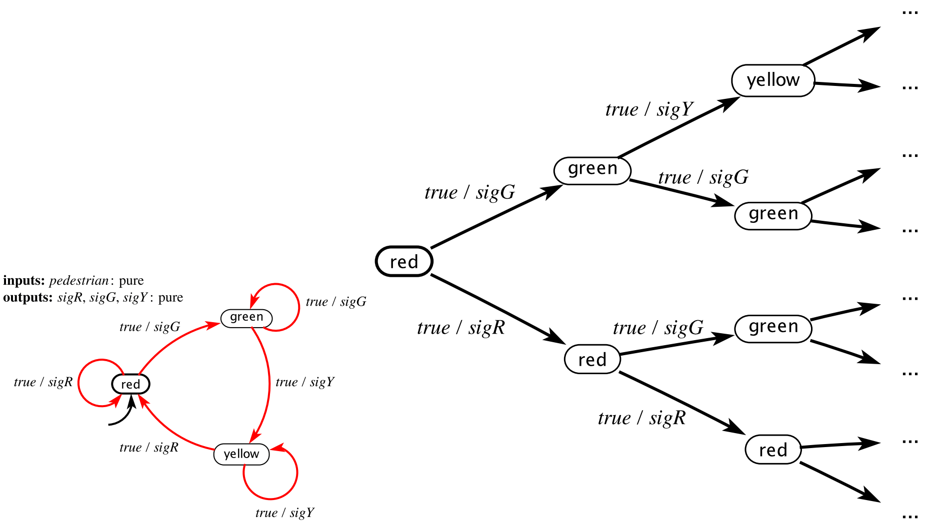


Input seqeuence
$s_{up} = (present, absent, present, absent, \ldots)$
$s_{down} = (present, absent, absent, present, \ldots)$

3.6 FSM Behaviors

- A **execution trace** is a sequence of values assigned to inputs, states, and outputs.
- A **observable trace** is a sequence of values assigned to inputs, and outputs.
- For a fixed input sequence:
  - A deterministic FSM exhibits a single behavior (trace).
  - A non-deterministic FSM exhibits a set of behaviors (traces) which can be visualized as a *computation tree*.

Computation Tree



Concluding Remarks

FSMs provide:

1. A way to represent the system for mathematical analysis, so that a computer program can manipulate it.
2. A way to model the environment of a system.
3. A way to represent what the system must do and must not do - (i.e., its specification).
4. A way to check whether the system satisfies its specification in its operating environment.
   - For example, using reachability analysis, one can determine that some unsafe state is not reachable.