Modeling: Discrete dynamics

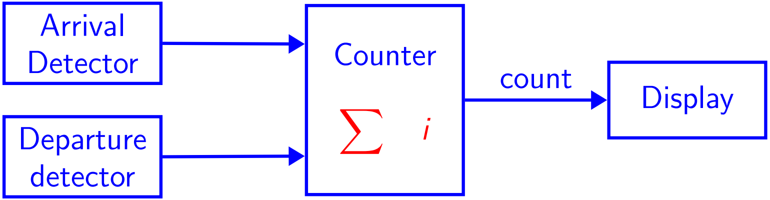
Arijit Mondal
Dept. of Computer Science & Engineering
Indian Institute of Technology Patna
arijit@iitp.ac.in

Introduction

- Embedded systems can include both discrete and continuous dynamics
- Continuous dynamics can be modeled by ordinary differential equation
- State machines are used to model discrete behavior of the systems
- A system operates in a sequence of discrete steps
- Example
  - Number of cars in a parking area

Car parking

- Arrival detector, departure detector



- Similar to integrator
- Input is not continuous $u : R \rightarrow \{absent, present\}$
  - Also known as pure signal

Event

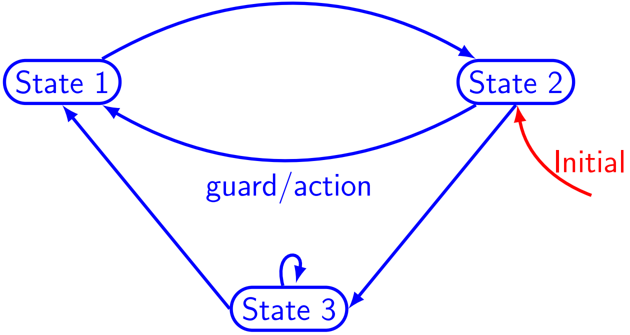
- Systems are **event triggered**
  - Sequence of steps known as reaction
- A particular reaction will observe the values of the inputs at a particular time and calculate output values for the same time
  - An actor has input ports $P = \{p_1, p_2, \ldots, p_N\}$
  - V_p denotes the type of p (values may be received)
  - At each reaction a variable can take $p \in V_p \cup \{absent\}$

Notion of state

- State of a system is its **condition** at a particular point of time
- State affects how the **system reacts to inputs**
- Integrator : discrete vs continuous
- Discrete modes with finite state space are called **finite state machine**

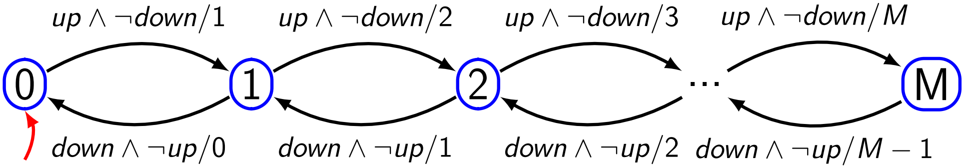
Finite State Machine

- A state machine is a model with discrete dynamics that maps valuations of the inputs to outputs where the map may depend on its current state



Finite State Machine: example

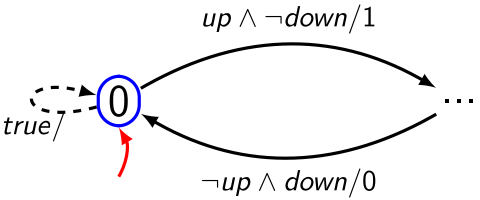
inputs: *up*, *down*: pure
outputs: *count*:$\{0, 1, \ldots, M\}$



Transition

- It governs the discrete dynamics of FSM
- Guard/Action
  - **Guard** determines whether the transition may take on a reaction
  - **Action** specifies the output for each reaction
- If p_1 and p_2 are inputs to FSM
  - *true* — transition is always enabled
  - p_1 — transition is enabled if p_1 is present
  - $\neg p_1$ — transition is enabled if p_1 is absent
  - $p_1 \wedge p_2$ — transition is enabled if both p_1 and p_2 are present
  - $p_1 \vee p_2$ — transition is enabled if either p_1 or p_2 are present

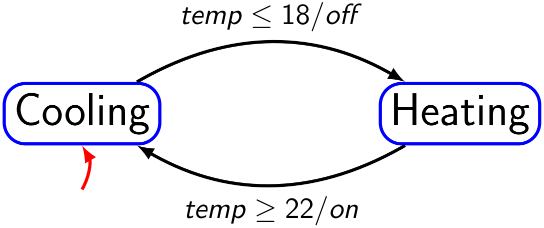
Default transition



Finite State Machine: example

inputs: *temp*: $\mathbb{R}$

outputs: *on*, *off* : pure



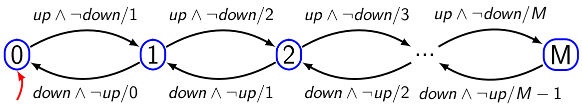
FSM Definition

- It is a tuple $\langle States, Inputs, Outputs, Update, InitialState \rangle$
- States — finite number of states
- Inputs — set of input valuations
- Outputs — set of output valuations
- Update — $States \times Inputs \to States \times Outputs$, mapping a state and input valuation to a next state and a output valuation
- InitialState — start state

FSM example

- States $= \{0, 1, 2, \ldots, M\}$
- Inputs $= \{up, down\} \to \{present, absent\}$
- Outputs $= \{count\} \to \{0, 1, 2, \ldots, M\}$
- InitialState $= 0$
- $$update(s, i) = \begin{cases} (s+1, s+1) \text{ if } s < M \wedge up = present \wedge down = absent \\ (s-1, s-1) \text{ if } s > 0 \wedge up = absent \wedge down = present \\ (s, absent) \text{ otherwise} \end{cases}$$

inputs: *up*, *down*: pure
outputs: *count*: $\{0, 1, \ldots, M\}$



A few terminologies

- **Determinacy** — If for each state there is at most one transition enabled by each input value
  - Update function is not one to many mapping
  - Same input will produce same output always
- **Receptiveness** — If for each state there is at least one transition possible on each input symbol
  - FSM is receptive as 'update' is a function, not a partial function
- **Chattering** — A system oscillates between two states rapidly
- **Stuttering** — A system remains in the state due to absence of inputs and outputs
- **Hysteresis** — Dependence of the state of a system on its history.

Mealy vs Moore machine

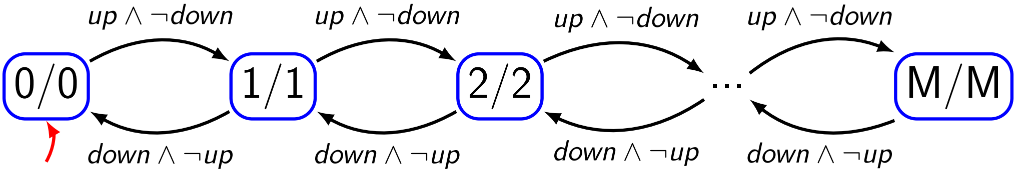
- **Mealy machine**
  - Named after George Mealy
  - Characterized by producing outputs when a transition is taken
- **Moore machine**
  - Named after Edward Moore
  - Produces the output when the machine is in a state
  - Output is function of state only
  - Strictly causal

- A Mealy machine can be converted into Moore machine
- A Moore machine can be converted into Mealy machine
- Mealy machine is **preferred** because of compactness and output is instantaneous with respect to inputs

Moore machine: example

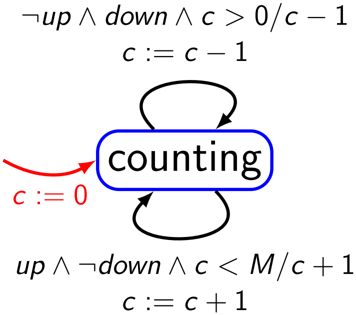
inputs: *up*, *down*: pure

outputs: *count*: $\{0, 1, \ldots, M\}$



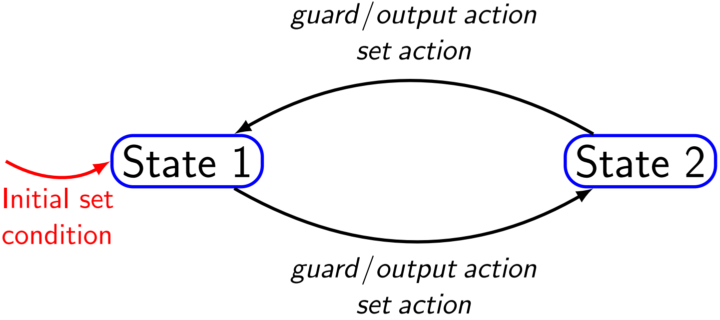
Extended FSM

variable: $c : \{0, 1, \ldots, M\}$
inputs: *up*, *down*: pure
outputs: $count$:$\{0, 1, \ldots, M\}$



Extended FSM

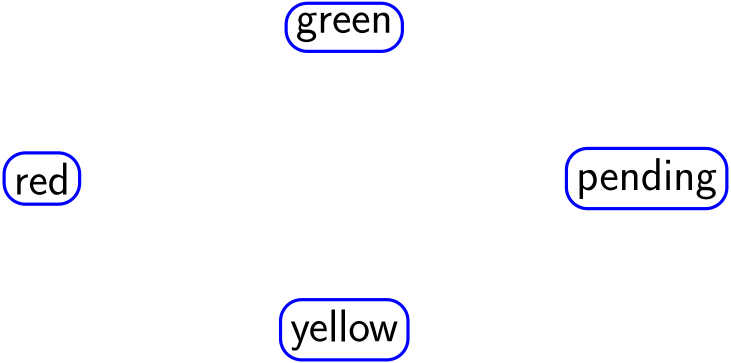
variable declaration
input declaration
output declaration



Example: pedestrian crosswalk

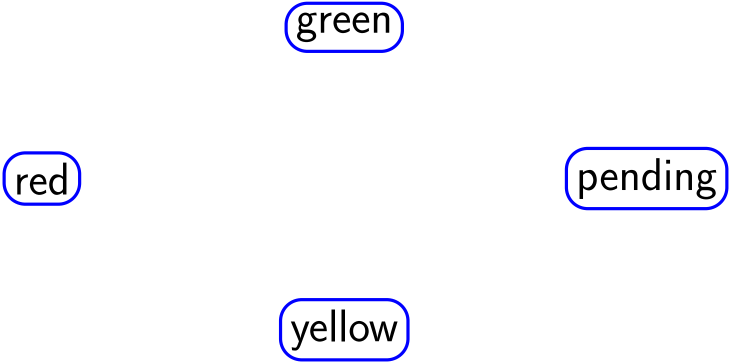
- It starts with red
- It moves to green after 60 seconds
- It will remain in green if there is no pedestrian
- If the light goes to green, then it remains there at least for 60 seconds
- If there is a pedestrian, light becomes yellow if it has been green for more than 60 seconds
- The yellow light will remain for 5 seconds before it turns to red

Example: pedestrian crosswalk



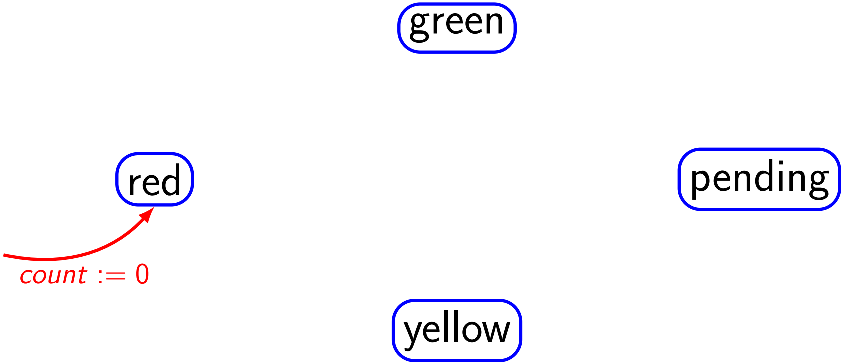
Example: pedestrian crosswalk

variable: $count : \{0, 1, \ldots, 60\}$
input: $pedestrian : pure$
output: $sigY, sigG, sigR : pure$



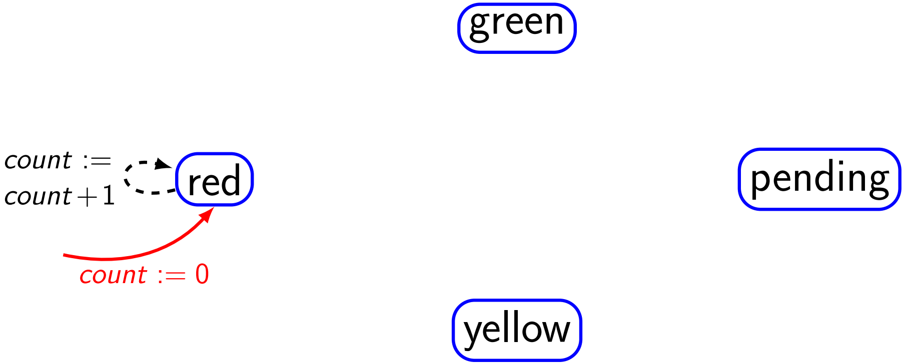
Example: pedestrian crosswalk

variable: $\mathit{count} : \{0, 1, \ldots, 60\}$
input: $\mathit{pedestrian} : \mathit{pure}$
output: $\mathit{sigY}, \mathit{sigG}, \mathit{sigR} : \mathit{pure}$



Example: pedestrian crosswalk

variable: $\mathit{count} : \{0, 1, \ldots, 60\}$
input: $\mathit{pedestrian} : \mathit{pure}$
output: $\mathit{sigY}, \mathit{sigG}, \mathit{sigR} : \mathit{pure}$

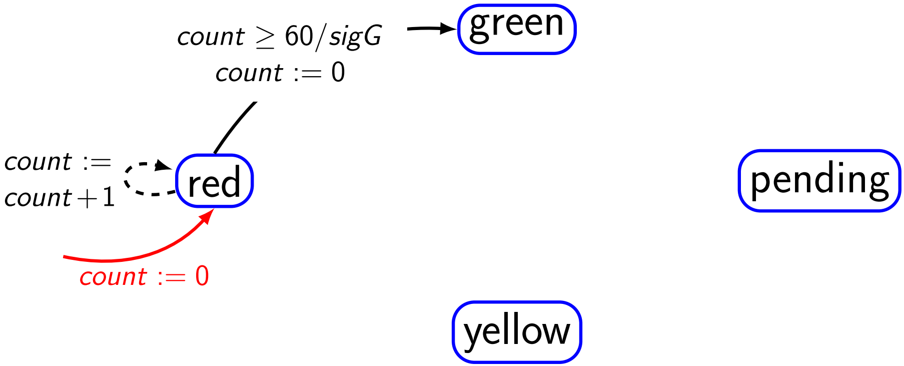


Example: pedestrian crosswalk

variable: $count : \{0, 1, \ldots, 60\}$

input: $pedestrian : pure$

output: $sigY, sigG, sigR : pure$

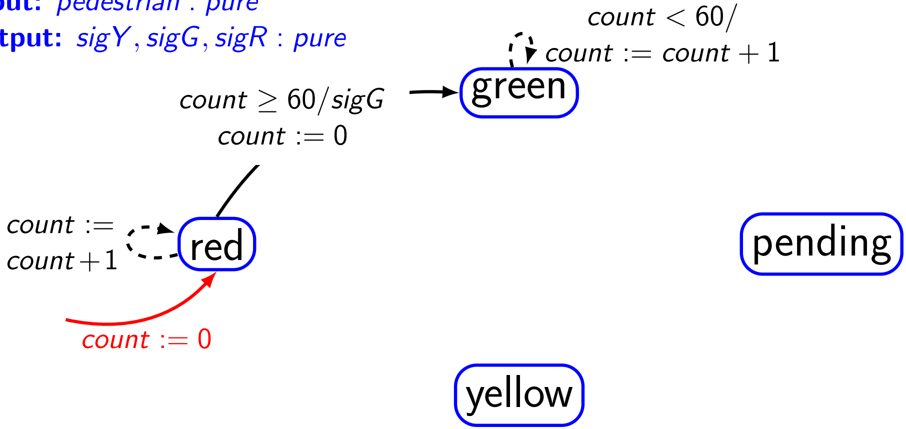


Example: pedestrian crosswalk

variable: $\mathit{count} : \{0, 1, \ldots, 60\}$

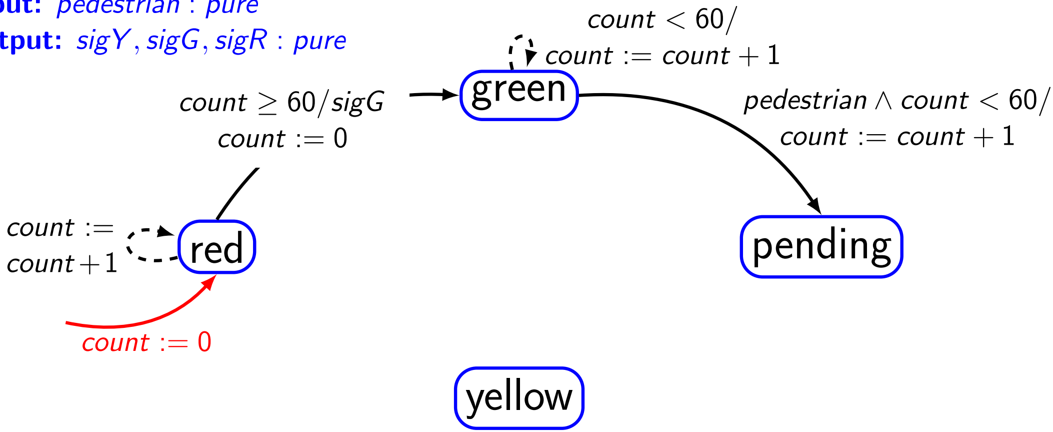
input: $\mathit{pedestrian} : \mathit{pure}$

output: $\mathit{sigY}, \mathit{sigG}, \mathit{sigR} : \mathit{pure}$



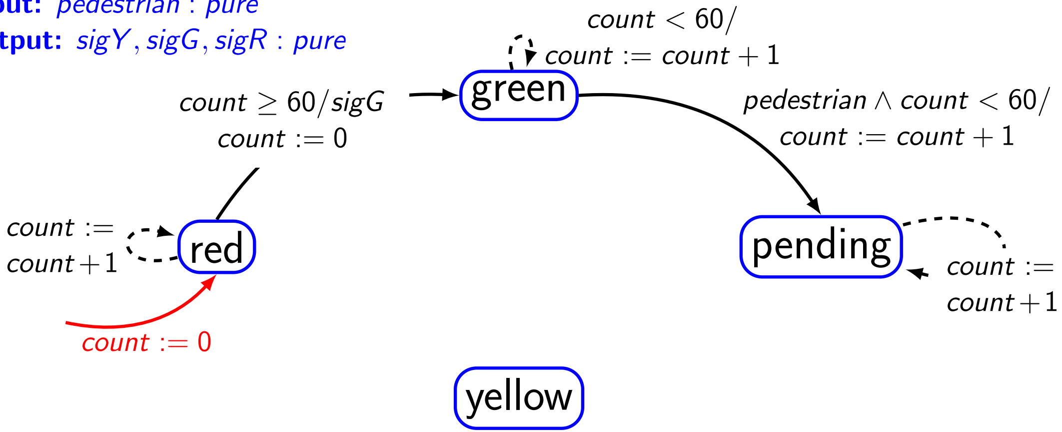
Example: pedestrian crosswalk

variable: $count : \{0, 1, \ldots, 60\}$
input: $pedestrian : pure$
output: $sigY, sigG, sigR : pure$



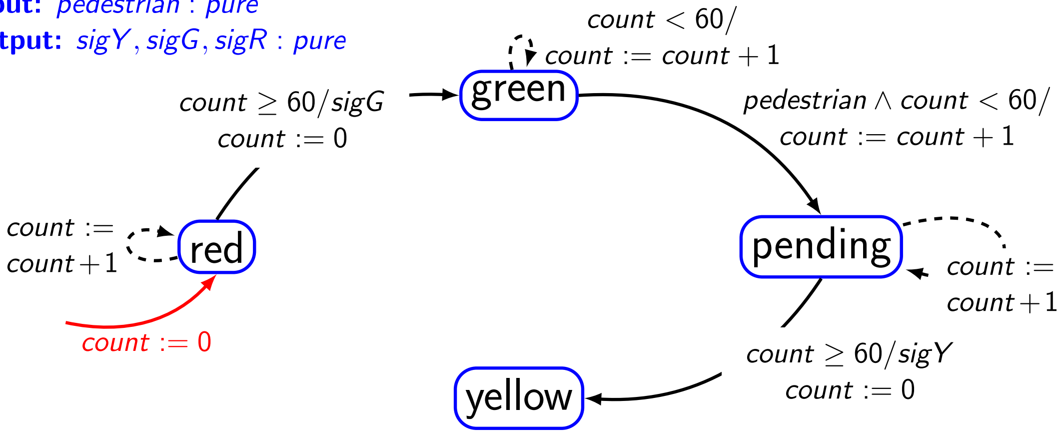
Example: pedestrian crosswalk

variable: *count* : $\{0, 1, \ldots, 60\}$
input: *pedestrian* : *pure*
output: *sigY*, *sigG*, *sigR* : *pure*



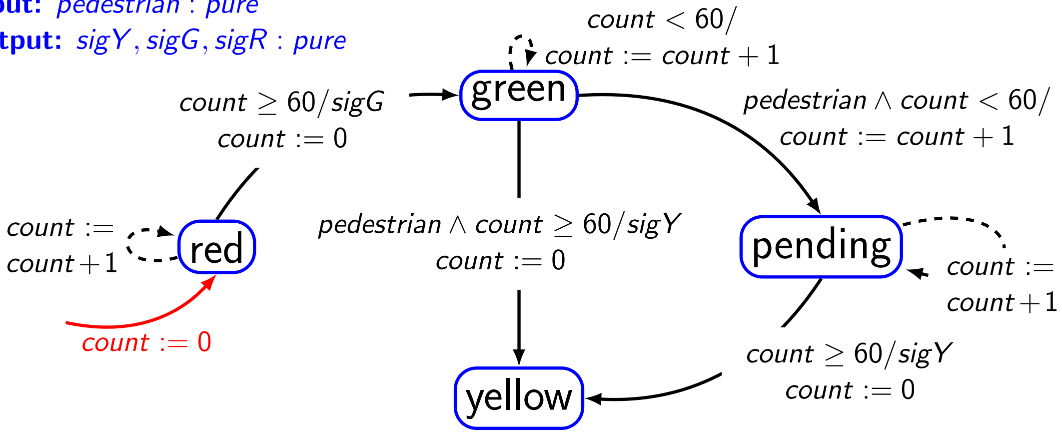
Example: pedestrian crosswalk

variable: $count : \{0, 1, \ldots, 60\}$
input: $pedestrian : pure$
output: $sigY, sigG, sigR : pure$



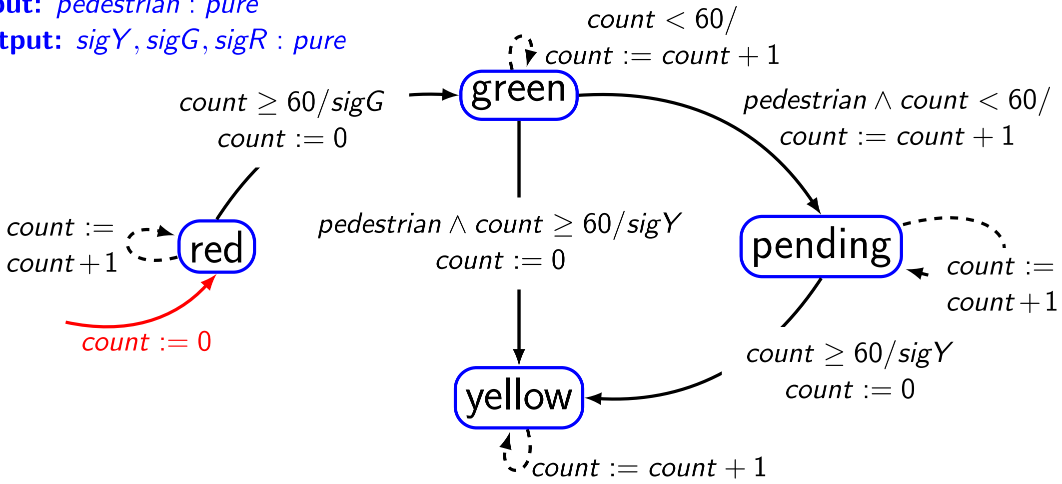
Example: pedestrian crosswalk

variable: $count : \{0, 1, \ldots, 60\}$
input: $pedestrian : pure$
output: $sigY, sigG, sigR : pure$



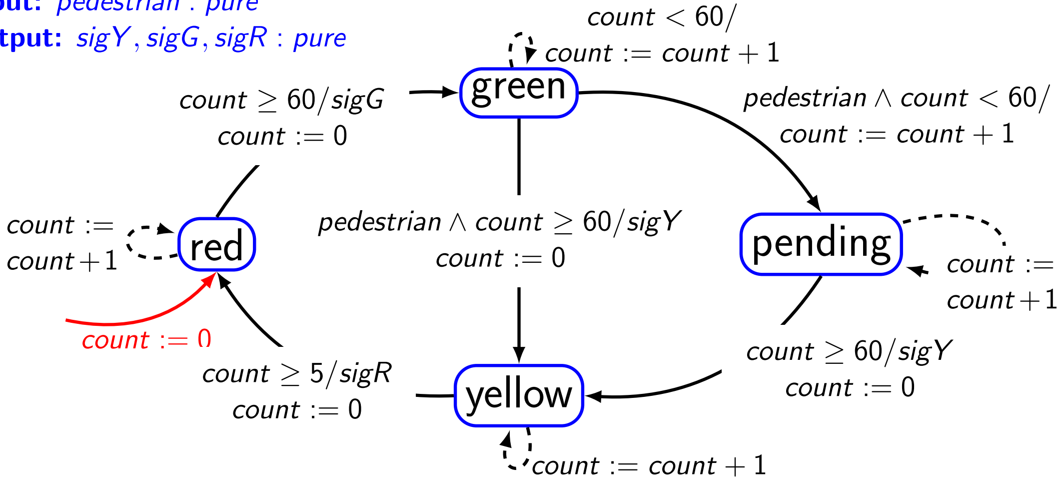
Example: pedestrian crosswalk

variable: $count : \{0, 1, \ldots, 60\}$
input: $pedestrian : pure$
output: $sigY, sigG, sigR : pure$



Example: pedestrian crosswalk

variable: $count : \{0, 1, \ldots, 60\}$
input: $pedestrian : pure$
output: $sigY, sigG, sigR : pure$

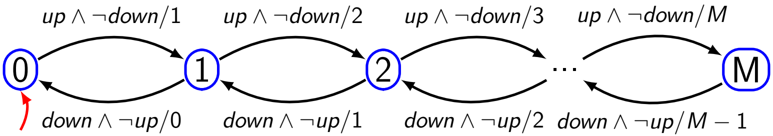


Extended FSM

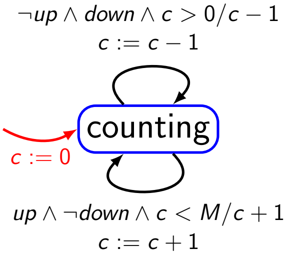
- The state of an extended machine includes not only the information about which discrete state the machine is in, but also what values any variables have.
  - Suppose there is, n discrete states, m variables each of which can take one of p possible values
  - Size of the state space will be $|States| = np^m$

Example

inputs: *up*, *down*: pure
outputs: $count$:$\{0, 1, \ldots, M\}$

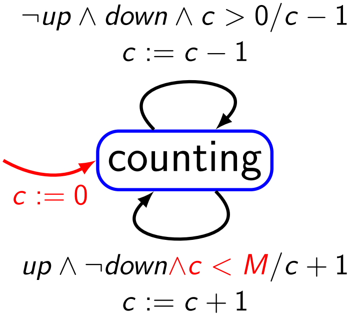


variable: $c : \{0, 1, \ldots, M\}$
inputs: *up*, *down*: pure
outputs: $count$:$\{0, 1, \ldots, M\}$

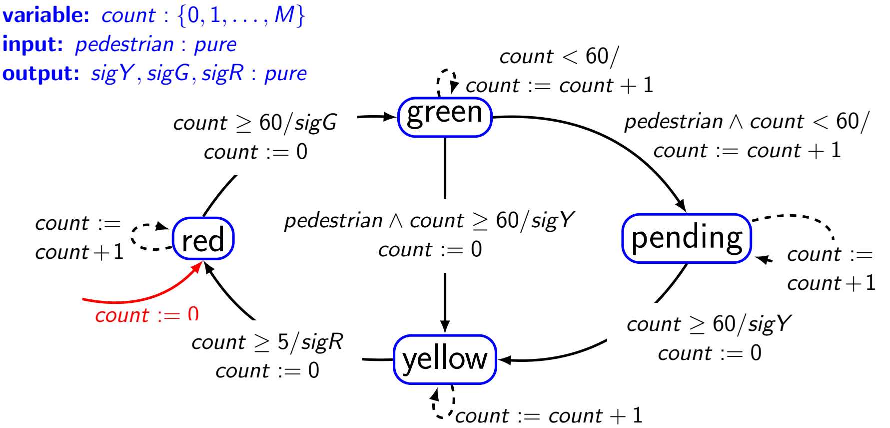


Example: infinite states

variable: $c : \{0, 1, \ldots, M\}$
inputs: *up*, *down*: pure
outputs: $count$:$\{0, 1, \ldots, M\}$



Pedestrian crosswalk: state count

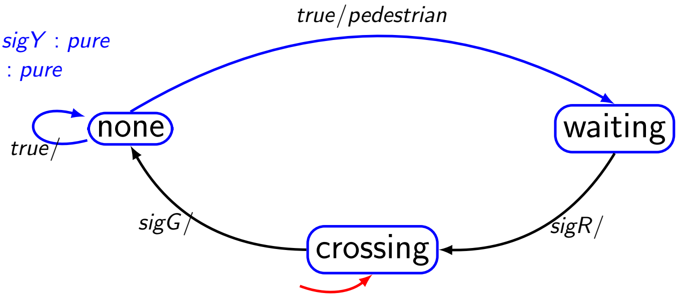


Nondeterminism

- A state machine interacts with the environment
- Modeling of pedestrian

- If for any state, two distinct transitions with guards that can evaluate to true in the same reaction, then the machine is **nondeterministic**

inputs: $sigR, sigG, sigY$: $pure$
output: $pedestrian$: $pure$

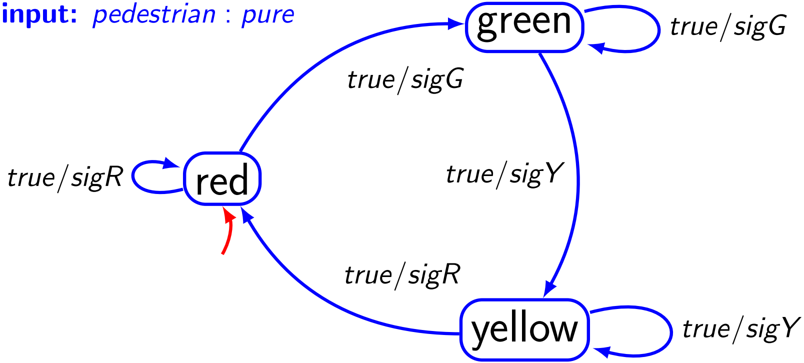


Nondeterministic FSM

- It is a tuple $\langle States, Inputs, Outputs, possibleUpdates, InitialStates \rangle$
- States — finite number of states
- Inputs — set of input valuations
- Outputs — set of output valuations
- possibleUpdates — $States \times Inputs \rightarrow 2^{States \times Outputs}$, mapping a state and input valuation to a next state and a set of possible (next state, output) pairs. Also known as **Transition Relation**
- InitialStates — start states

Nondeterministic FSM

output: *sigR, sigG, sigY : pure*
input: *pedestrian : pure*



Uses of nondeterminism

- Environment modeling — to hide irrelevant details
- Specifications — system requirements imposes constraints on some features while the others are unconstrained

- Probabilistic FSM is different from Non-deterministic FSM
  - In probabilistic FSM, every transition is associated with some probability

Behavior & Traces

- Behavior of state machine is an assignment of such signals to each port such that the signal on any output port is the output sequence produced by the input signals
- Example: garage counter

$$s_{up} = \{absent, absent, present, absent, present, present, \ldots\}$$
$$s_{down} = \{absent, absent, absent, present, absent, absent, \ldots\}$$
$$s_{count} = \{absent, absent, 1, 0, 1, 2, \ldots\}$$

- $s_{up}, s_{down}, s_{count}$ together form the behavior
- For deterministic FSM if input sequence is known the output sequence can be determined
- Set of all behaviors of a state machine M is called its language $L(M)$

Behavior & Traces (contd.)

- A behavior may be more conveniently represented as a sequence of valuations called observable trace
  - If x_i is input and Y_i is output then following is an observable sequence $((x_0, y_0), (x_1, y_1), \ldots)$
- An execution trace may be defined as

$$((x_0, s_0, y_0), (x_1, s_1, y_1), \ldots)$$

$$s_0 \xrightarrow{x_0/y_0} s_1 \xrightarrow{x_1/y_1} s_2 \ldots$$

Computation trees

- For nondeterministic machine, it may be useful to represent all possible traces