[dcc]

MODELLING MODAL BEHAVIOUR

Pedro Brandão, Sérgio Crisóstomo

Sistemas Embutidos 2020/21

U.PORTO

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

1

[dcc]

References

- Slides are from Edward A. Lee & Sanjit Seshia, UC Berkeley, EECS 149 Fall 2013
  -

U.PORTO

2

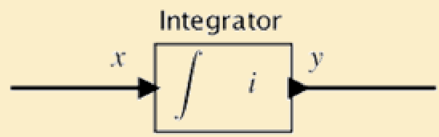
SE 2020/21 - Modelling Modal Behaviour - pbrandao

2

[dcc]

Recall Actor Model of a Continuous-Time System

- Example: integrator:



- Continuous-time signal:

$$x\colon \mathbb{R} \to \mathbb{R}, \qquad x \in (\mathbb{R} \to \mathbb{R}), \qquad x \in \mathbb{R}^{\mathbb{R}}$$

- Continuous-time actor, integrator

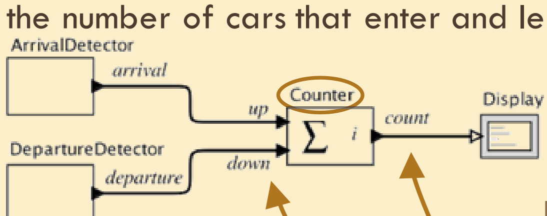
$$\mathbb{R}^{\mathbb{R}} \to \mathbb{R}^{\mathbb{R}}$$

U.PORTO

[dcc]

Discrete Systems

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



Pure signals: either present or absent. No other intrinsic value in the present input.

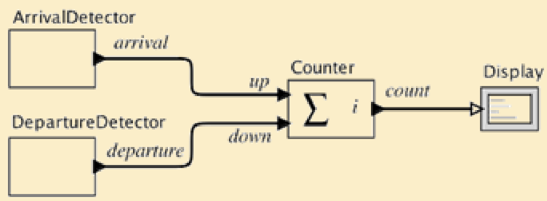
- Pure signal: $up\colon\ \mathbb{R} \longrightarrow \{absent, present\}$
  $down\colon\ \mathbb{R} \longrightarrow \{absent, present\}$
- Discrete actor:

$$Counter\colon (\mathbb{R} \longrightarrow \{absent, present\})^P \longrightarrow (\mathbb{R} \to \{absent\} \cup \mathbb{N})$$
$$P = \{up, down\}$$

U.PORTO

Reaction

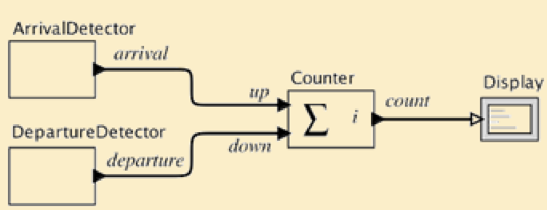
- For any $t \in \mathbb{R}$ where $\mathrm{up}(\mathrm{t}) \neq absent$ or $\mathrm{down}(\mathrm{t}) \neq absent$ the Counter **reacts**. It produces an output value in $\mathbb{N}$ and changes its internal **state**.



$$Counter: (\mathbb{R} \longrightarrow \{absent, present\})^P \longrightarrow (\mathbb{R} \rightarrow \{absent\} \cup \mathbb{N})$$
$$P = \{up, down\}$$

Inputs and Outputs at a Reaction

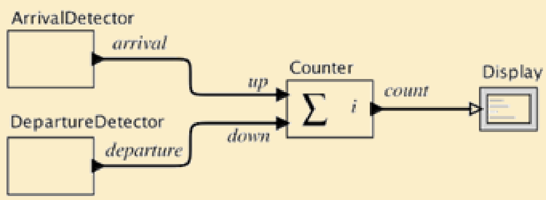
- For $t \in \mathbb{R}$ the inputs are in a set
$$Inputs = (\{up, down\} \longrightarrow \{absent, present\})$$
- And the outputs are in a set
$$Outputs = (\{count\} \rightarrow \{absent\} \cup \mathbb{N})$$



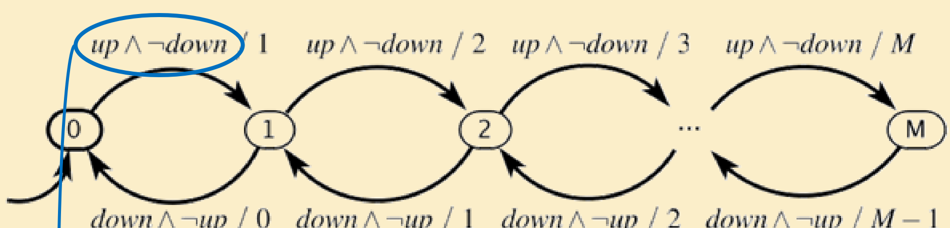
State Space

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

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



Garage Counter Finite State Machine (FSM)

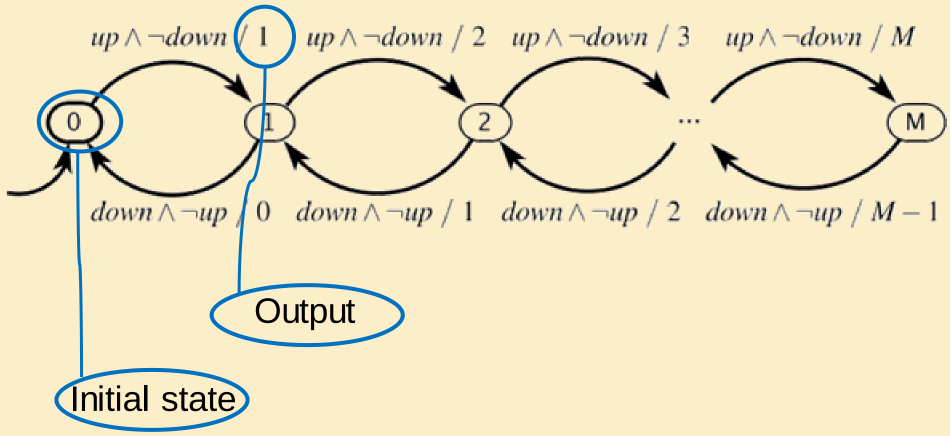


Guard g $\subseteq Inputs$ is specified using the shorthand

$$up\ \wedge \neg down$$

[dcc]

Garage Counter FSM



U.PORTO

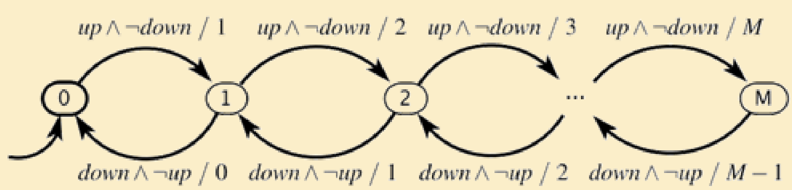
9

SE 2020/21 - Modelling Modal Behaviour - pbrandao

9

[dcc]

Garage Counter Mathematical Model



- Formally (States, Inputs, Outputs, update, initialState), where:
  - $States = \{0,1,2,\dots,M\}$
  - $Inputs = (\{up, down\} \longrightarrow \{absent, present\})$
  - $Outputs = (\{count\} \rightarrow \{absent\} \cup \mathbb{N})$
  - $Update: States \times Inputs \rightarrow States \times Outputs$
  - $initialState = 0$

The picture above defines the update function.

U.PORTO

10

SE 2020/21 - Modelling Modal Behaviour - pbrandao

10

Update for garage

- $States \times Inputs \to States \times Outputs$

Next State

output

State

$$update(s,i) = \begin{cases} (s+1, s+1) & \text{if } s < M \\ & \wedge\, i(up) = present \\ & \wedge\, i(down) = absent \\ (s-1, s-1) & \text{if } s > 0 \\ & \wedge\, i(up) = absent \\ & \wedge\, i(down) = present \\ (s, absent) & otherwise \end{cases}$$

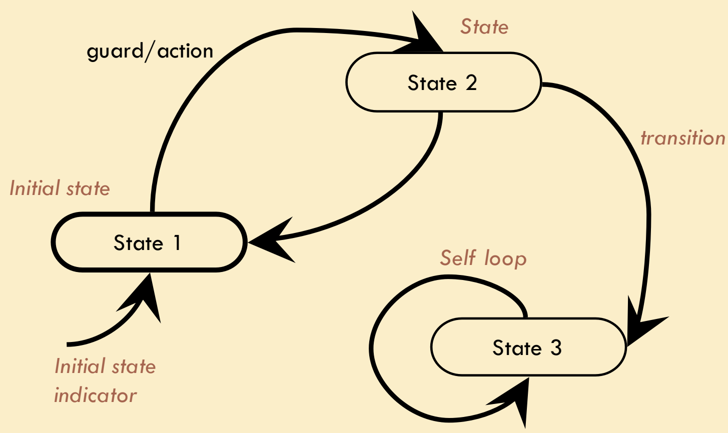
input

Update for garage

- $States \times Inputs \to States \times Outputs$

$$update(s,i) = \begin{cases} (s+1, s+1) & \text{if } s < M \\ & \wedge\, i(up) = present \\ & \wedge\, i(down) = absent \\ (s-1, s-1) & \text{if } s > 0 \\ & \wedge\, i(up) = absent \\ & \wedge\, i(down) = present \\ (s, absent) & otherwise \end{cases}$$

FSM Notation



SE 2020/21 - Modelling Modal Behaviour - pbrandao

13

13

Examples of Guards for Pure Signals

Pure signals: either present or absent. No other intrinsic value in the present input.

$true$	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
$p_1 \wedge \neg p_2$	Transition is enabled if p_1 is present and p_2 is absent

SE 2020/21 - Modelling Modal Behaviour - pbrandao

14

14

[dcc]

Examples of Guards for Signals with Numerical Values

Numerical signals: the signal's value explicitly conveys a value different for each different input value.

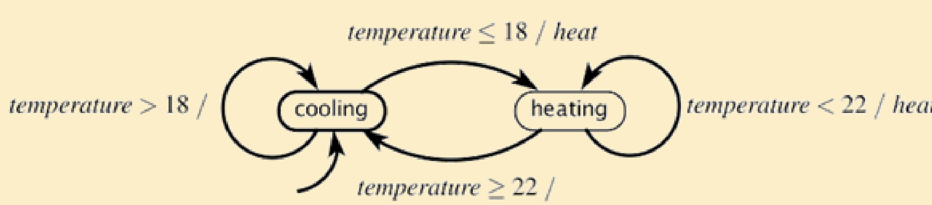
p_3	Transition is enabled if p_3 is present (not absent)
$p_3 = 1$	Transition is enabled if p_3 is present and has value 1
$p_3 = 1 \land p_1$	Transition is enabled if p_3 has value 1 and p_1 is present
$p_3 > 5$	Transition is enabled if p_3 is present and has value greater than 5

U.PORTO

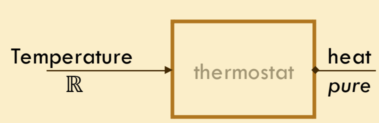
[dcc]

Example: Thermostat

If temperature is less or equal than 18°C heat
If temperature is greater or equal than 22°C let it cool



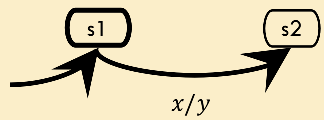
Use of hysteresis to avoid chattering



U.PORTO

When does a reaction occur?

Input: $x \in \{present, absent\}$
Output: $y \in \{present, absent\}$



- Suppose all inputs are discrete and a reaction occurs when any input is present. Then the above transition will be taken whenever the current state is $s1$ and x is present.

- This is an event-triggered model.

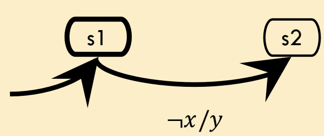
17

SE 2020/21 - Modelling Modal Behaviour - pbrandao

17

When does a reaction occur?

Input: $x \in \{present, absent\}$
Output: $y \in \{present, absent\}$



- Suppose x and y are discrete and pure signals. When does the transition occur?
  - *when the environment triggers a reaction and x is absent.*

18

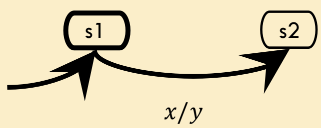
SE 2020/21 - Modelling Modal Behaviour - pbrandao

18

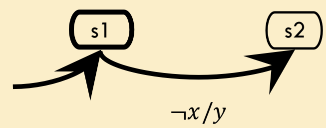
[dcc]

When does a reaction occur?

Input: x ∈ $\{present, absent\}$
Output: y ∈ $\{present, absent\}$



Input: x ∈ $\{present, absent\}$
Output: y ∈ $\{present, absent\}$



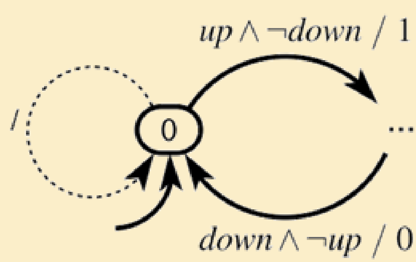
- Suppose all inputs are discrete and a reaction occurs on the tick of an external clock.
- This is a time-triggered model.

U.PORTO

19

[dcc]

More Notation: 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?

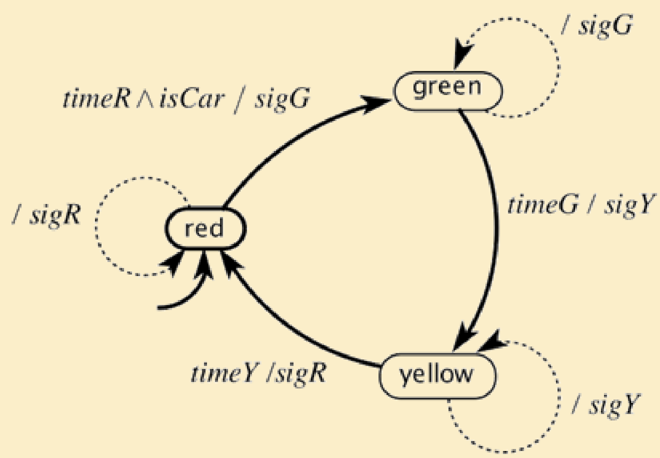
U.PORTO

20

[dcc]

Default transitions

Only show default transitions if they are guarded or produce outputs



Example: Traffic Light Controller

U.PORTO

21

SE 2020/21 - Modelling Modal Behaviour - pbrandao

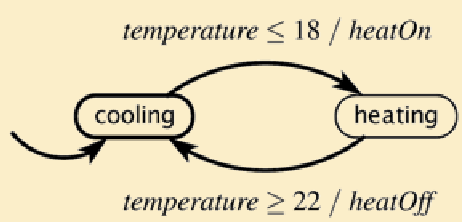
21

[dcc]

Default transitions II

- Example where default transitions need not be shown

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



- *Time or event triggered?*
- *Formally write down the description of this FSM as a 5-tuple: (States, Inputs, Outputs, update, initialState)*

U.PORTO

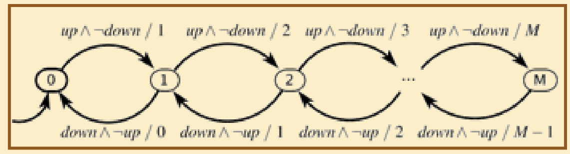
22

SE 2020/21 - Modelling Modal Behaviour - pbrandao

22

Garage Counter and Cooling

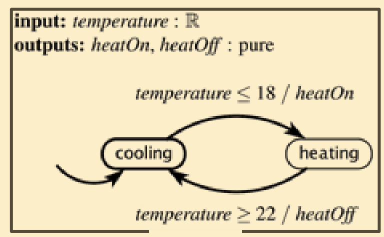
$States = \{0,1,2,\ldots,M\}$
$Inputs = (\{up, down\} \longrightarrow \{absent, present\})$
$Outputs = (\{count\} \rightarrow \{absent\} \cup \mathbb{N})$
$Update\colon States \times Input \rightarrow States \times Outputs$
$initialState = 0$



Garage Counter

$$update(s,i) = \begin{cases} (s+1, s+1) & \text{if } s < M \\ & \wedge\, i(up) = present \\ & \wedge\, i(down) = absent \\ (s-1, s-1) & \text{if } s > 0 \\ & \wedge\, i(up) = absent \\ & \wedge\, i(down) = present \\ (s, absent) & otherwise \end{cases}$$

Garage Counter

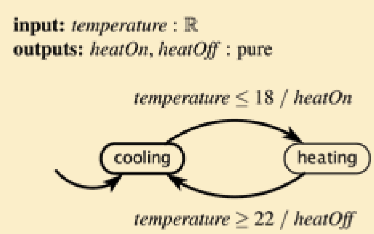


Cooling

23

Cooling

$States = \{cooling, heating\}$
$Inputs = (temperature \longrightarrow \mathbb{R})$
$Outputs = (\{heatOn, HeatOff\} \rightarrow \{absent, present\})$
$Update\colon States \times Input \rightarrow States \times Outputs$
$initialState = cooling$

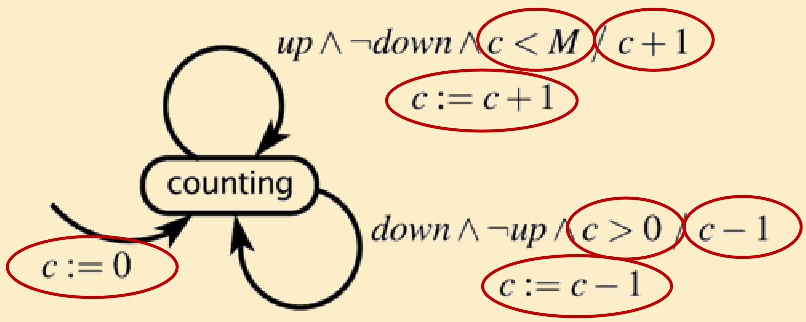


$$update(s,i) = \begin{cases} (heating, present, absent) & \text{if } s = cooling \\ & \wedge\, i(temperature) \leq 18 \\ (cooling, absent, present) & \text{if } s = heating \\ & \wedge\, i(temperature) \geq 22 \\ (s, absent, absent) & otherwise \end{cases}$$

24

Extended State Machines

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



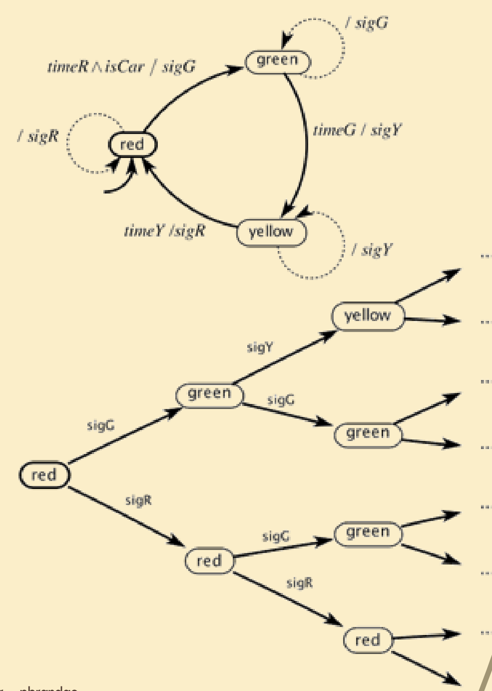
SE 2020/21 - Modelling Modal Behaviour - pbrandao

25

25

Behaviours and Traces

- FSM **behaviour** is a sequence of (non-stuttering) steps.
- A **trace** is the record of inputs, states, and outputs in a behaviour.
- A **computation tree** is a graphical representation of all possible traces.
- FSMs are suitable for formal analysis. For example, **safety** analysis might show that some unsafe state is not reachable.



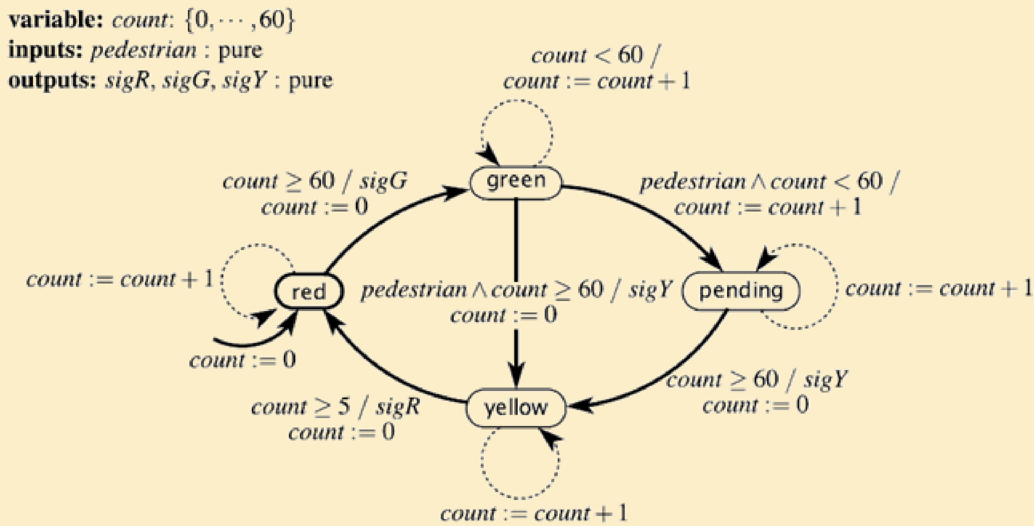
SE 2020/21 - Modelling Modal Behaviour - pbrandao

26

26

[dcc]

Traffic Light Controller



U.PORTO

27

SE 2020/21 - Modelling Modal Behaviour - pbrandao

27

[dcc]

Definitions

- Stuttering transition: (possibly implicit) default transition that is enabled when all inputs are absent, that does not change state, and that produces absent outputs.
- Receptiveness: For any input values, some transition is enabled, for each state. 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.
  - *Only one is enabled*

U.PORTO

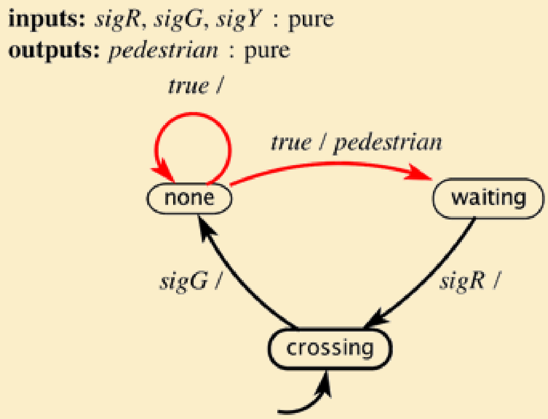
28

SE 2020/21 - Modelling Modal Behaviour - pbrandao

28

Example: Non determinate FSM

- Model of the environment for the pedestrian arrival/presence, abstracted using nondeterminism:



SE 2020/21 - Modelling Modal Behaviour - pbrandao

29

Example: Non determinate FSM

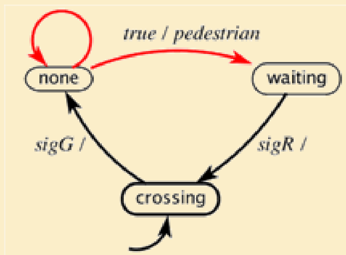
- Formally, the update function is replaced by an update relation that maps a `(state, input)` combination to a set of possible `(next state, output)` pairs.
- And there is a set of initial states
- Formally, the update function is replaced by a relation

$$possibleUpdates: States \times Inputs \rightarrow 2^{States \times Outputs}$$

SE 2020/21 - Modelling Modal Behaviour - pbrandao

30

Example for the pedestrian



$$States = \{none, waiting, crossing\}$$
$$Inputs = (\{sigG, sigY, sigR\} \longrightarrow \{present, absent\})$$
$$Outputs = (\{pedestrian\} \rightarrow \{present, absent\})$$
$$initialState = \{crossing\}$$

$$possibleUpdates(s,i) = \begin{cases} \{(none, absent)\} & if\ s = crossing \\ & \wedge\, i(sigG) = present \\ \{(none, absent), (waiting, present),\} & if\ s = none \\ \{(crossing, absent)\} & if\ s = waiting \\ & \wedge\, i(sigR) = present \\ \{(s, absent)\} & otherwise \end{cases}$$

31

SE 2020/21 - Modelling Modal Behaviour - pbrandao

31

Uses of nondeterminism

1. Modelling unknown aspects of the environment or system
   – *Such as: how the environment changes a robot's orientation*
2. Hiding detail in a specification of the system
   – *If specification for traffic light only specified light order (R →G →Y →R) and not time in each colour*

- Any other reasons why nondeterministic FSMs might be preferred over deterministic FSMs?

32

SE 2020/21 - Modelling Modal Behaviour - pbrandao

32

Size Matters

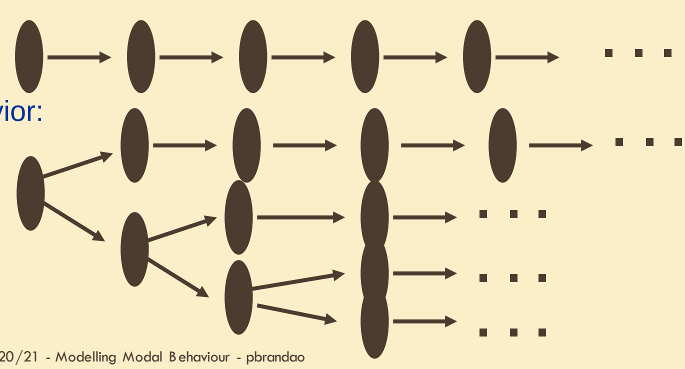
- Non-deterministic FSMs are more compact than deterministic FSMs
  - *A classic result in automata theory shows that a nondeterministic FSM has a related deterministic FSM that is equivalent in a technical sense (language equivalence, covered in Chapter 13).*
  - *But the deterministic machine has, in the worst case, many more states (exponential in the number of states of the nondeterministic machine, see Appendix B).*

Non-deterministic Behaviour: Tree of Computations

- For a fixed input sequence:
- A deterministic system exhibits a single behaviour
- A non-deterministic system exhibits a set of behaviours
  - *visualized as a computation tree*

Deterministic FSM behavior:

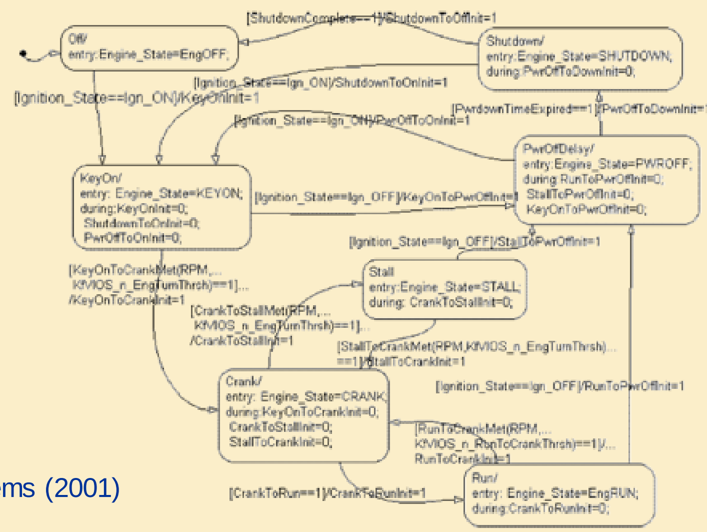
Non-deterministic FSM behavior:



Related points

- What does receptiveness mean for non-deterministic state machines?
- Non-deterministic ≠ Probabilistic

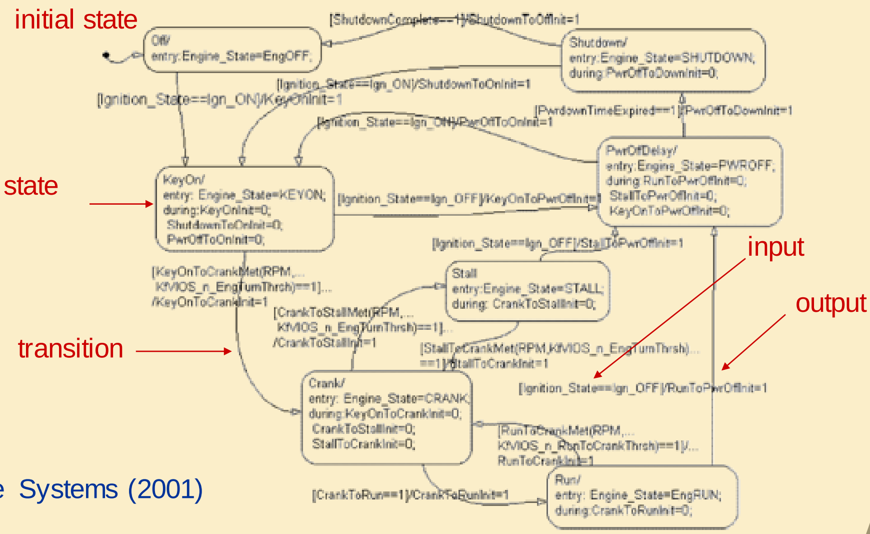
Example from Industry: Engine Control



Source:
Delphi Automotive Systems (2001)

[dcc]

Elements of a Modal Model (FSM)

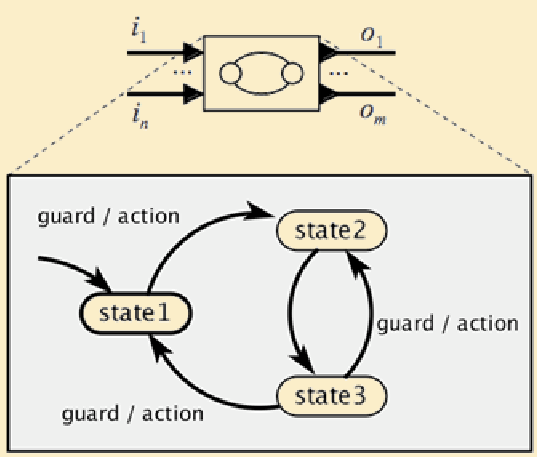


Source:
Delphi Automotive Systems (2001)

U.PORTO

[dcc]

Actor Model of an FSM



This model enables **composition** of state machines.

U.PORTO

[dcc]

What we will be able to do with FSMs

FSMs provide:

- A way to represent the system for:
  - *Mathematical analysis*
  - *So that a computer program can manipulate it*
- A way to model the environment of a system.
- A way to represent what the system must do and must not do – its specification.
- A way to check whether the system satisfies its specification in its operating environment.

U.PORTO

39

39

[dcc]

Summary

- Discrete systems
- Finite State Machines
- Mathematical models
- Non-determinism
- Trees and behaviours

U.PORTO

40

40