Extended State Machines

CS 447–Wireless Embedded Systems
Outline

• Motivation
• Extended state machine
• Example

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

$\text{counting}$

$up \land \neg down \land c < M \rightarrow c + 1$
$c := c + 1$

$c := 0$

$down \land \neg up \land c > 0 \rightarrow c - 1$
$c := c - 1$
Motivation

Notation for FSM gets unwieldy when number of states large

• E.g., garage counter
• M is small, easy to draw
• M is large, can be cumbersome
Motivation

Extended state machine solves this problem
• Augments FSM model w/ variables
Outline

• Motivation
• Extended state machine
• Example

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

$up \land \neg down \land c < M \implies c + 1$
$c := c + 1$

c := 0$

down \land \neg up \land c > 0 \implies c - 1$
$c := c - 1$
Extended State Machine

• Adds variables to FSM model
• Variables can be read / written during transitions
• E.g., garage counter
Extended State Machine

• E.g., garage counter extended state machine
• Adds a variable: c
• Indicated in state machine
• **NOT** input or output

```
variable: c: \{0, \ldots, M\}
inputs: up, down: pure
output: count: \{0, \ldots, M\}

\begin{align*}
up \land \neg down \land c < M & / c + 1 \\
c & := c + 1
\end{align*}

\begin{align*}
c & := 0 \\
\neg up \land c > 0 & / c - 1 \\
c & := c - 1
\end{align*}
```
Extended State Machine

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

$up \land \neg down \land c < M / c + 1$
$c := c + 1$

$c := 0$

$down \land \neg up \land c > 0 / c - 1$
$c := c - 1$
Aside: Mealy vs. Moore State Machines

Mealy
• Output determined by state + input
• Output occurs during transition

Moore
• Output determined by state only
• Output occurs within new state
Extended State Machines

General notation:

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

```
State 1
```

```
State 2
```

```
guard / output action
set action
```

```
initial set action
guard / output action
set action
```
Extended State Machines

All variables must be declared (in addition to inputs, outputs)
Extended State Machines

Variables initialized in initial transition
Extended State Machines

Guard / action same as regular FSM, can also refer to variables

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

guard / output action
set action

State 1

initial set action

guard / output action
set action

State 2
Extended State Machines

Transitions have an additional line to modify variable
Extended State Machines

Variable modified **after** guard evaluated and output produced
- Variable used in guard / output **before** it’s modified
- Similar to post increment (e.g., \( x = i++ \))
Extended State Machines

More than one "set action" is allowed for each transition
• Actions made in sequence (e.g., i++, j--)

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

guard / output action
set action

State 1
intial set action

State 2

guard / output action
set action
set action
set action
Outline

• Motivation
• Extended state machine
• Example

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

\[ \begin{align*}
up \land \neg down \land c < M & / c + 1 \\
c & := c + 1 \\
\neg up \land c > 0 & / c - 1 \\
c & := c - 1 
\end{align*} \]
Example

Stop light with pedestrian button
• Time triggered state machine (1 Hz)
• 60 seconds in red then transitions to green
• Remains green until pedestrian present
• Must stay green for at least 60 seconds
• Transitions from green to “pending” if 60 seconds not done
• Transitions to yellow for 5 seconds
• Then red for 60 seconds...
Example

States: red, green, pending, yellow
Variable: count: \{ 0, 1, ..., 60 \}
Inputs: pedestrian: pure
Outputs: sigR, sigG, sigY : pure
Example

Start w/ states

red

green

yellow

pending
Example

Start w/ states

red
green
yellow

pending

count = 0
Example

Go from red to green? Yes, when \( \text{count} \geq 60 \)

\[
\begin{align*}
\text{count} & \geq 60 / \text{sigG} \\
\text{count} & := 0 \\
\text{count} & = 0
\end{align*}
\]
Example

Use default transition to update count in red state

- count >= 60 / sigG
  count := 0

- count = 0

- count++
Example

Use default transition to update count in red state

```plaintext
- count >= 60 / sigG
- count := 0
- count = 0
- count++
```

Diagram:
- Red state
- Green state
- Yellow state
Example

Now you fill in the remaining transitions...

Stop light with pedestrian button
• Time triggered state machine (1 Hz)
• 60 seconds in red then transitions to green
• Remains green until pedestrian present
• Must stay green for at least 60 seconds
• Transitions from green to “pending” if 60 seconds not done
• Transitions to yellow for 5 seconds
• Then red for 60 seconds...
Example

**variable:** \( \text{count}: \{0, \cdots, 60\} \)

**inputs:** \( \text{pedestrian} : \text{pure} \)

**outputs:** \( \text{sigR, sigG, sigY} : \text{pure} \)

\[
\begin{align*}
\text{green} & : \quad \text{count} < 60 / \\
& \quad \text{count} := \text{count} + 1 \\
\text{red} & : \quad \text{count} \geq 60 / \text{sigG} \\
& \quad \text{count} := 0 \\
\text{yellow} & : \quad \text{pedestrian} \land \text{count} \geq 60 / \text{sigY} \\
& \quad \text{count} := 0 \\
\text{pending} & : \quad \text{count} \geq 60 / \text{sigY} \\
& \quad \text{count} := 0 \\
& \quad \text{count} := \text{count} + 1
\end{align*}
\]