

POLITECNICO DI MILANO

AUTONOMOUS VEHICLES

S. Arrigoni



POLITECNICO
MILANO 1863



Finite State Machines

introduction

Introduction

What are Finite State Machines (FSM)?

Definition:

State machines are a ***method of modeling*** systems whose ***output depends on the entire history*** of their inputs, (and not just on the most recent input).

It can be used for:

- ✓ Design ***control logics*** based on:
 - Events
 - Time-based
 - External signals
- ***Model behavior*** of a system (due to environment interaction)
- ***Predict*** future behavior (based on past inputs)

Introduction

What are Finite State Machines (FSM)?

It is *mathematically* defined as a six-tuple (I, S, s_0, τ, o, O) where:

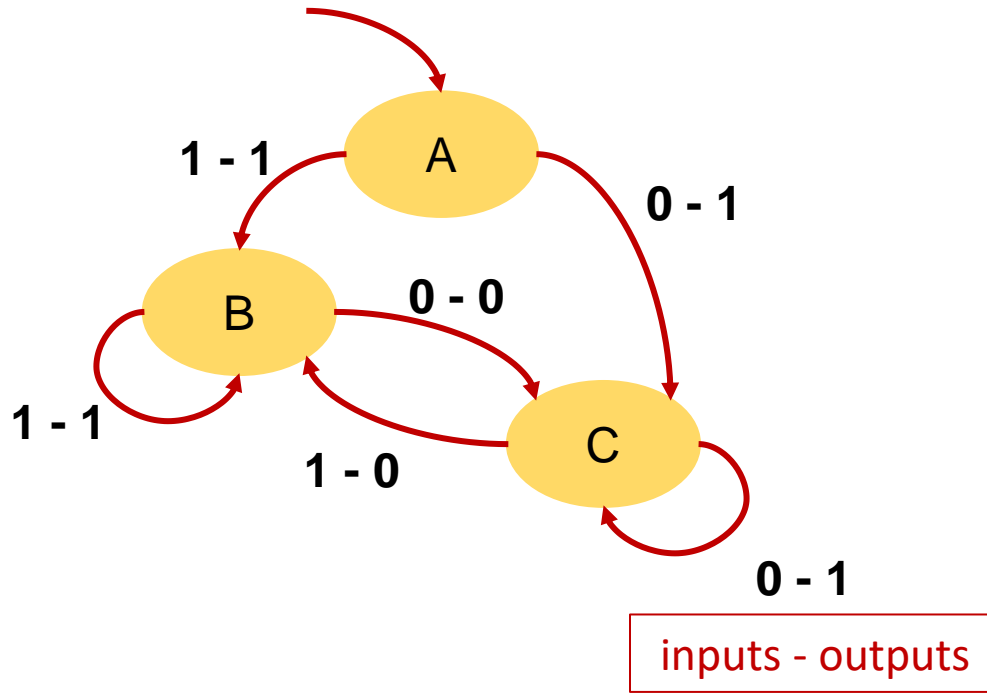
- ✓ $I \rightarrow$ inputs set (a finite, non-empty set of symbols) aka input vocabulary
- ✓ $S \rightarrow$ ***finite***, non-empty set of states
- ✓ $s_0 \rightarrow$ initial state at t_0 ($s_0 \in S$)
- ✓ $O \rightarrow$ outputs set (set of symbols) aka output vocabulary
- ✓ $\tau \rightarrow$ transition (next-state) function, $\tau(i_t, s_t) \rightarrow s_{t+1}$
- ✓ $o \rightarrow$ output function, $o(i_t, s_t) \rightarrow o_t$

Introduction

What are Finite State Machines (FSM)?

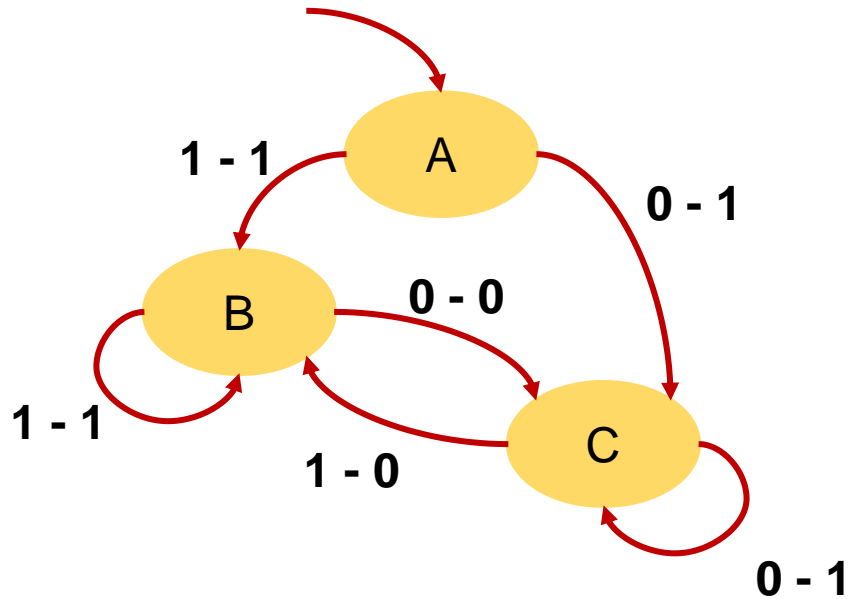
Let's look at the example:

- ✓ inputs set $I \rightarrow \{0,1\}$
- ✓ set of states $S \rightarrow \{A, B, C\}$
- ✓ $s_0 \rightarrow$ initial state at t_0 A
- ✓ outputs set $O \rightarrow \{0,1\}$



Introduction

What are Finite State Machines (FSM)?



Transition and outputs functions (τ, o) can be defined by a tabular:

- ✓ states on the rows
- ✓ inputs on the columns
- ✓ Each cell contains s_{t+1}, o_t

	0	1
A	B,1	C,1
B	C,0	B,1
C	C,1	B,0

Introduction

Example: Parking Gate Control



Gate positions:

- ✓ Top
- ✓ Middle
- ✓ Bottom

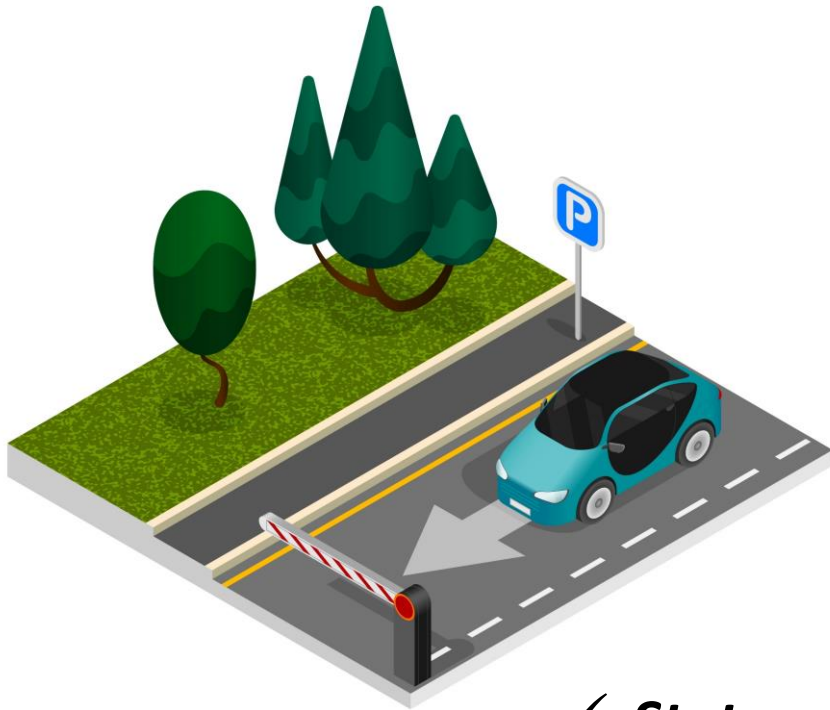
#sensor1: a car is waiting

#sensor2: a car has just passed through

Actions:

- ✓ raise the gate
- ✓ lower the gate
- ✓ no operation (nop).

Introduction



Example: Parking Gate Control

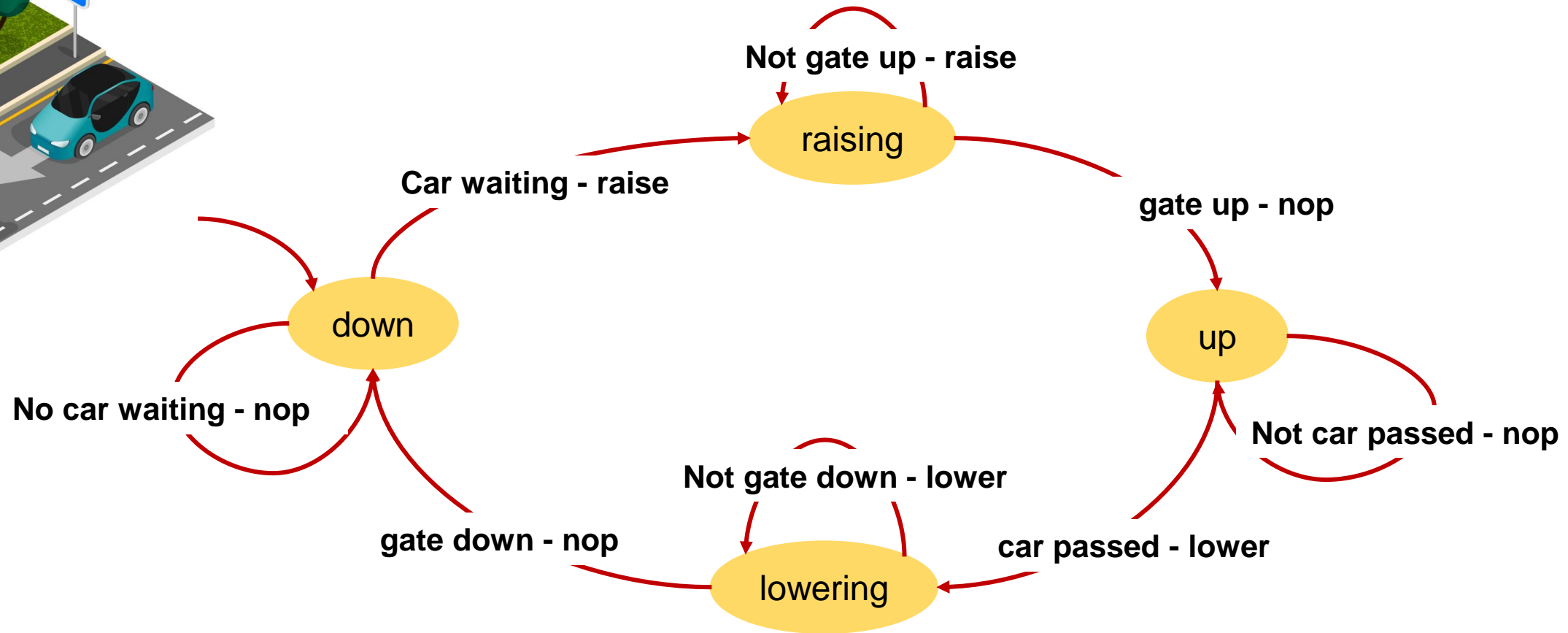
Expected behavior:

- If a car is close, need to raise the arm until 'top' position
- Gate has to stay up until the car has passed
- Gate has to lower after the car has gone

- ✓ **States**: 'down', 'raising', 'up', 'lowering'
- ✓ **Input**: 'no car waiting', 'car waiting', 'gate up', 'not gate up', 'gate down', 'not gate down', 'car passed', 'not car passed'
- ✓ **Output**: 'raise', 'lower', 'nop'

Introduction

Example: Parking Gate Control



Introduction

Control complexity in real world

The complexity of a finite state machine (FSM) can easily explode!
(new states can generate an exponential number of transitions).

The main strategies to keep the architecture tractable are:

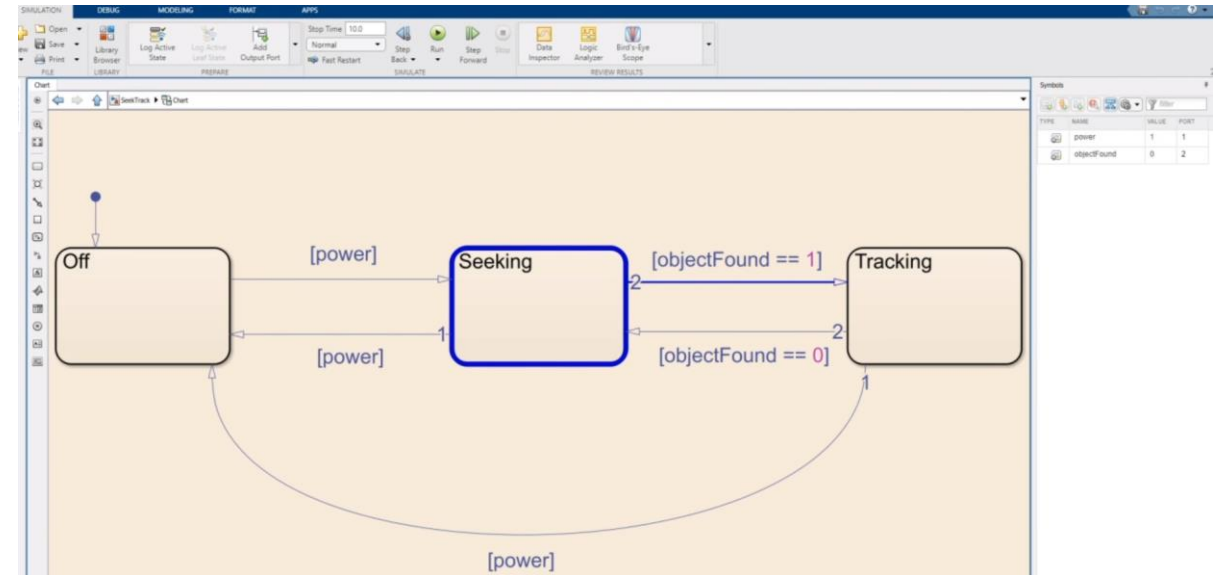
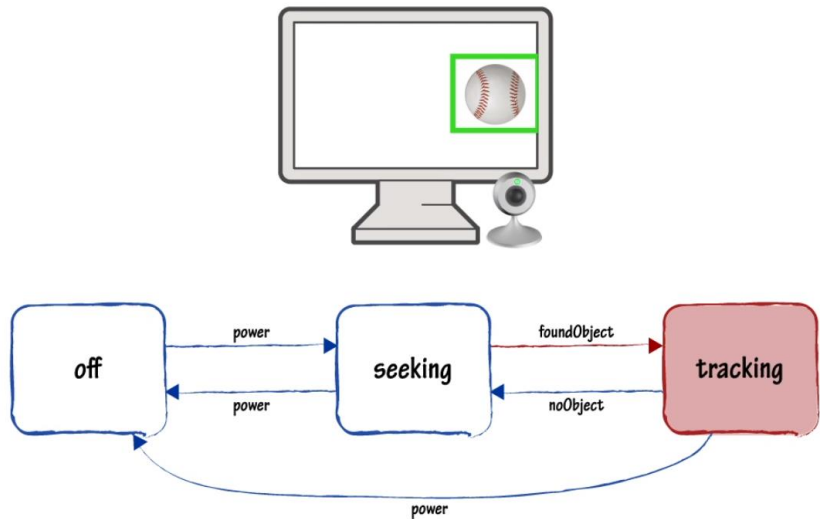
- ✓ Reduction of redundant states
- ✓ ***Hierarchical finite state machines***
- ✓ ***Composition using common patterns (decomposition)***

Software requirements



FSM using Stateflow


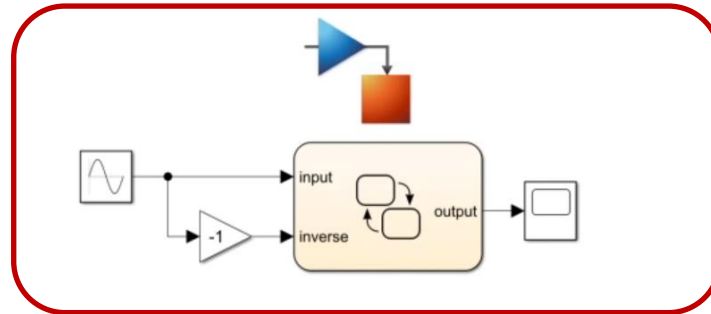
Stateflow allows to easily design / test a state machine



Ref: *stateflow onramp mathworks®*

FSM using Stateflow

Stateflow allows to easily design / test a state machine



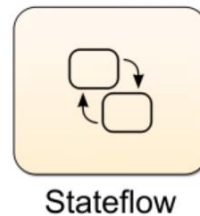
```
>> chart = SeekTrack()

chart =


Stateflow Chart

Execution Function
step(chart, Name, Value)
Active States: {'Off'}

chart 1x1 SeekTrack
```

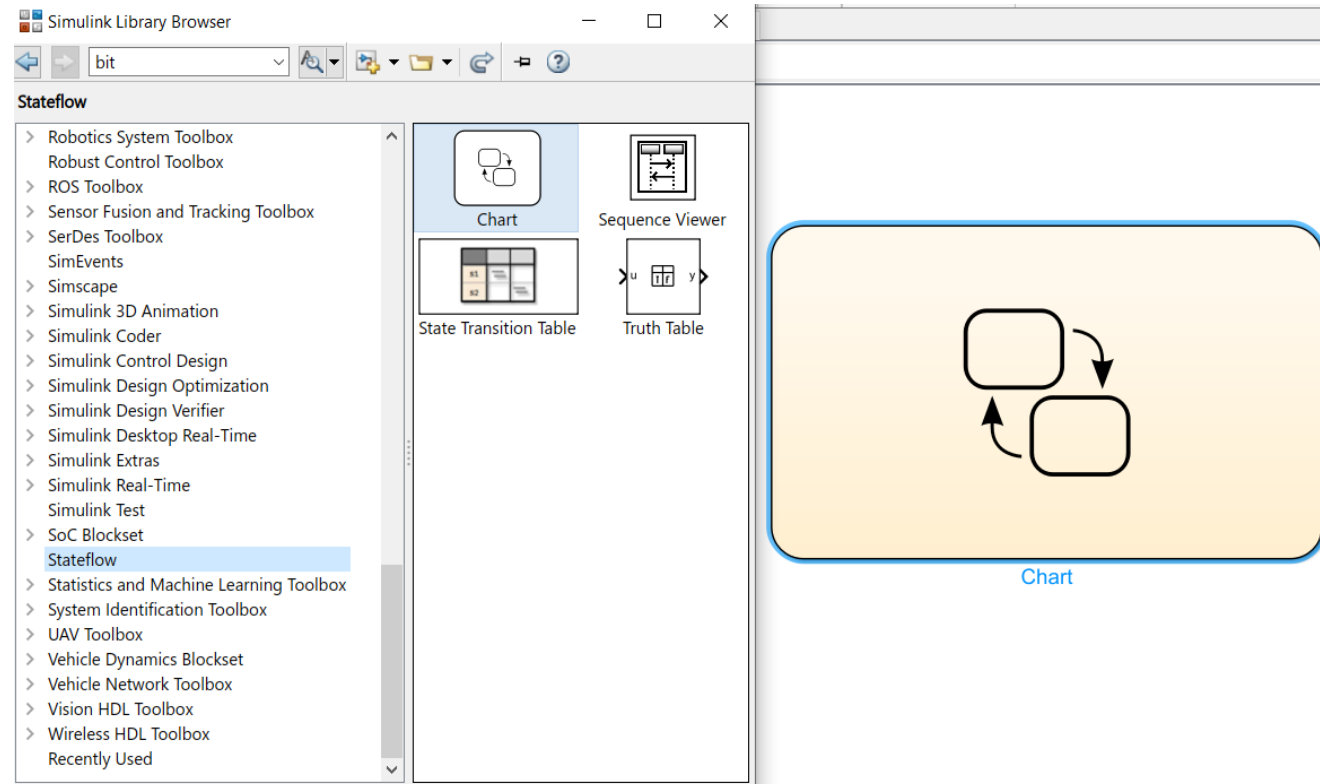


```
Code
SeekTrack.c
128 /* Chart: '<Root>/Chart' */
129 if (rtDW.is_active_c3_SeekTrack == 0U) {
130     rtDW.is_active_c3_SeekTrack = 1U;
131     rtDW.is_c3_SeekTrack = IN_Off;
132 } else {
133     switch (rtDW.is_c3_SeekTrack) {
134     case IN_Off:
135         if (rtDW.FromIs[0] != 0.0) {
136             rtDW.is_c3_SeekTrack = IN_Seeking;
137         }
138         break;
139
140     case IN_Seeking:
141         if (rtDW.FromIs[0] != 0.0) {
142             rtDW.is_c3_SeekTrack = IN_Off;
143         }
144     }
```



FSM using Stateflow

Stateflow is a graphical programming language that allows you to quickly create a runnable model of your system

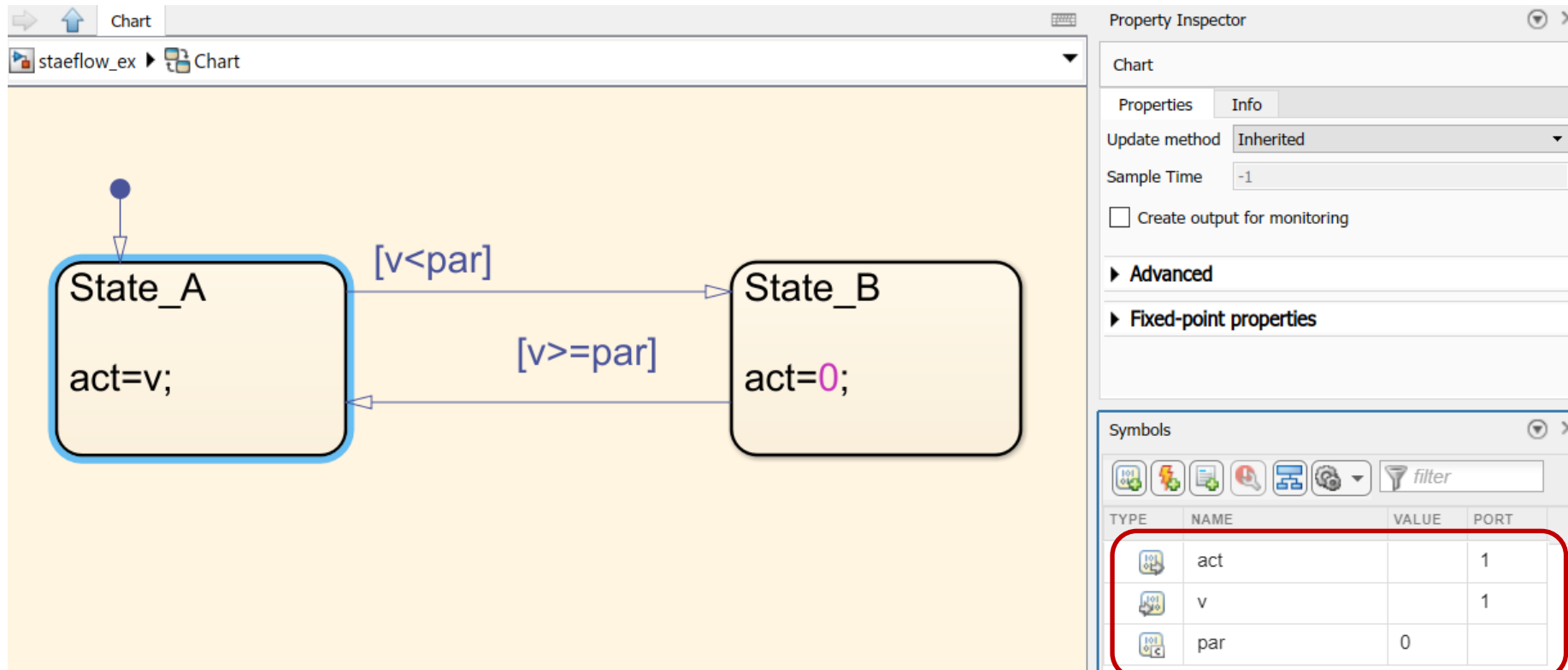


FSM using Stateflow

Main components:




Design panel: draw states, transitions, ...

Symbols: add new data, fix unknown, ...



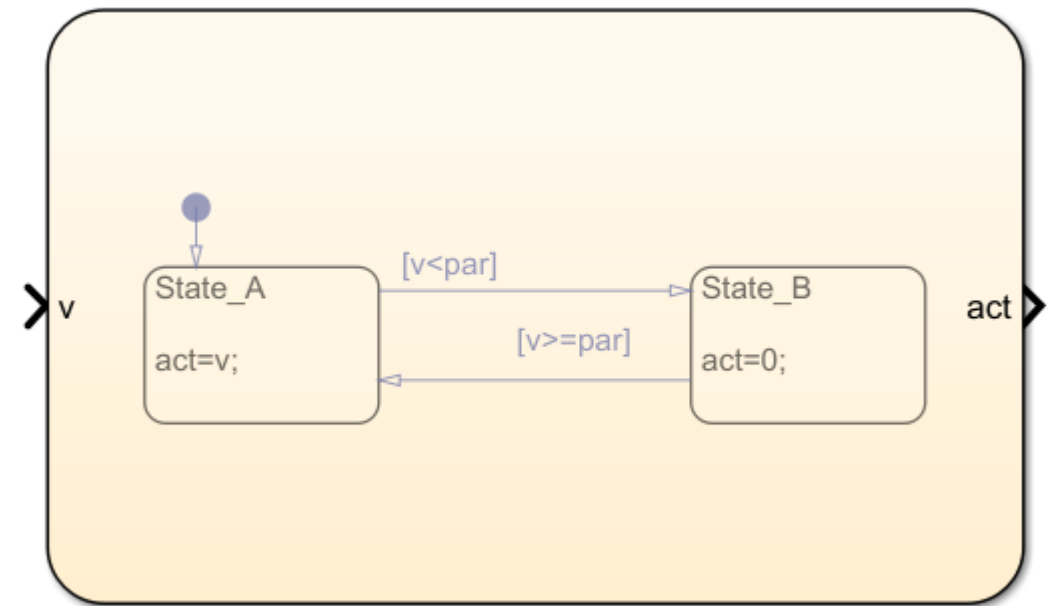
FSM using Stateflow

#example 1

Symbols			
TYPE	NAME	VALUE	PORT
	act		1
	v		1
	par	0	

outputs
inputs
constant

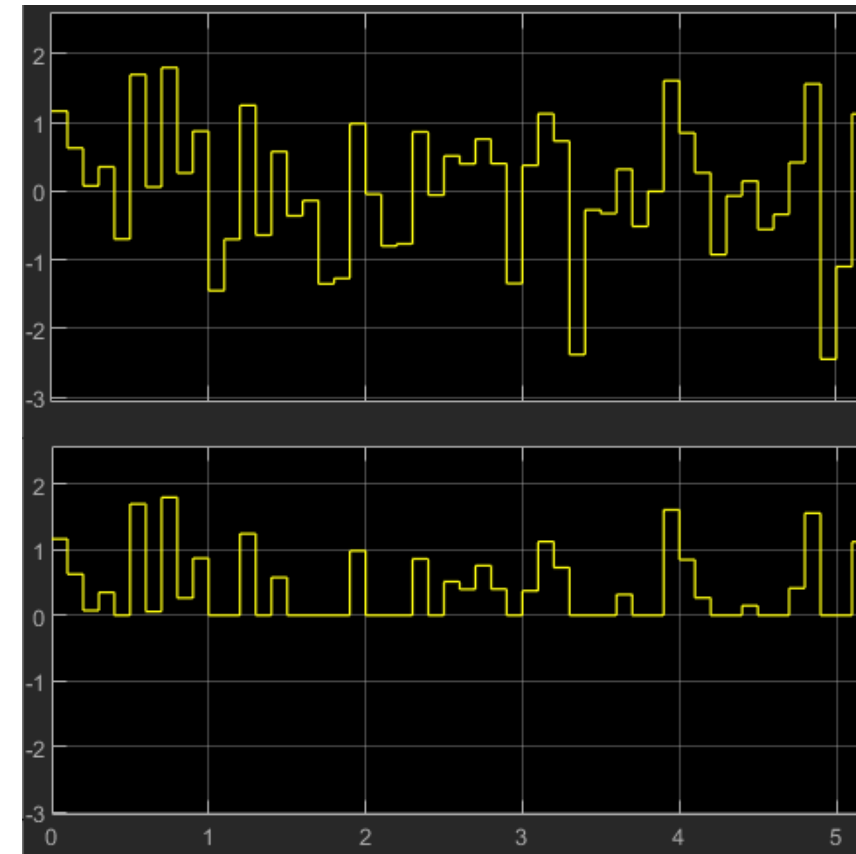
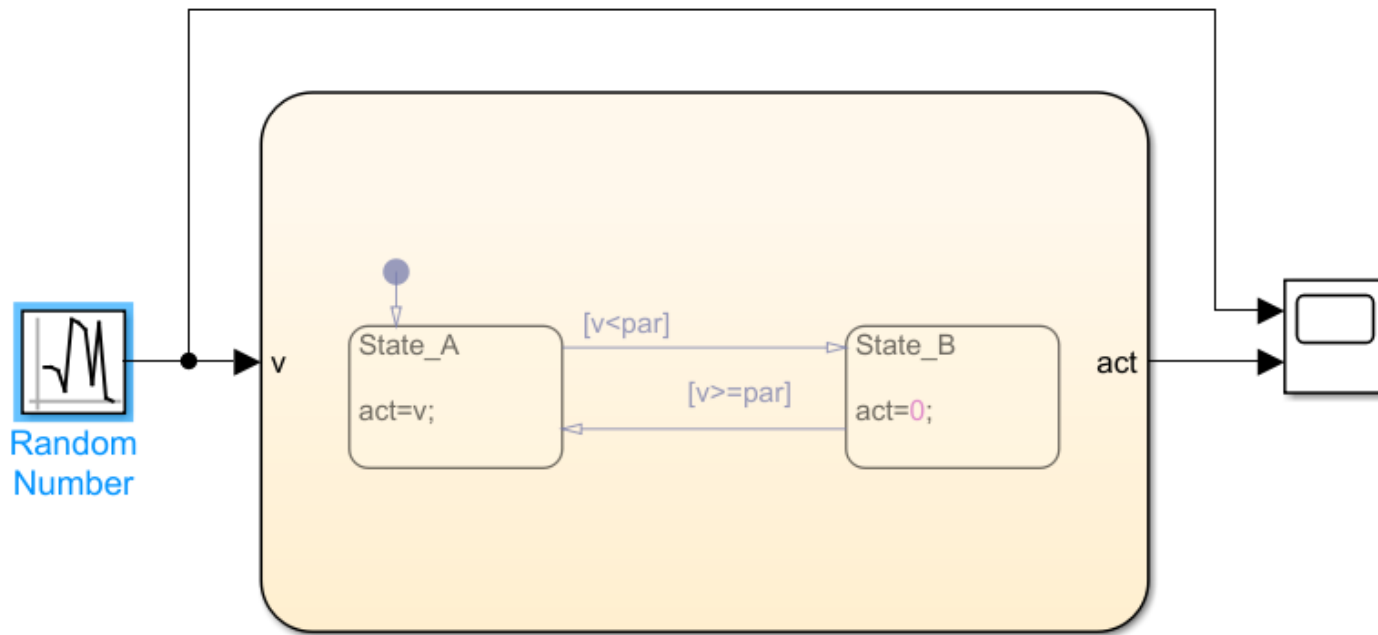
Fix data types



Chart

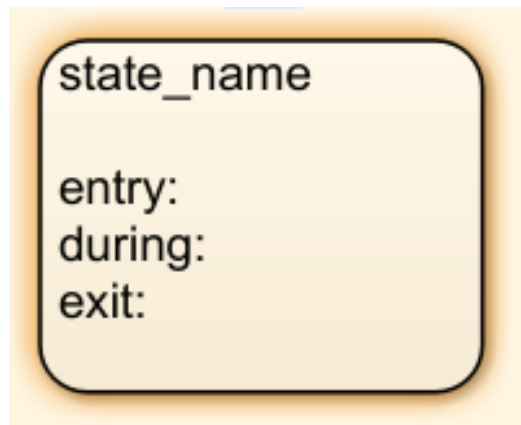
FSM using Stateflow

#example 1



FSM using Stateflow

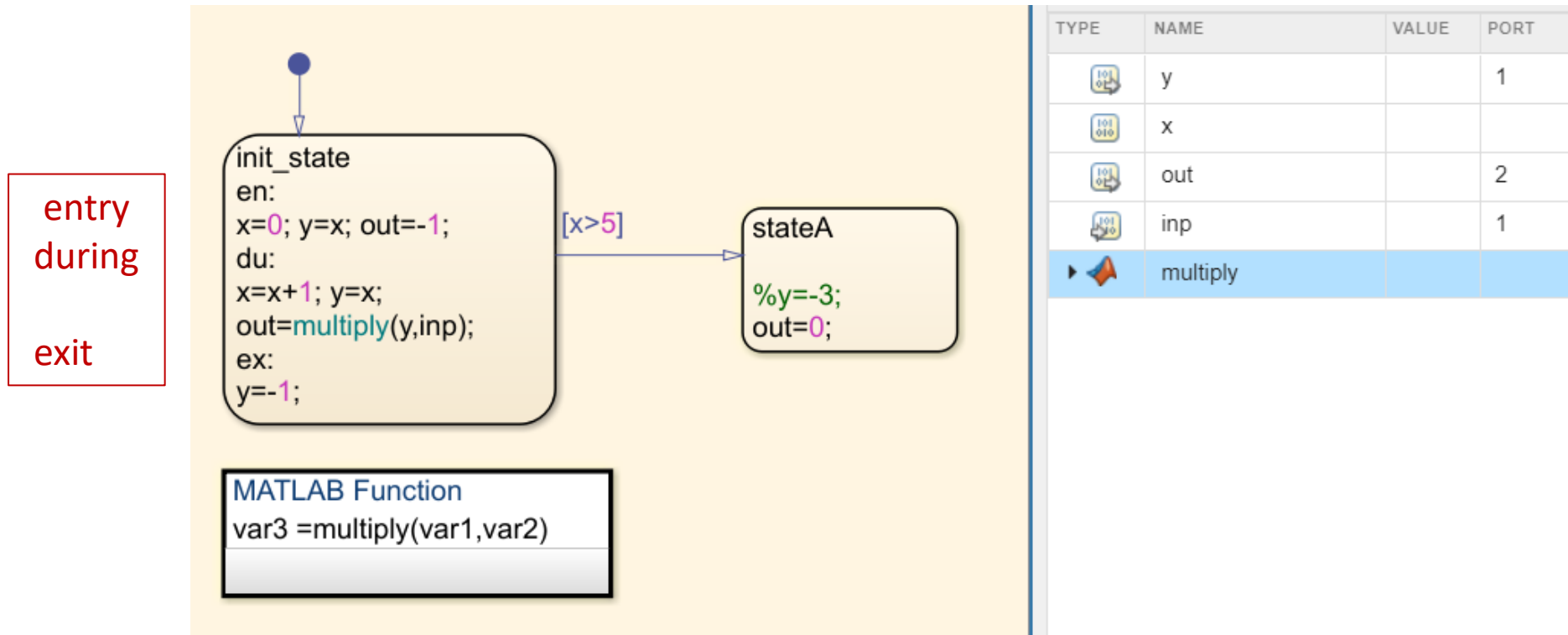
State actions



entry	en	Action occurs on a time step when the state becomes active.
during	du	Action occurs on a time step when the state is already active and the chart does not transition out of the state.
exit	ex	Action occurs on a time step when the chart transitions out of the state.

FSM using Stateflow

#example 2



FSM using Stateflow

#example 2

```
out=multiply(y,inp);
```

MATLAB Function

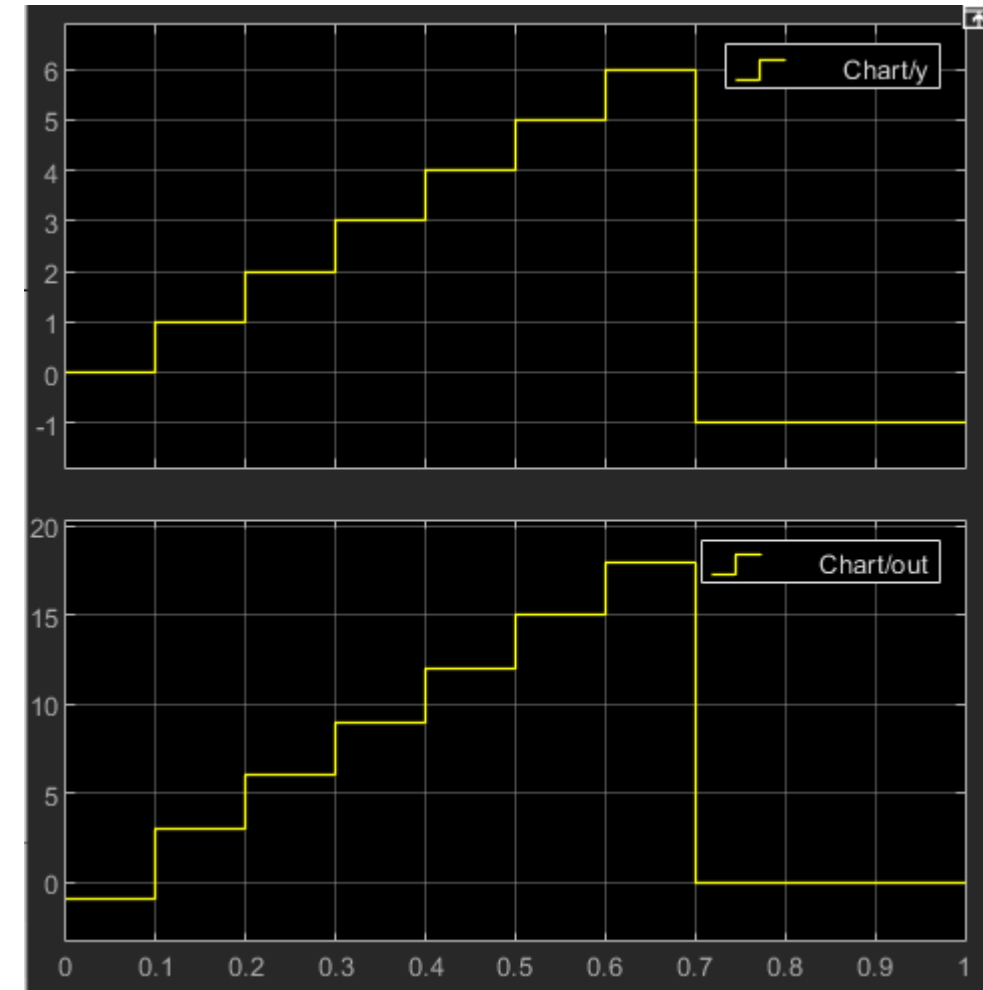
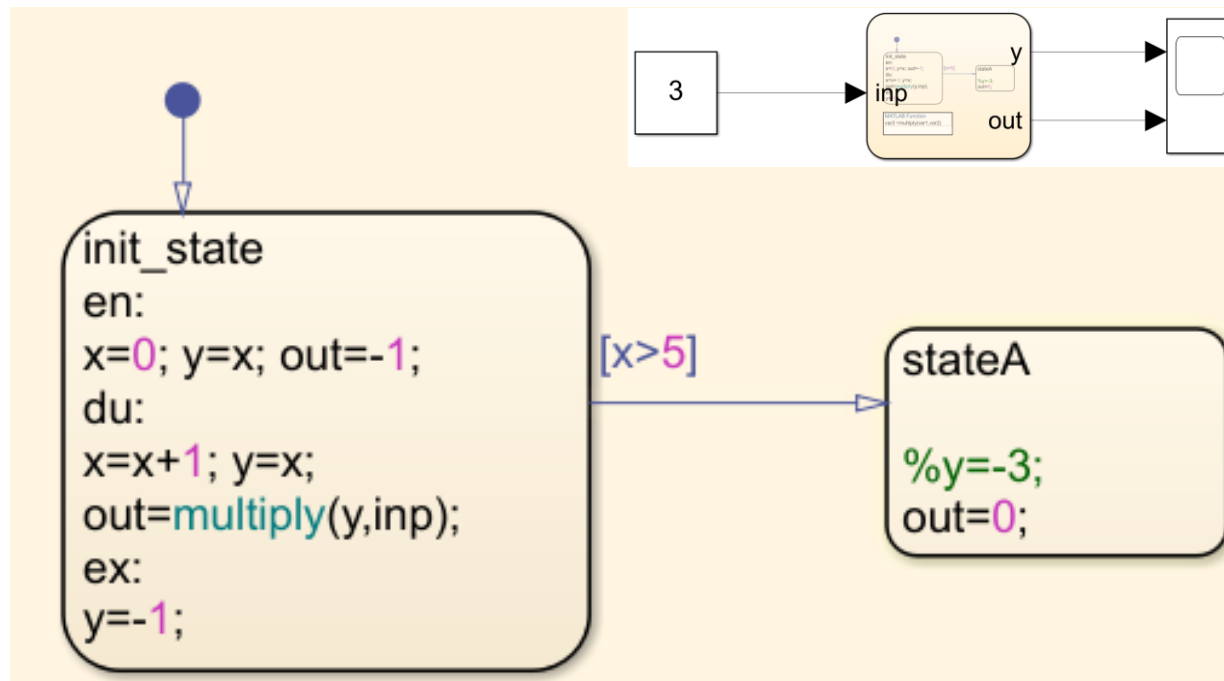
```
var3 =multiply(var1,var2)
```

stateflow_ex2 ▶ Chart ▶ multiply

```
1  function var3 =multiply(var1,var2)
2
3  var3=var1*var2;
```

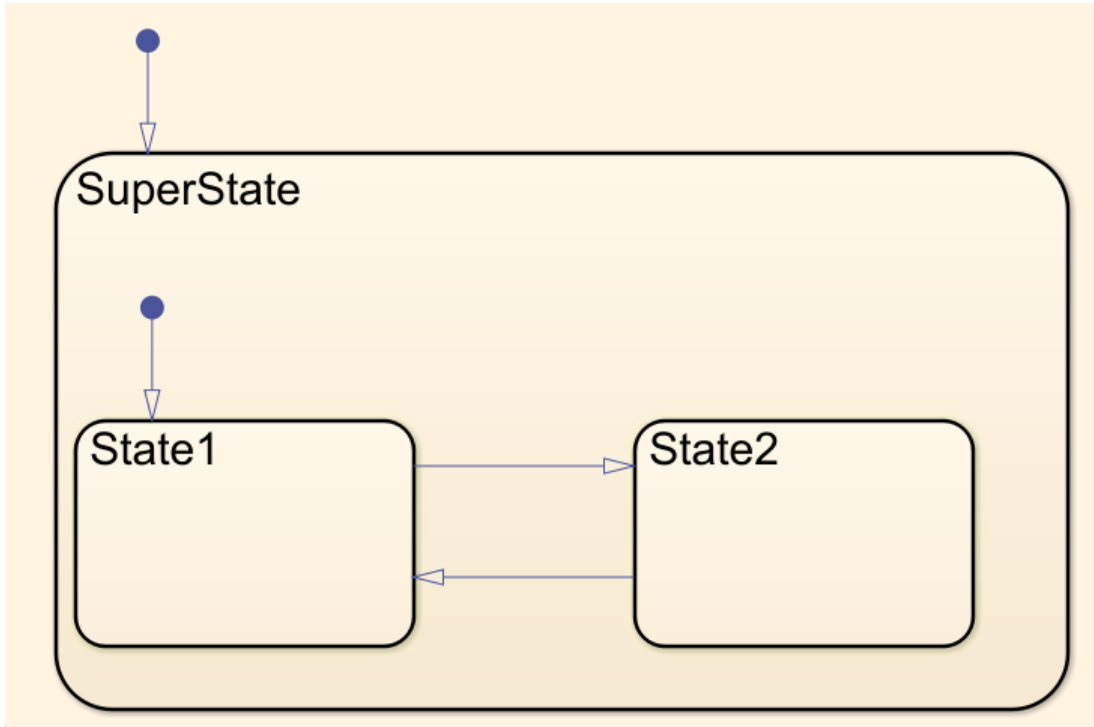
FSM using Stateflow

#example 2



FSM using Stateflow

State Hierarchy & parallelization (decomposition)

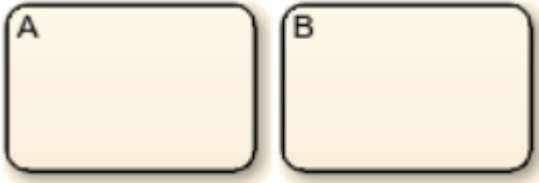


Hierarchy:

- When a parent state becomes active, one of its child states also becomes active.
- When the parent state becomes inactive, all of its child states become inactive.

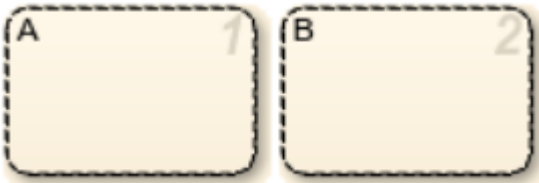
FSM using Stateflow

State Hierarchy & parallelization (decomposition)



Series (OR) : mutually exclusive modes of operation (default).

- Only one state active or execute at the same hierarchical simultaneously.
- Stateflow represents each exclusive state by a solid rectangle.

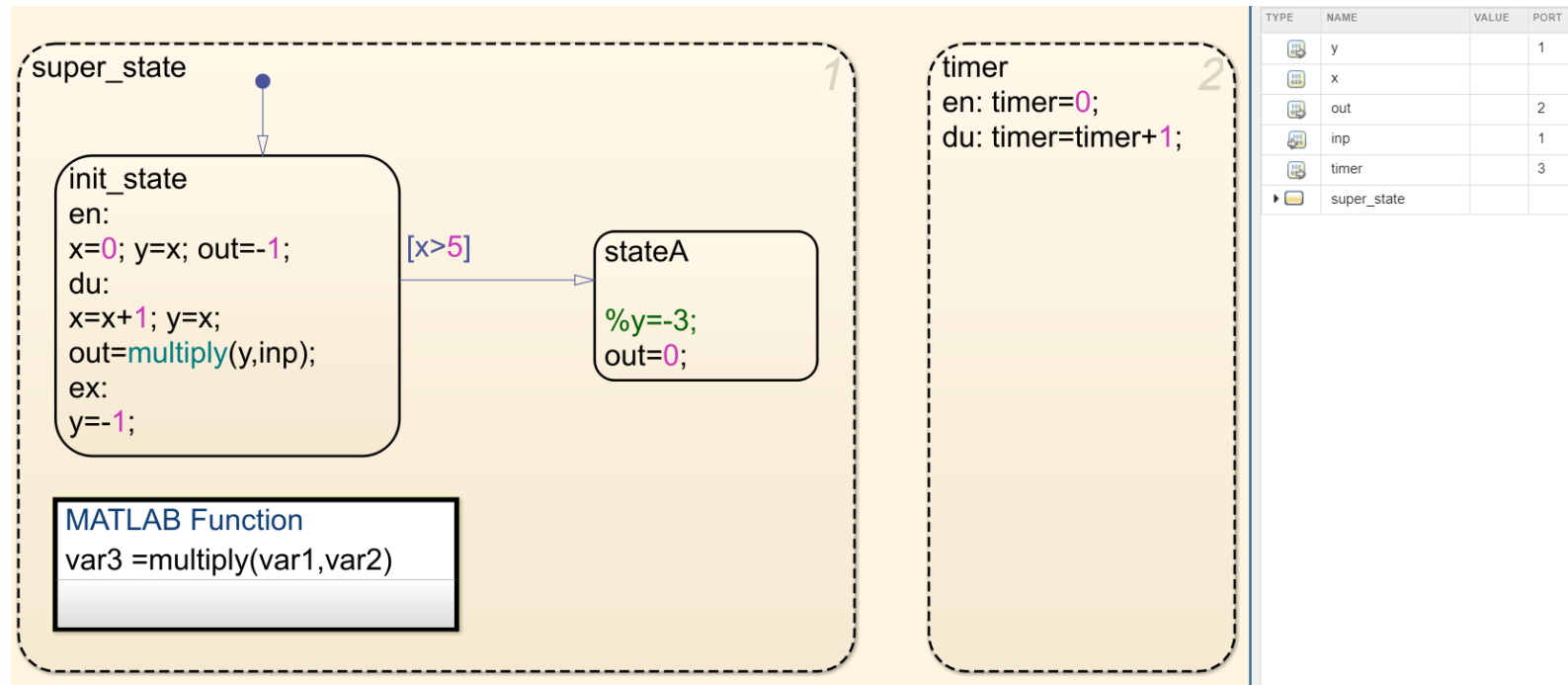


Parallels (AND) : independent modes of operation.

- Two or more parallel states can be active at the same time.
- Stateflow represents each parallel state by a dashed rectangle with a number indicating its execution order.

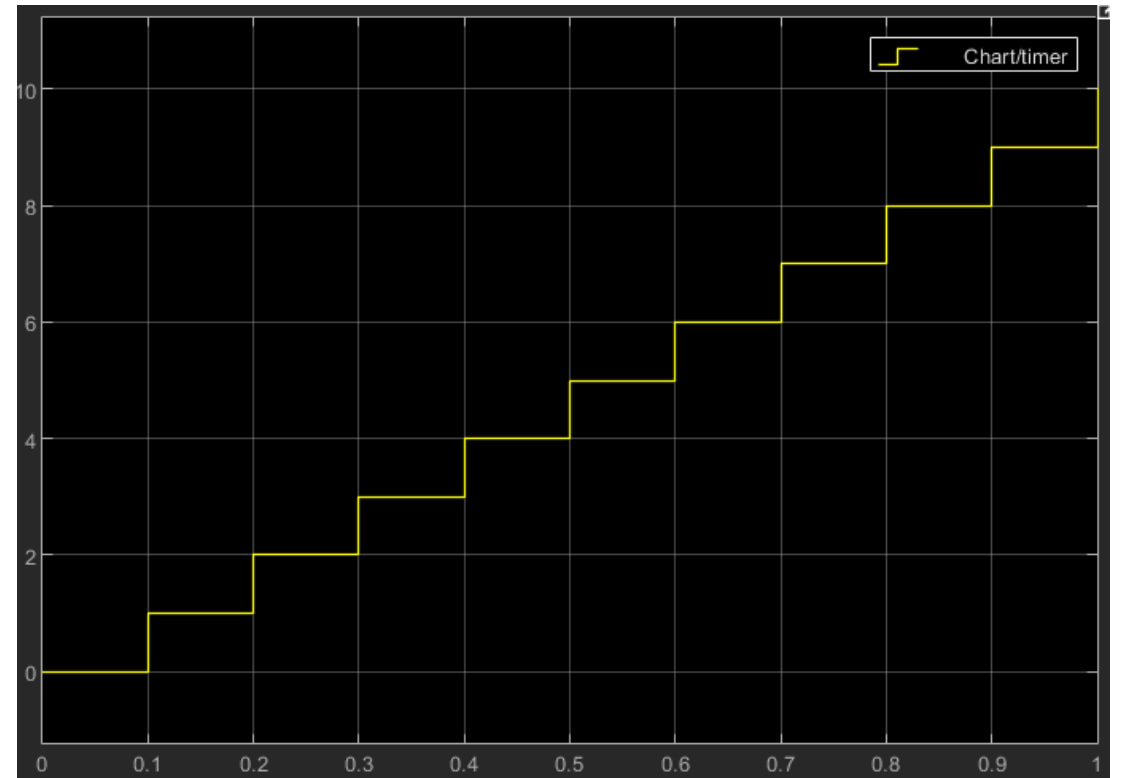
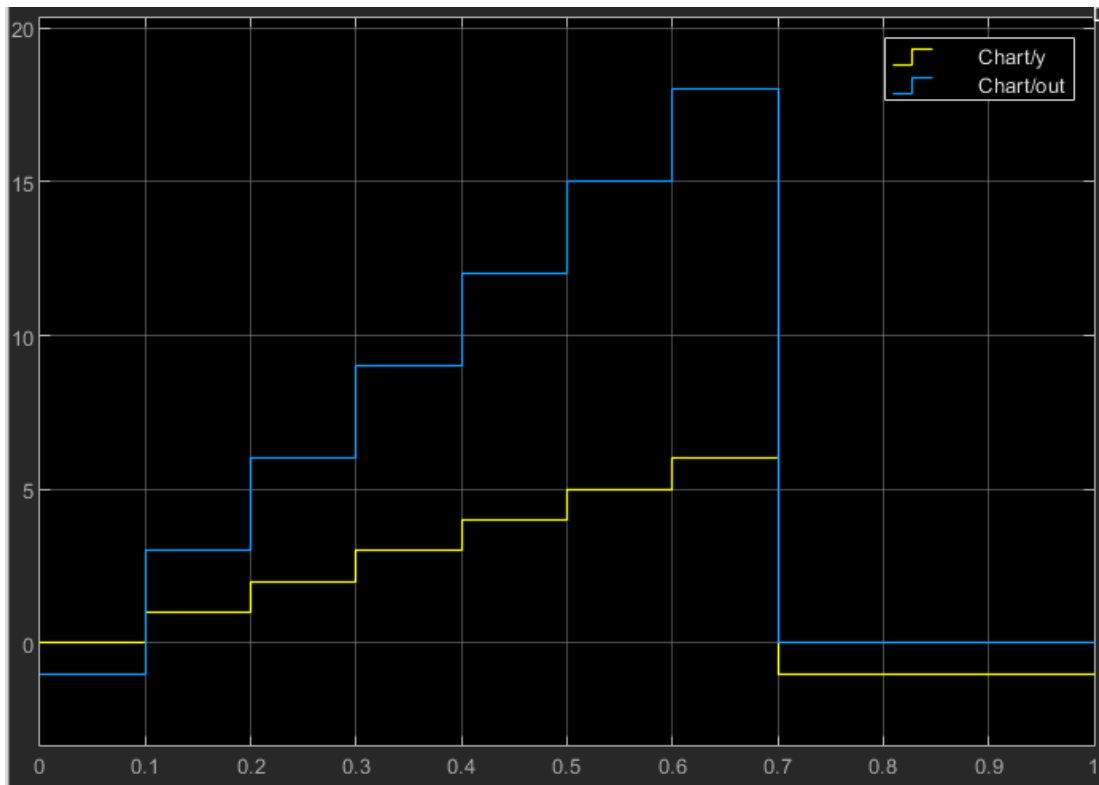
FSM using Stateflow

#example 3



FSM using Stateflow

#example 3



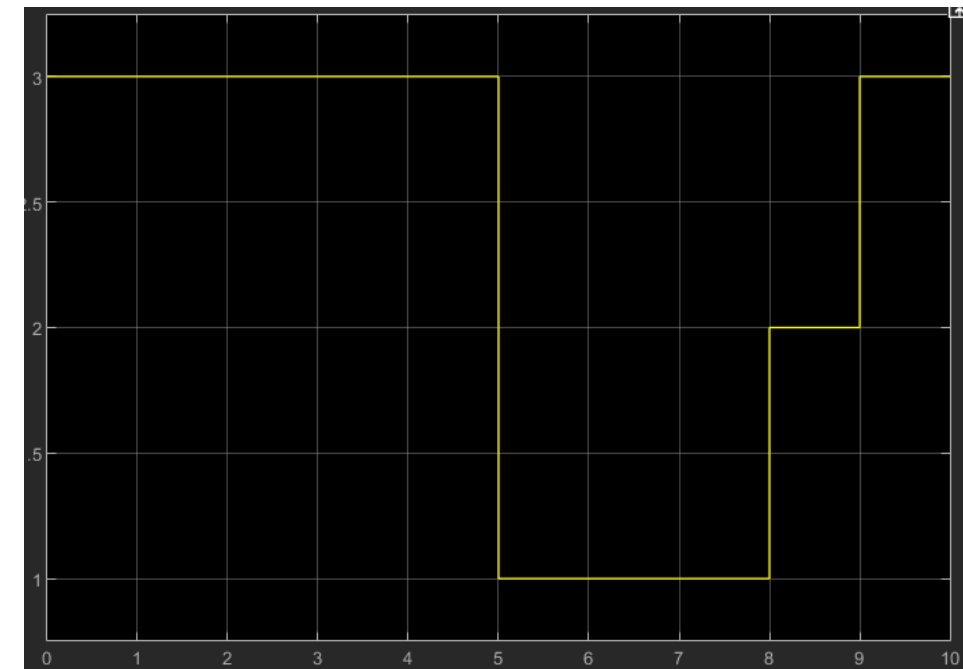
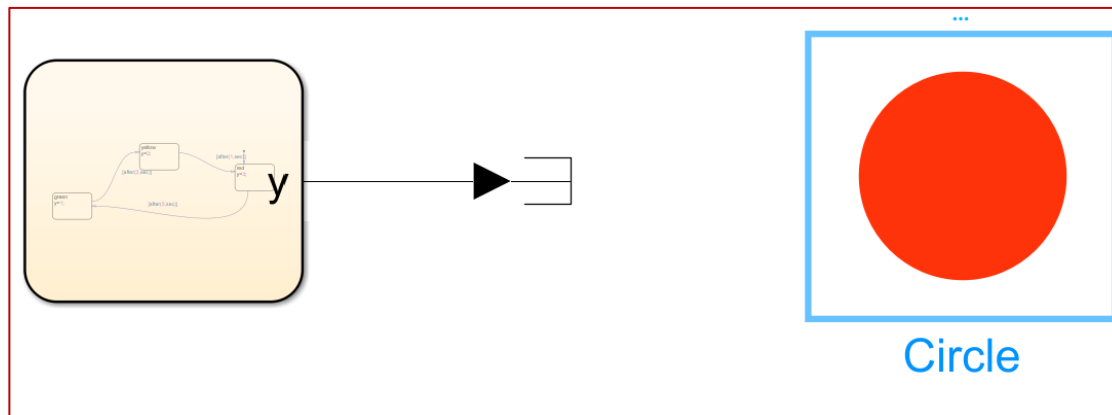
FSM using Stateflow

temporal logic operators

after(n,sec)	Returns true if n seconds of simulation time elapsed since the activation of the associated state.
elapsed(sec)	Returns the seconds of simulation time elapsed since the activation of the associated state.
duration(C)	Returns the seconds of simulation time that have elapsed since the Boolean condition C becomes true.

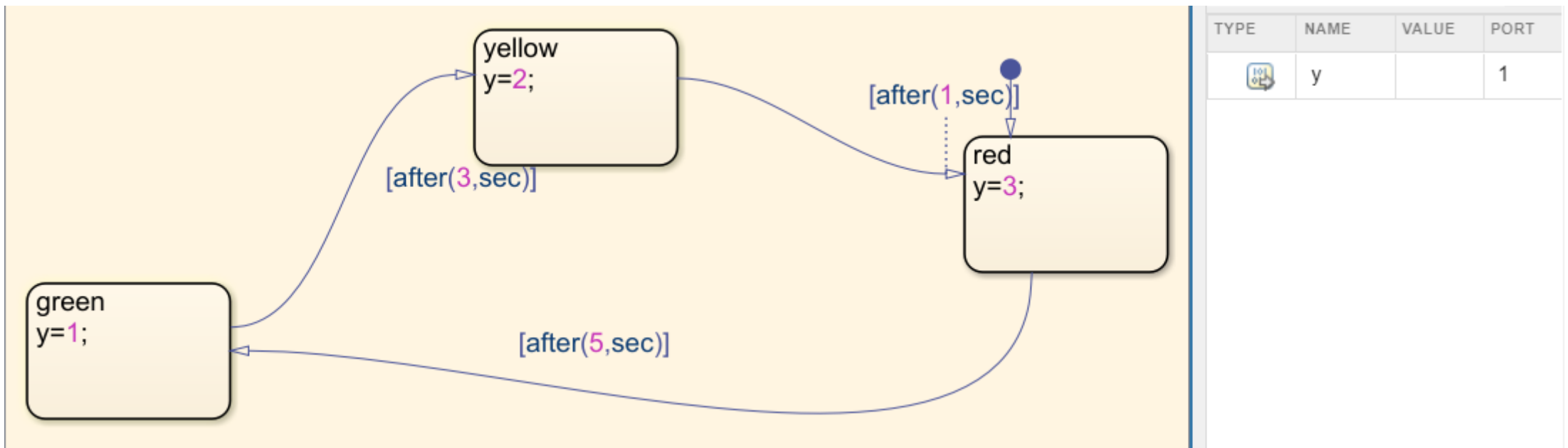
FSM using Stateflow

example 4



FSM using Stateflow

example 4





Assignment IV

What we expect from you

Starting from the “parking gate control” example:

1. Analyze and learn how “vehicle FSM” provided works
2. Define “raise” & “lower” functions of the gate according to specifics (next page)
3. Define a FSM to model “parking gate control”
4. Integrate the 2 FSM and run it for 100s

What we expect from you

“raise” & “lower” functions:

1. Max angular speed = 4 deg/sec
2. Down=0°; up=90°
3. Max angular acc/dec = |1| deg/sec²

Results

- What is mandatory for the report?
 - Plot of travelled distance of vehicles / gate motion in time;
 - Describe FSM designed;
 - Describe “raise” & “lower” functions;
 - (comment results)

That's it for today...

See you next time!

S. Arrigoni



POLITECNICO
MILANO 1863