Lecture 11: Quantitative Model Checking

So far we only answered Yes/No questions

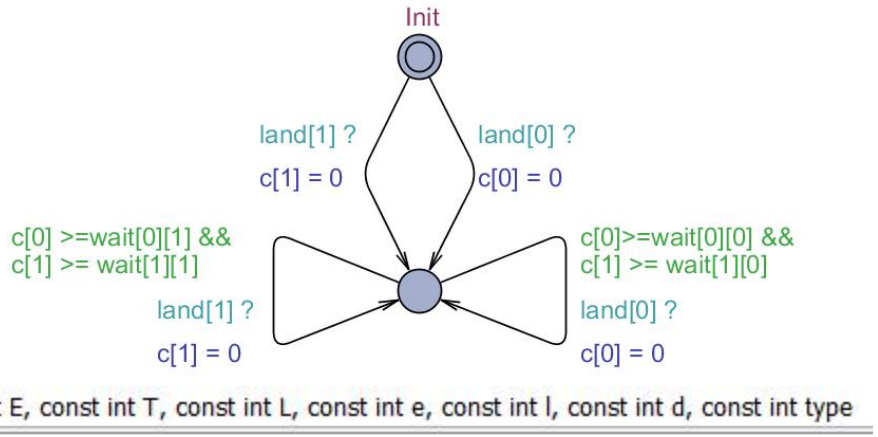
- There is need for quantitative verification
  - Quantify uncertainty
    - How often does bad events happen?
  - Quantify performance
    - What's the minimum battery consumption?
- There are tools available to evaluate
  - Probability
  - Cost/reward

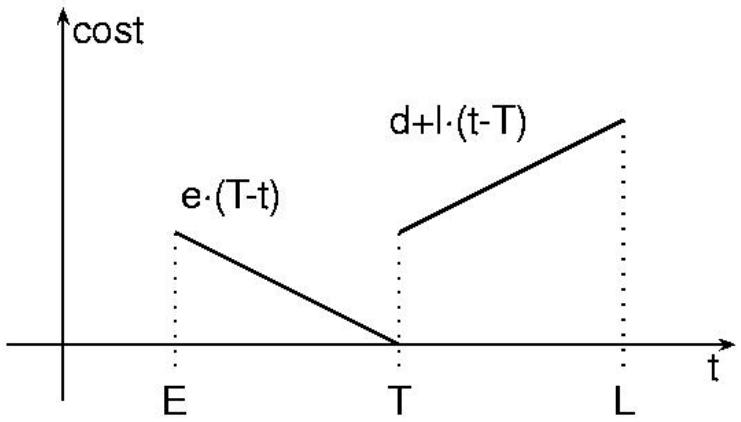
UPPAAL Tool Family

- UPPAAL CORA
  - Cost Optimal Reachability Analysis
- UPPAAL SMC
  - Statistical Model Checking
- UPPAAL TIGA
  - Controller Synthesis

UPPAAL CORA: Cost Optimal Reachability Analysis

- Linearly priced timed automata (LPTA)
- Add cost/reward to each location
- Calculates the path with minimum cost
- Can be used to model power consumption, etc





E earliest landing time

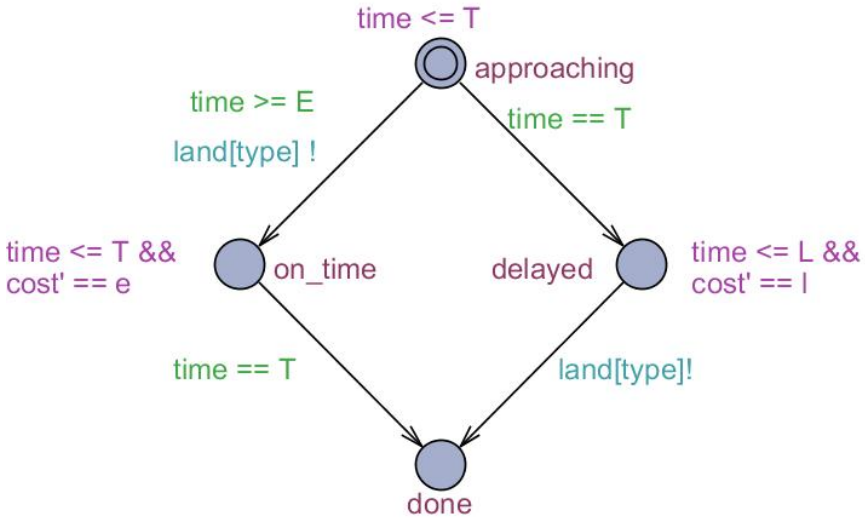
T target (cruise) landing time

L latest landing time

e early cost rate

l late cost rate

d late penalty



UPPAAL SMC

- Statistical Model Checking (SMC)
  - Non-exhaustive evaluation of the model's state space
  - Through statistical simulations within certain time bound

- Statistical Timed Automata
  - Resolve non-determinism with stochastic behaviors
  - Based on Monte Carlo Simulation

Monte Carlo Simulation

Suppose you timed 20 athletes running the 50m dash and tallied the information into the four time intervals below.

You then count the tallies and make a frequency distribution.

Then convert the frequencies into percentages.

Finally, use the percentages to develop the random number intervals.

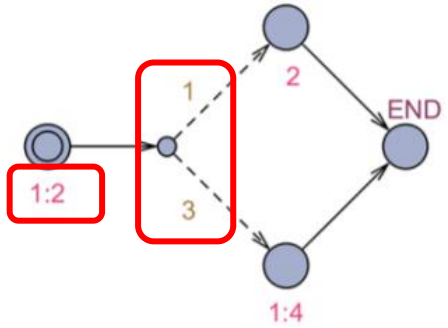
Seconds	Tallies	Frequency	%	RN Intervals
0-5.99	𝍸	4	20	01-20
6-6.99	𝍸 𝍸	10	50	21-70
7-7.99	𝍸	4	20	71-90
8 or more	||	2	10	91-100

Monte Carlo Simulation: NBA Draft

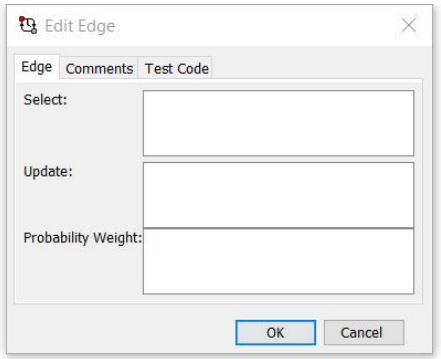
- 14 ping pong balls numbered 1 through 14 are placed in a drum.
  - C_{14}^{4}=1,001
- Prior to the Lottery, 1,000 combinations are assigned to the Lottery teams based on their order of finish during the regular season.
  - The worst team has 250 combinations (25% chance for No.1 pick)
- 4 balls are drawn from the drum with a combination
- The team that has been assigned that combination will receive the number one pick.
- The four balls are placed back in the drum and the process is repeated to determine the number two and three picks.

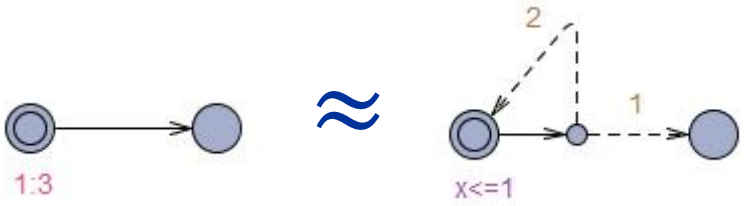
UPPAAL SMC: New Syntax

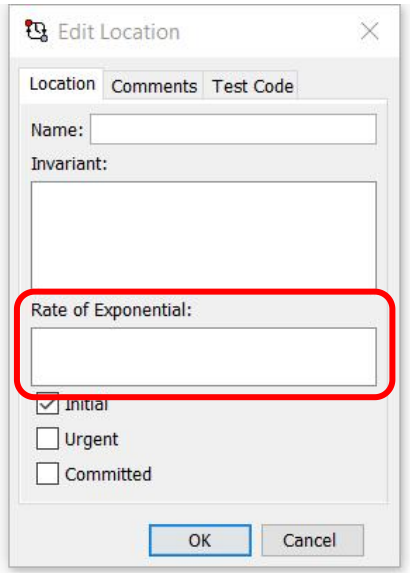
- Probabilistic transition
  - Resolves nondeterminism
  - i.e. ¼ chance going up, ¾ chance going down



- Rate of Exponential
  - "How eager you want to exit the state"

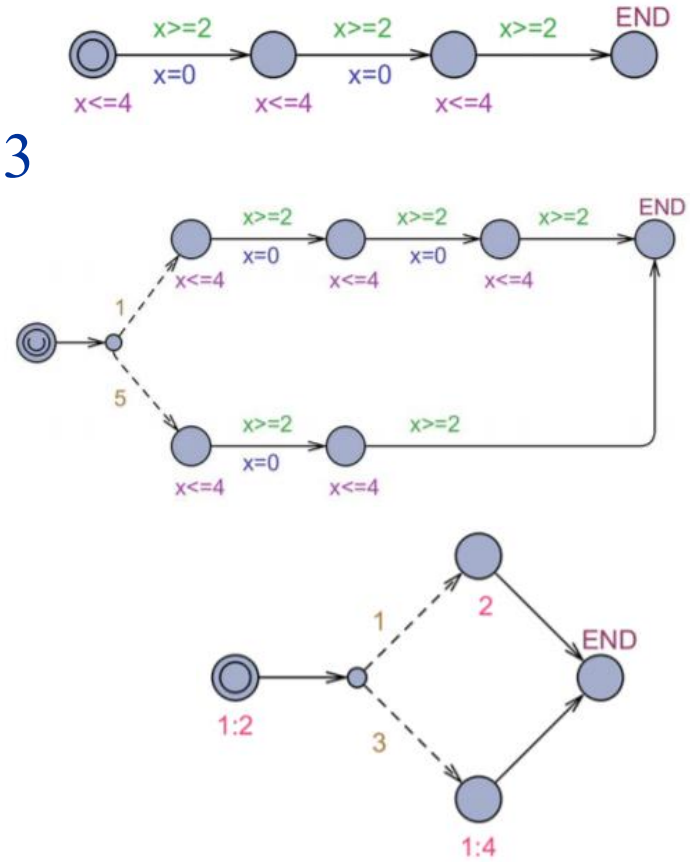


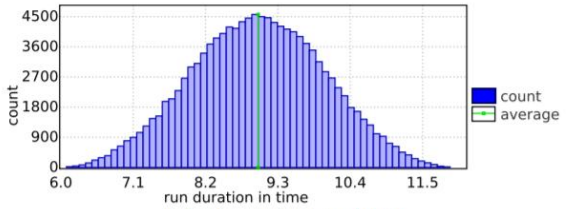




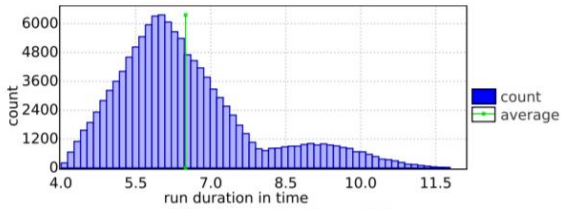
UPPAAL SMC: Semantics

- The time it takes to reach END
- Uniform distribution
  - Transition out at time 2 and time 3 are equal
- Probabilistic transition
- Exponential distribution

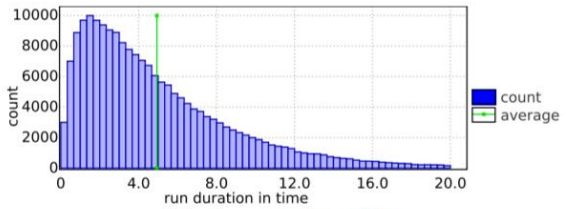




(a) A_1 arrival to END.



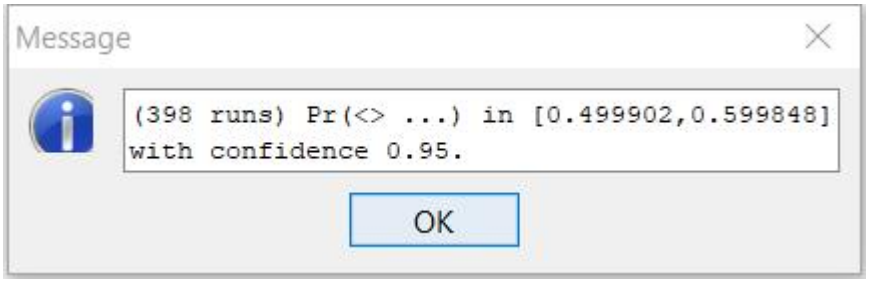
(b) A_2 arrival to END.



(c) A_3 arrival to END.

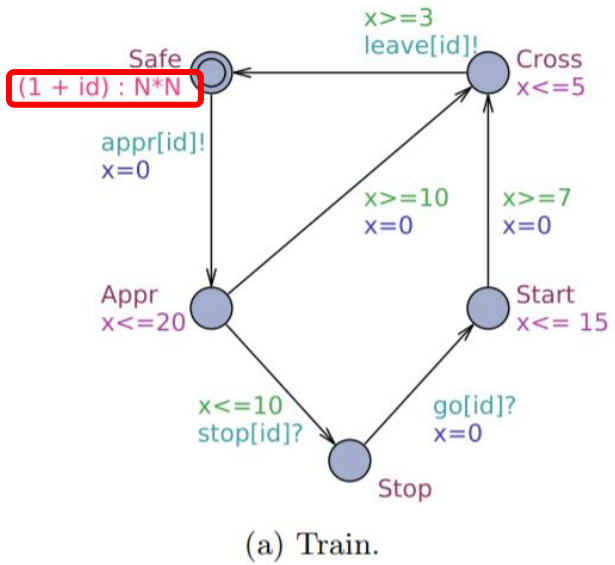
UPPAAL SMC New Queries

- Simulation
  - simulate N [<=bound] { E1,..,Ek }
- Probability Estimation
  - Pr[bound](<> psi)
- Hypothesis Testing
  - Pr[bound](psi) >= p_0
- Probability Comparison
  - Pr[bound1](psi1) >= Pr[bound2](psi2)
- Expected min/max for certain expression
  - E[bound ; N] (min/max: expr)

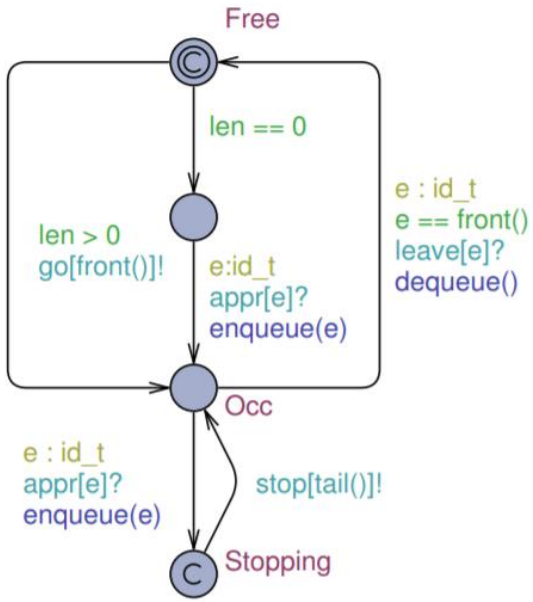


Example: Stochastic Train-Gate

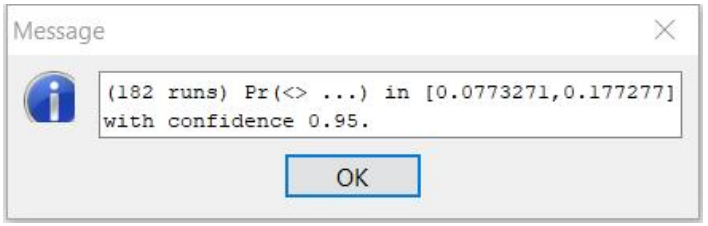
- Train with larger id is more eager to start the approach
- simulate 1 [<=300] { Train(0).Cross, Train(5).Cross, Gate.len}
- Pr[<=300](<> Gate.len < 3 and t > 20)

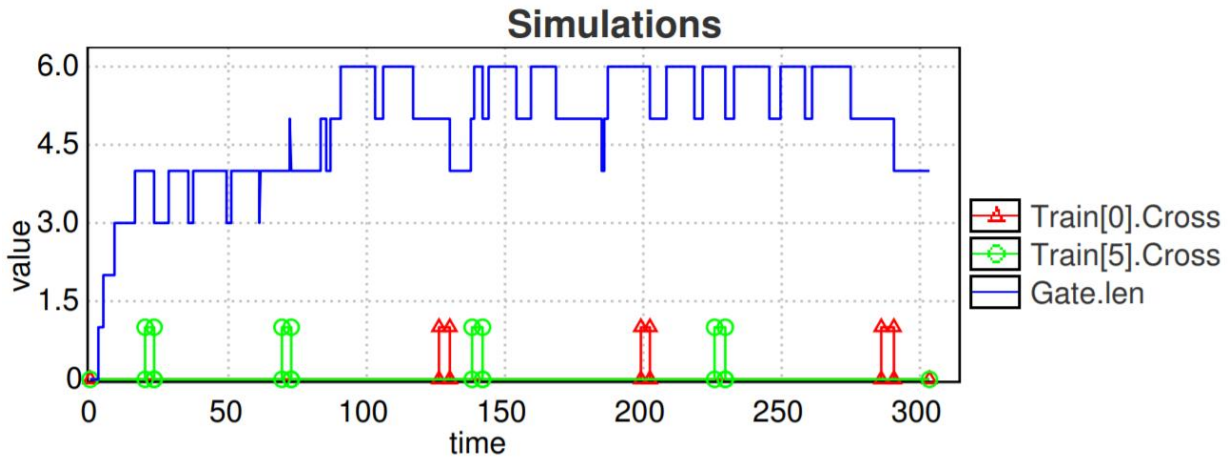


(a) Train.



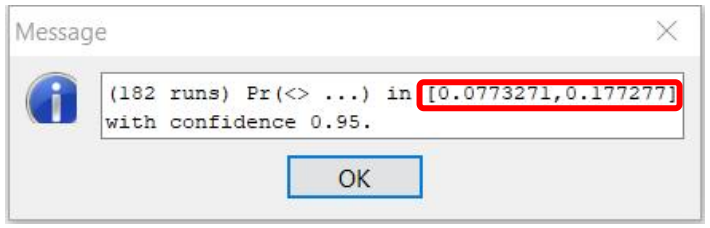
(b) Gate controller.

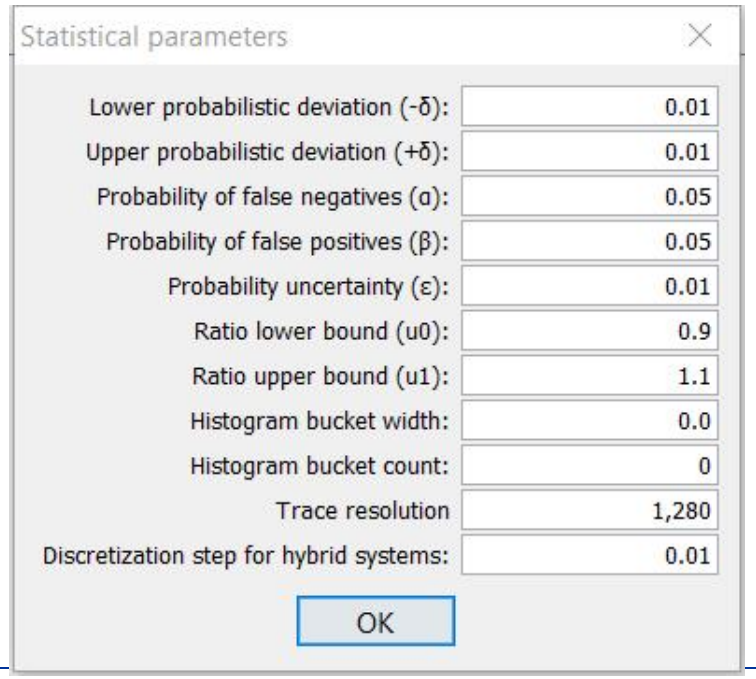




Stochastic Parameters

- δ, α, β: hypothesis testing
- ε: uncertainty for the output
  - The smaller the range, the more simulations needed
- $u0, u1$: for probability comparison

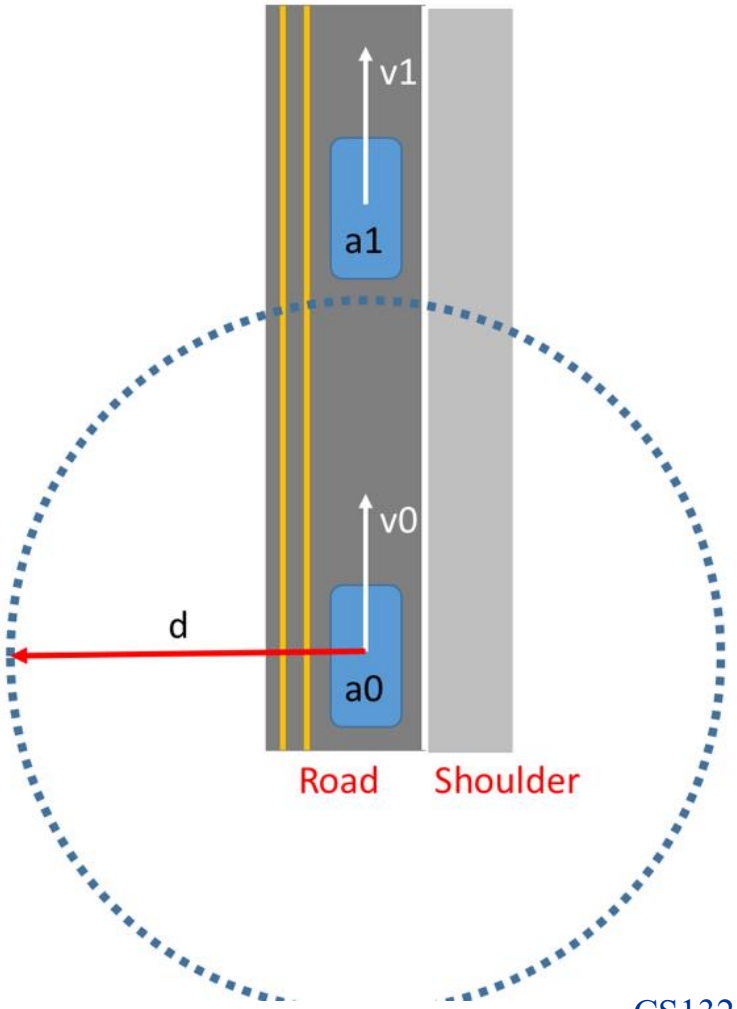




Controller Synthesis

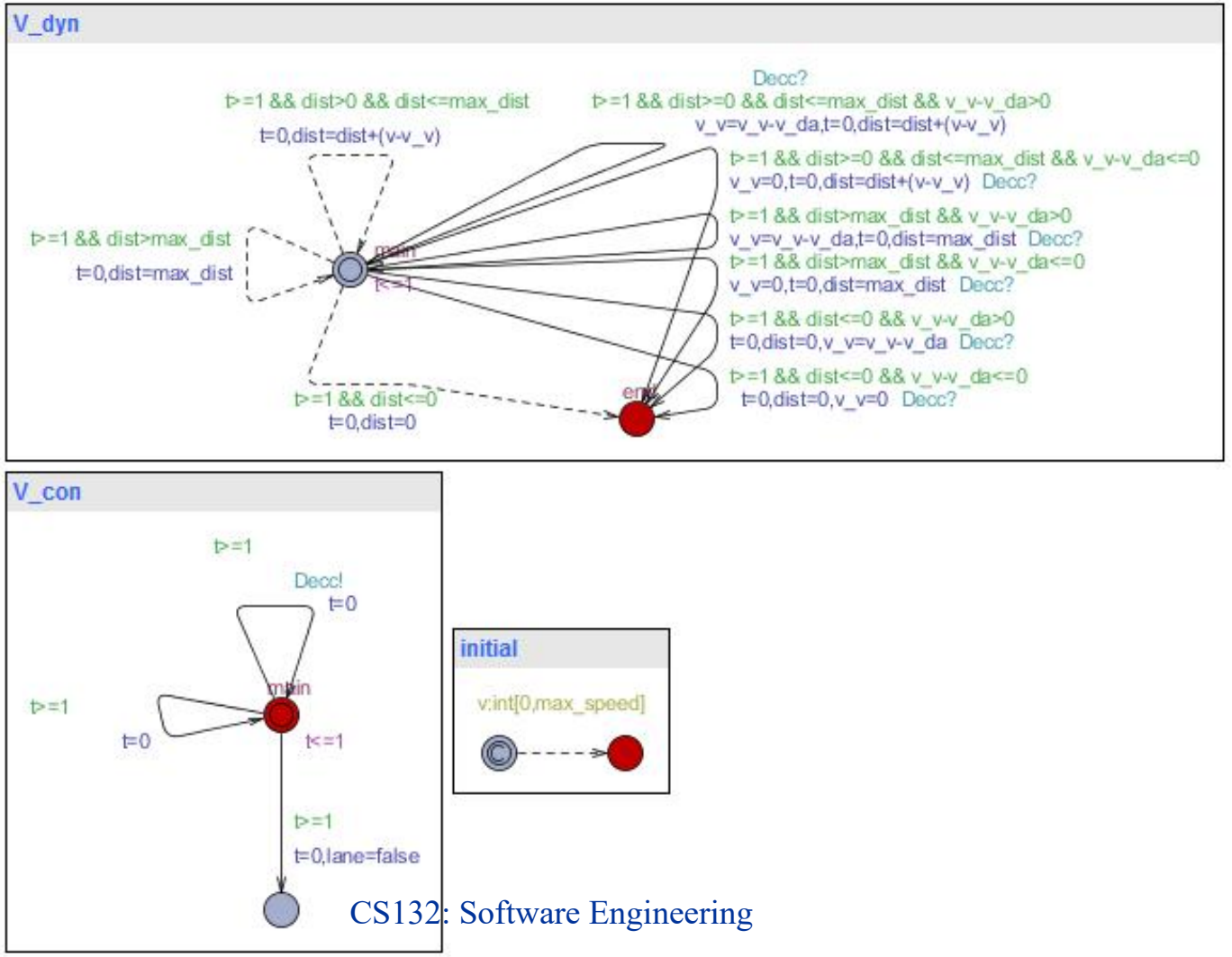
- Synthesize a controller that satisfy the requirement
- Two player game: Controller vs. Environment
- Return the winning strategy for controller

Toy Example

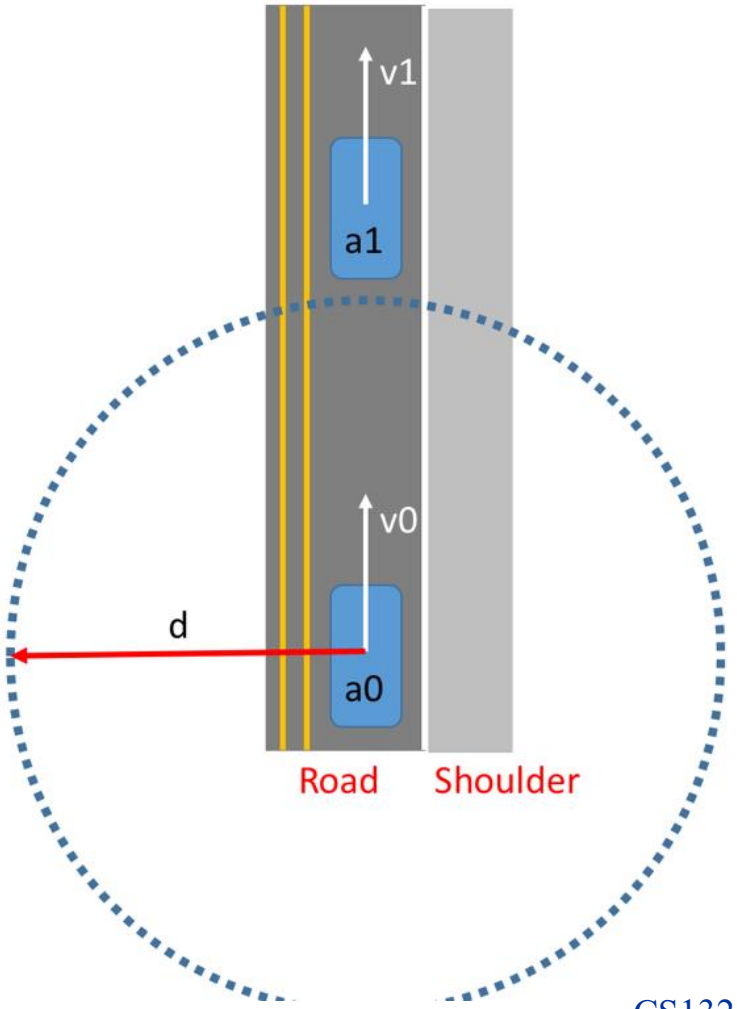


R1: No collision
R2: No driving on shoulder
R3: No hard braking

UPPAAL TIGA


V_dyn
⊳=1 && dist>0 && dist<=max_dist
t=0,dist=dist+(v-v_v)
⊳=1 && dist>max_dist
t=0,dist=max_dist
main
t<=1
⊳=1 && dist<=0
t=0,dist=0
Decc?
⊳=1 && dist>=0 && dist<=max_dist && v_v-v_da>0
v_v=v_v-v_da,t=0,dist=dist+(v-v_v)
⊳=1 && dist>=0 && dist<=max_dist && v_v-v_da<=0
v_v=0,t=0,dist=dist+(v-v_v) Decc?
⊳=1 && dist>max_dist && v_v-v_da>0
v_v=v_v-v_da,t=0,dist=max_dist Decc?
⊳=1 && dist>max_dist && v_v-v_da<=0
v_v=0,t=0,dist=max_dist Decc?
⊳=1 && dist<=0 && v_v-v_da>0
t=0,dist=0,v_v=v_v-v_da Decc?
⊳=1 && dist<=0 && v_v-v_da<=0
t=0,dist=0,v_v=0 Decc?
end
V_con
⊳=1
Decc!
t=0
main
⊳=1
t=0
t<=1
⊳=1
t=0,lane=false
initial
v:int[0,max_speed]

Naïve Solution



R1: No collision
R2: No driving on shoulder
R3: No hard braking

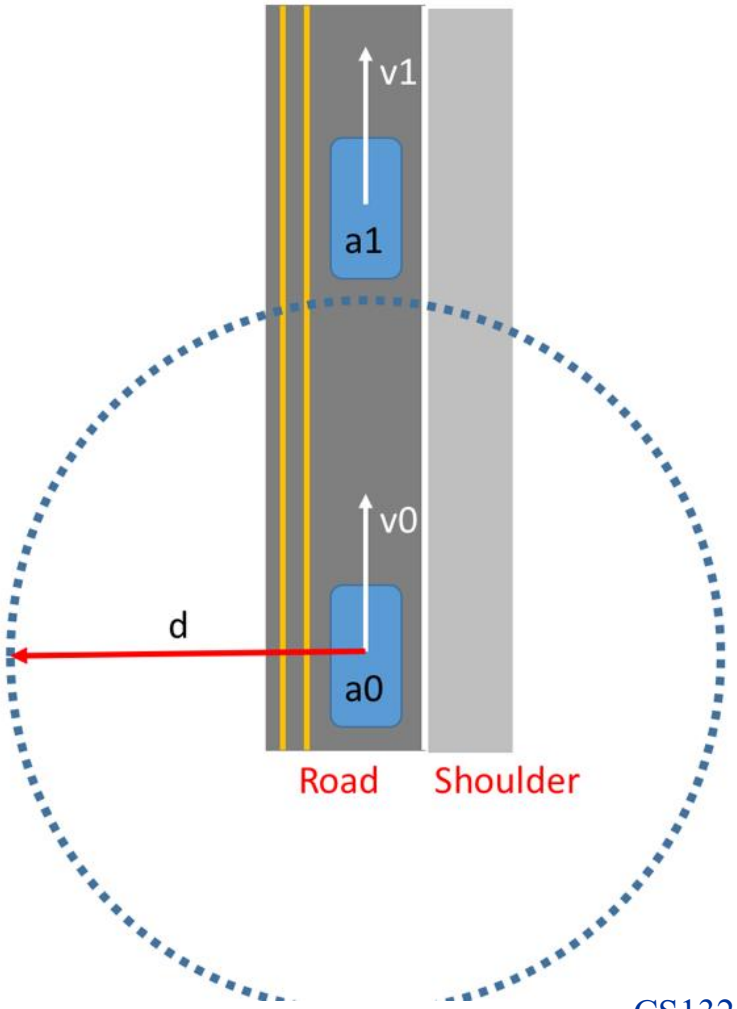
A: v1=[0,v_max]
G: R1 R2 R3

Winning Strategy

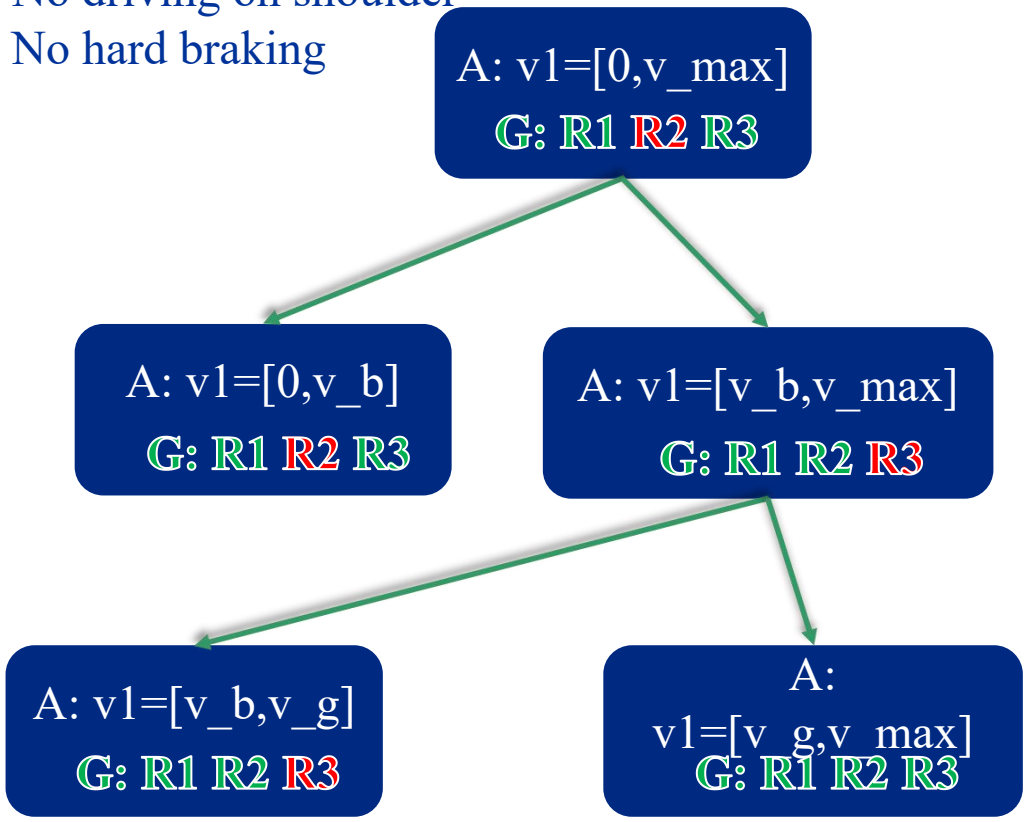
- Change lane to avoid collision

```
Strategy to avoid losing:

State: ( V_dyn.main V_con.main initial._id4 ) v_v=4 lane=1 dist=6 v=0
When you are in (V_dyn.t==1 && V_dyn.t==V_con.t && V_con.t==1), take transition
V_con.main->V_con._id0 ( t >= 1, tau, t := 0, lane := 0 )

State: ( V_dyn.main V_con.main initial._id4 ) v_v=4 lane=1 dist=2 v=0
When you are in (V_dyn.t-V_con.t==-1 && V_con.t==1), take transition V_con.main-
>V_con._id0 ( t >= 1, tau, t := 0, lane := 0 )
```

Model/Strategy Refinement



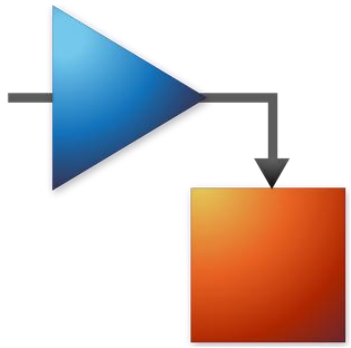
R1: No collision
R2: No driving on shoulder
R3: No hard braking



Reference

- Downlaod
  - www.uppaal.org
- Tutorials
  - On the same webpage
  - Recommended:
    - UPPAAL 4.0: Small Tutorial.
    - Uppaal SMC Tutorial

Lecture 12: Simulink & Stateflow

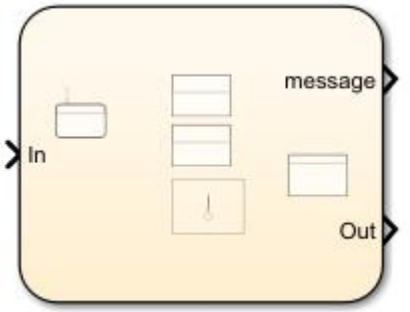


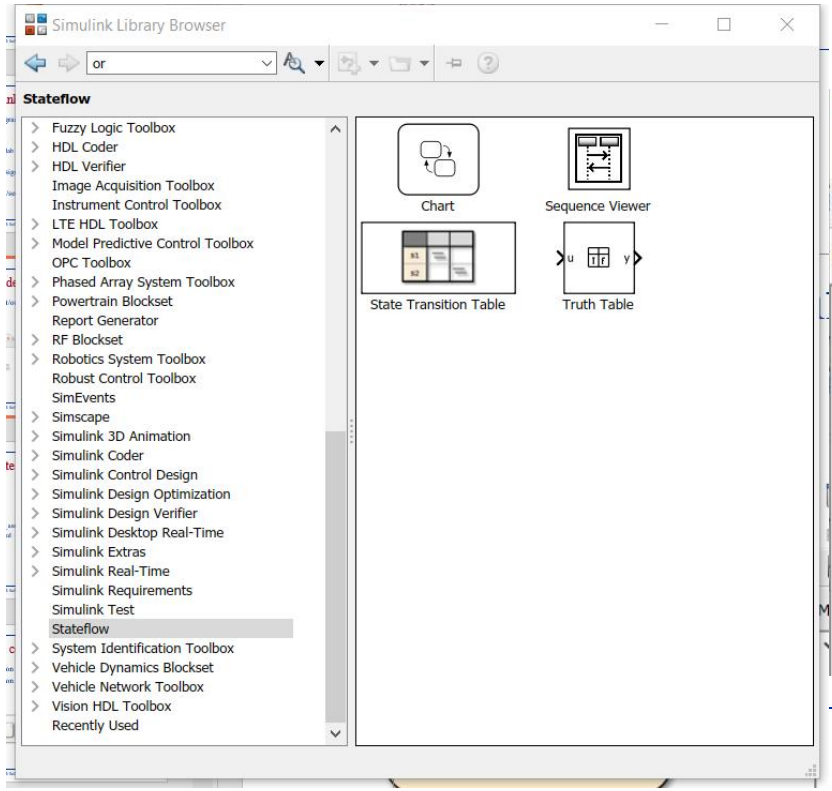
Simulink & Stateflow

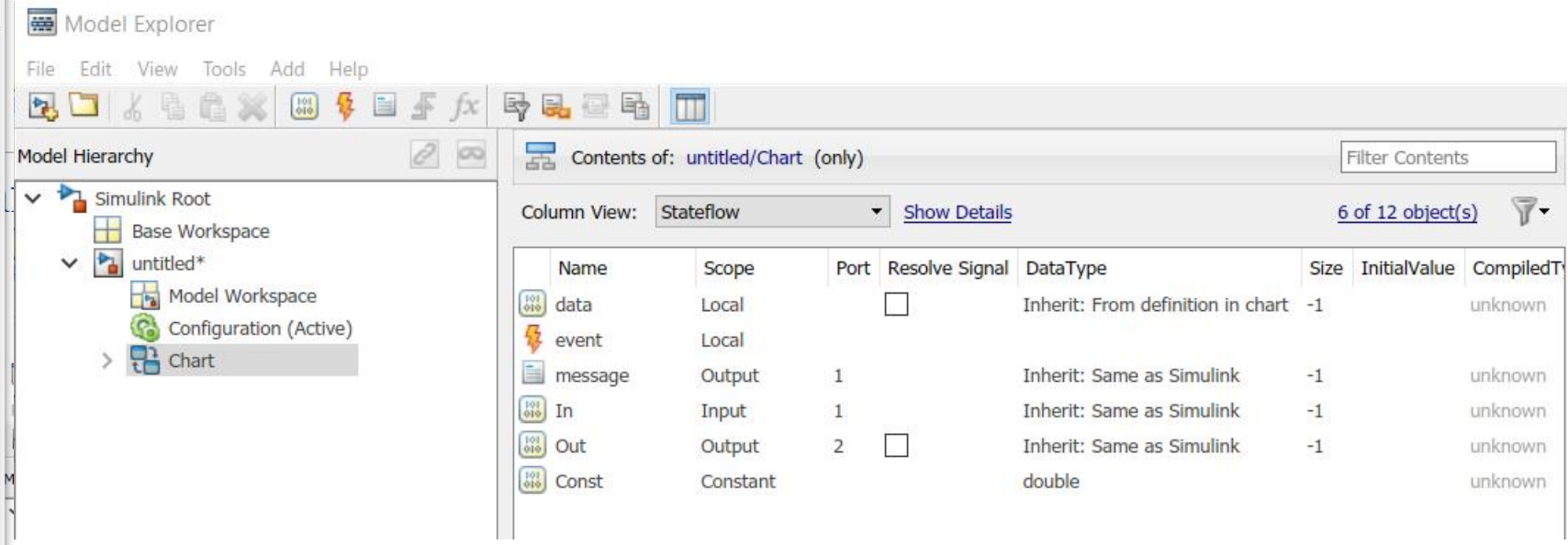
- A graphical modeling/programming language
- Developed by Mathworks
  - Highly integrated with Matlab
- Has a full model-based design toolchain
- Widely adapted by system/software developers
- Rich expressiveness

Model Explorer

- Define and configure input/output, event/message, variables.

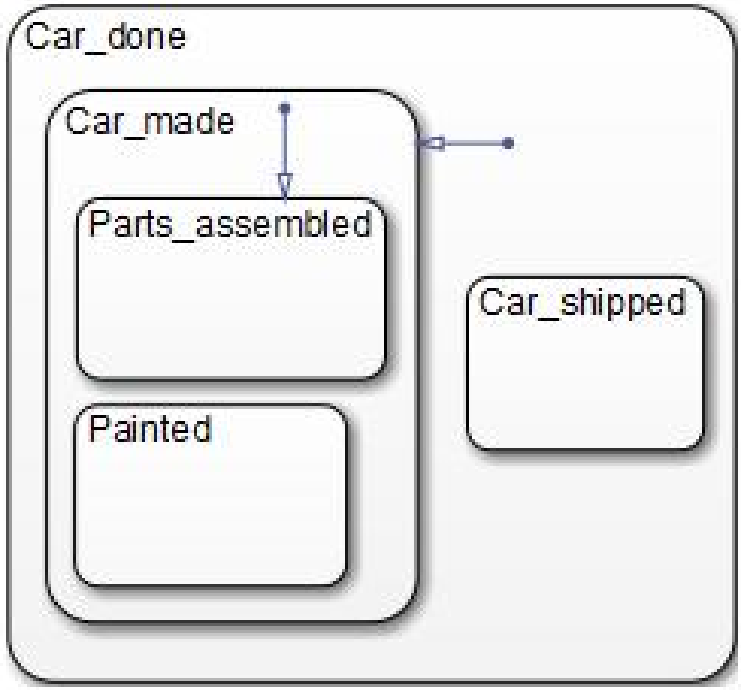






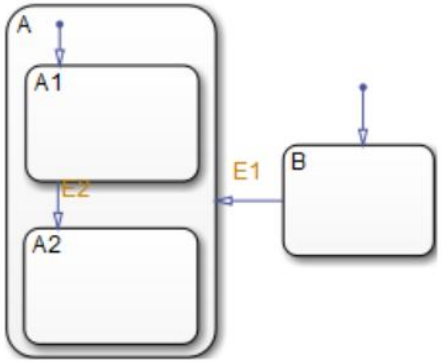
State Hierarchy

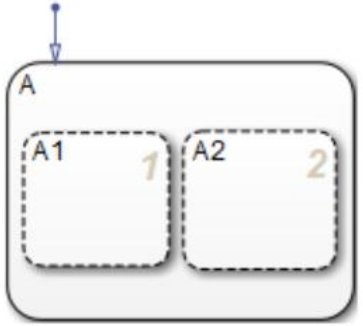
- Object oriented modeling
  - /Car_done
  - /Car_done.Car_made
  - /Car_done.Car_shipped
  - /Car_done.Car_made.Parts_assembled
  - /Car_done.Car_made.Painted

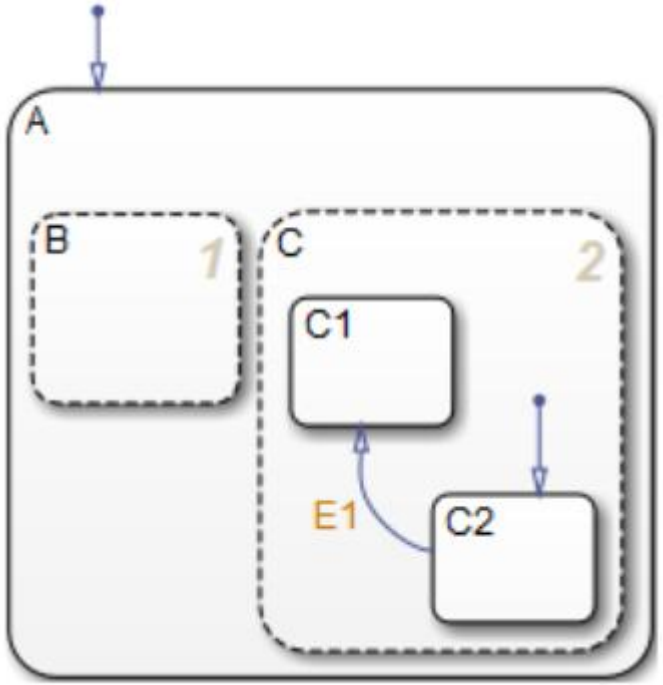


State compositions

- "OR"/exclusive composition
- "AND"/parallel composition

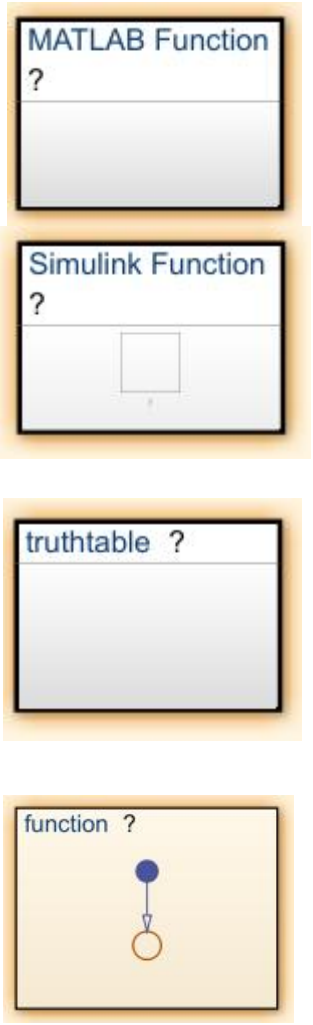






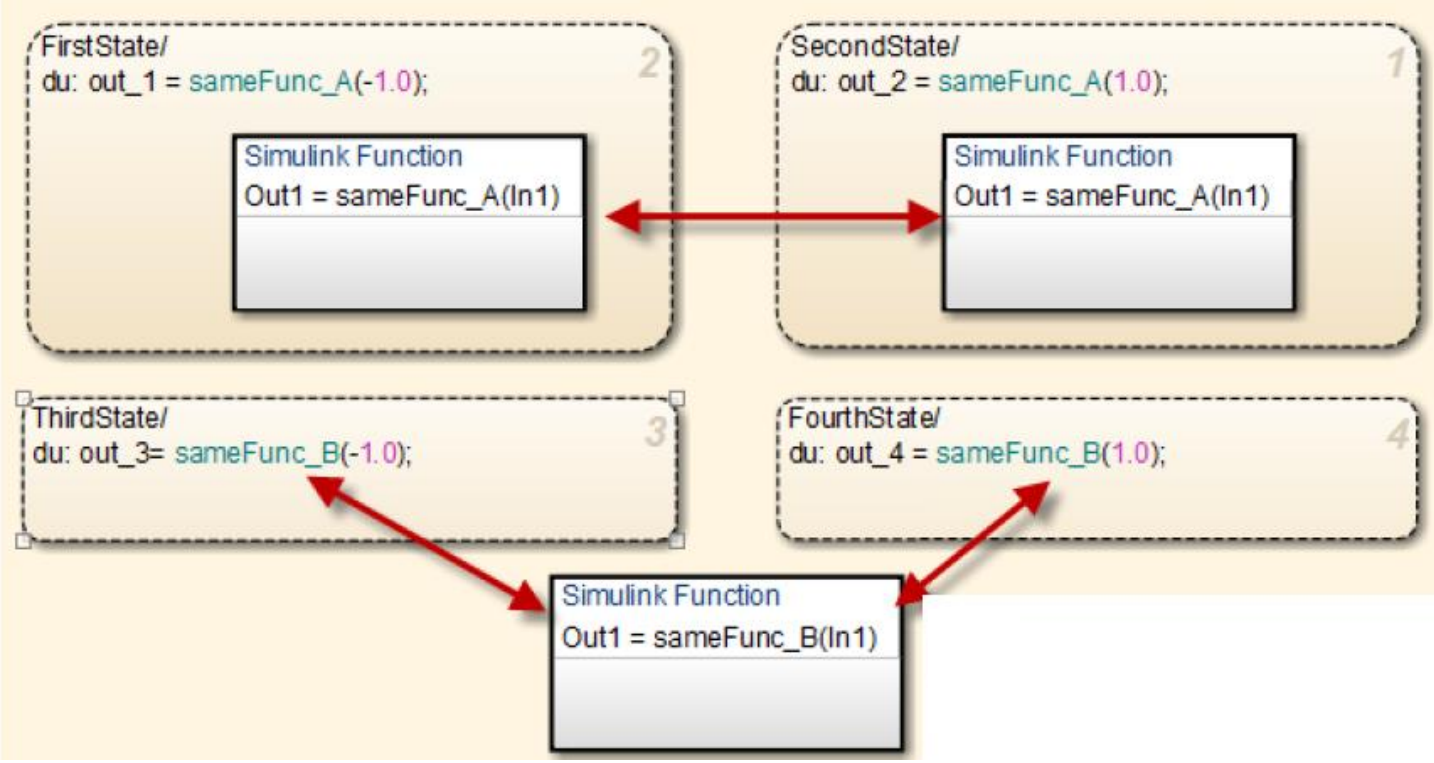
Embedded Functions

- Matlab function
- Simulink function
- Truthtable
- Graphical function



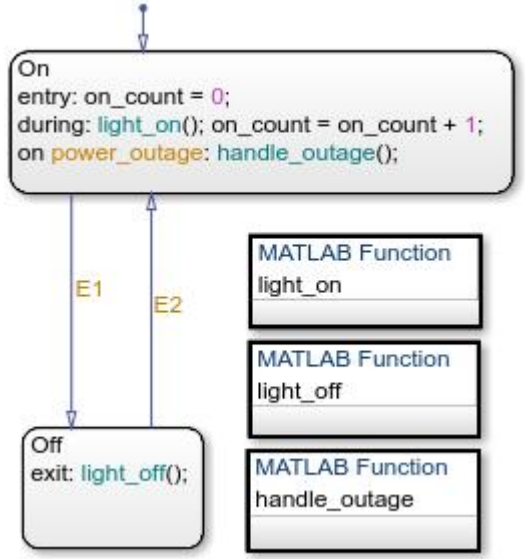
Function availability

- Only available to states at the same level



State Actions

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



Reduce redundancy

```
en:
  fcn1();
  fcn2();
du: fcn1();
ex: fcn1();
```

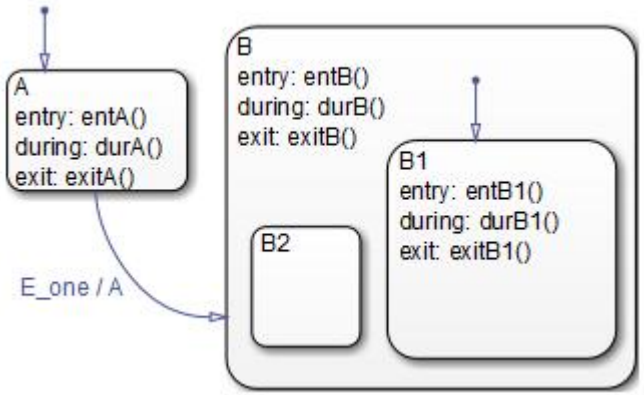
```
en, du, ex: fcn1();
en: fcn2();
```

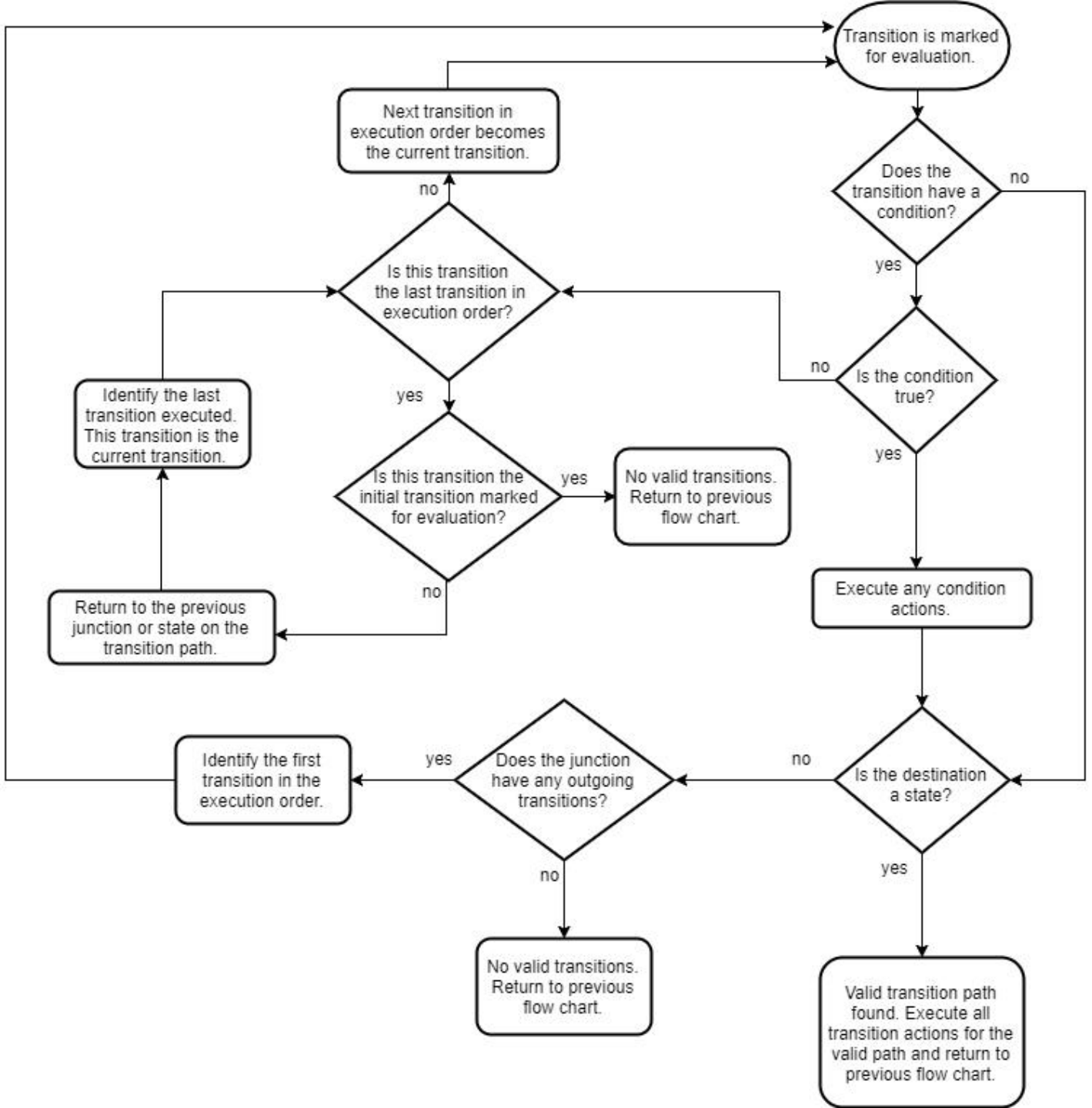
Transition

- event_or_message[condition]{condition_action}/transition_action
- Condition
  - Boolean expression that specifies that a transition path is valid if the expression is true; part of a transition label
- Condition actions
  - Executes after the condition for the transition is evaluated as true, but before the transition to the destination is determined to be valid
- Transition actions
  - Executes after the transition to the destination is determined to be valid

Default transitions

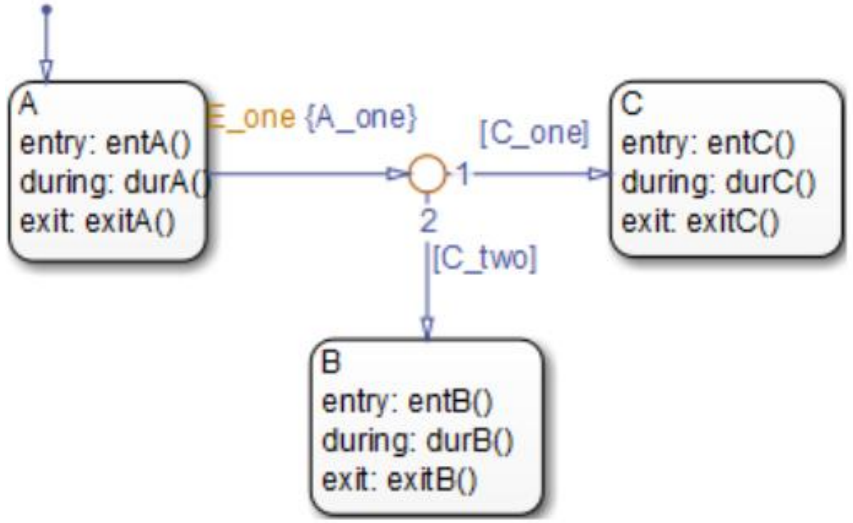
- State A is active. Event E_one occurs and awakens the chart
- State A exit actions (exitA()) execute and complete.
- State A is marked inactive.
- The transition action, A, is executed and completed.
- State B is marked active.
- State B entry actions (entB()) execute and complete.
- State B detects a valid default transition to state B.B1.
- State B.B1 is marked active.
- State B.B1 entry actions (entB1()) execute and complete.
- The chart goes back to sleep.




Transition is marked for evaluation.
Does the transition have a condition?
no
yes
Is the condition true?
no
yes
Execute any condition actions.
Is the destination a state?
no
yes
Valid transition path found. Execute all transition actions for the valid path and return to previous flow chart.
Does the junction have any outgoing transitions?
yes
Identify the first transition in the execution order.
no
No valid transitions. Return to previous flow chart.
Is this transition the last transition in execution order?
no
yes
Is this transition the initial transition marked for evaluation?
yes
No valid transitions. Return to previous flow chart.
no
Return to the previous junction or state on the transition path.
Identify the last transition executed. This transition is the current transition.
Next transition in execution order becomes the current transition.

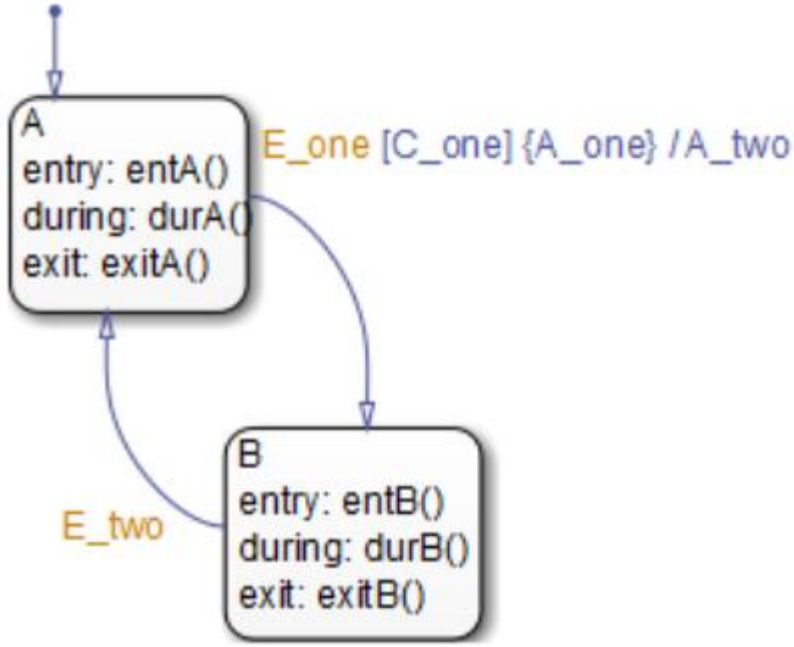
Condition Action Behavior

- E_one happened when state A is active. C_one and C_two are false
- A valid transition segment from state A to a connective junction is detected.
- The condition action A_one is immediately executed and completed. State A is still active.
- No complete transitions is valid.
- State A during actions (durA()) execute and complete.
- State A remains active.
- The chart goes back to sleep.



Condition and Transition Action Behavior

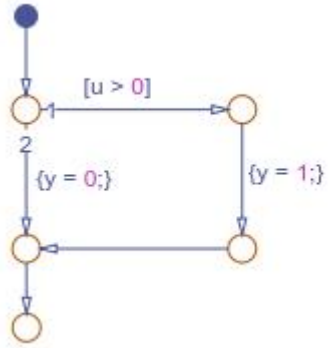
- E_one happened and awaked the chart.
- The condition C_one is true. The condition action A_one is immediately executed.
- State A is still active.
- State A exit actions (ExitA()) execute and complete.
- State A is marked inactive.
- The transition action A_two is executed.
- State B is marked active.
- State B entry actions (entB()) execute.
- The chart goes back to sleep.



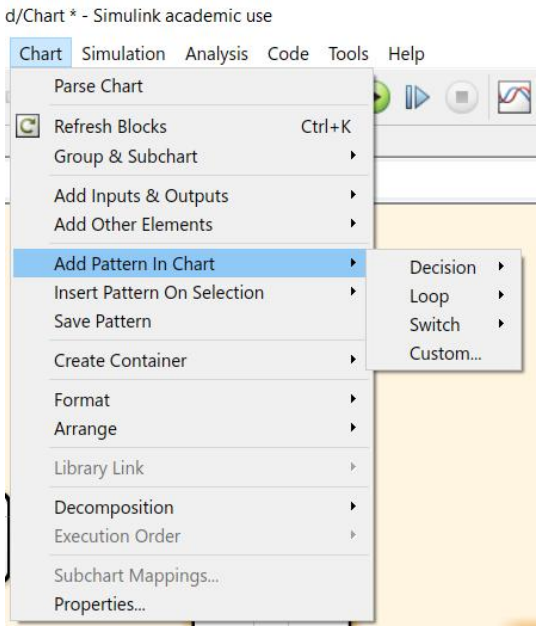
Flow chart

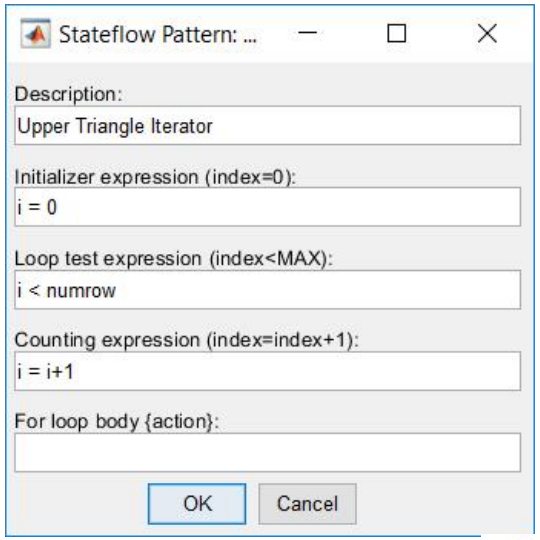
- No time consumption during execution
- Can be used for graphical function definition

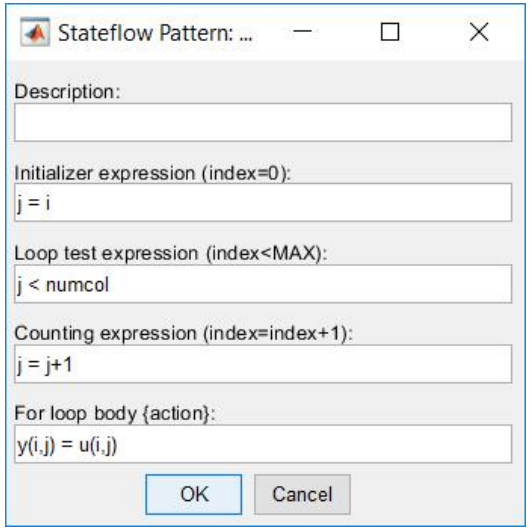
```
if u > 0
   y = 1;
else
   y = 0;
end
```

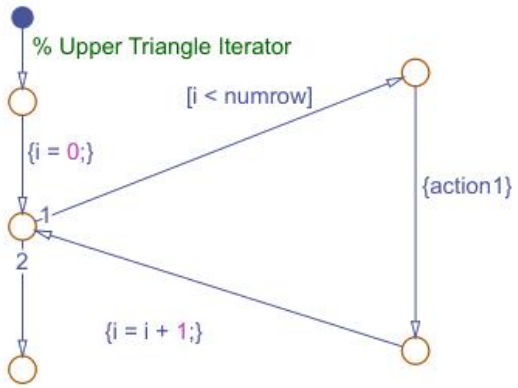


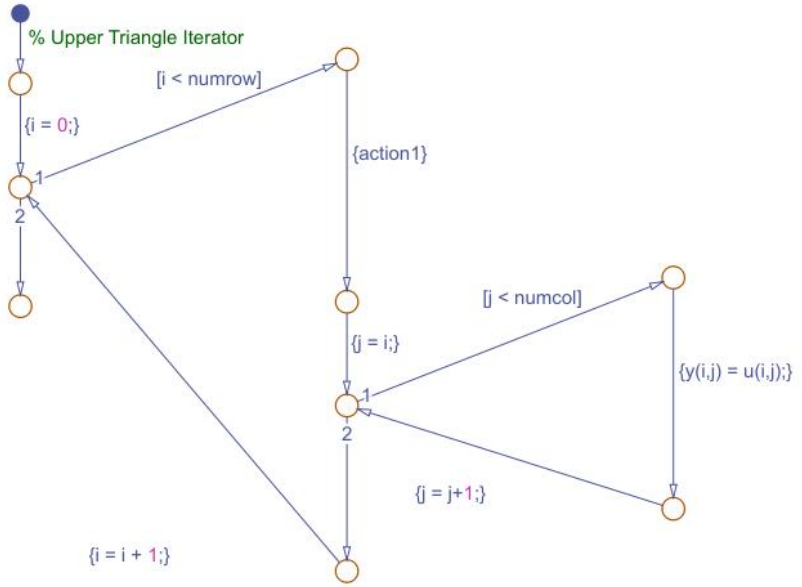
Add pattern in chart







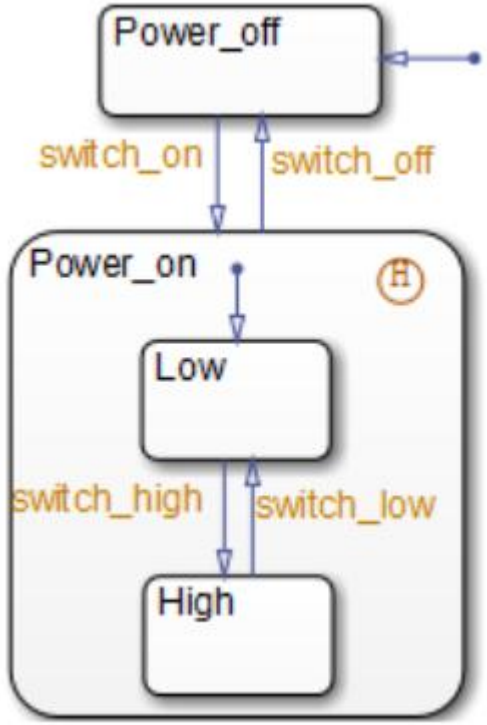




History Junction

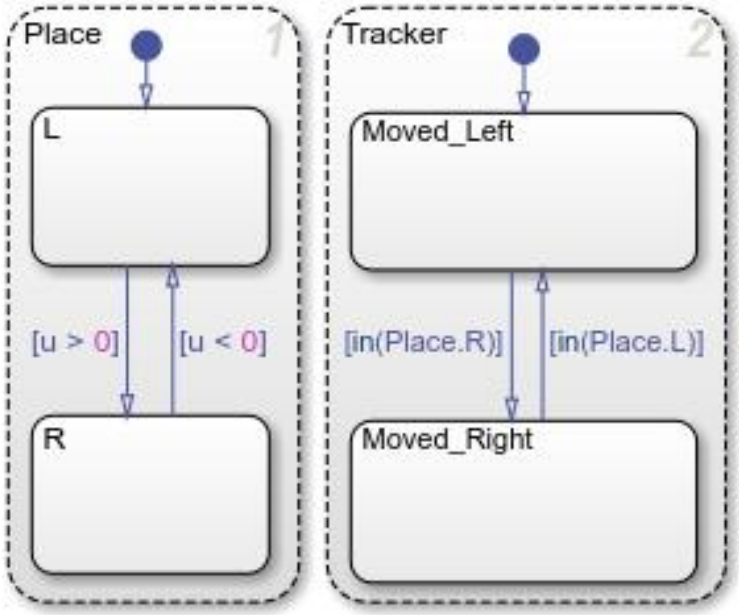
- Restores the state that is on the same level of the composite state as the history state itself

- If the system was switched off when the system was at the “High” state, when the system is switched back on, it will start from the “High” state



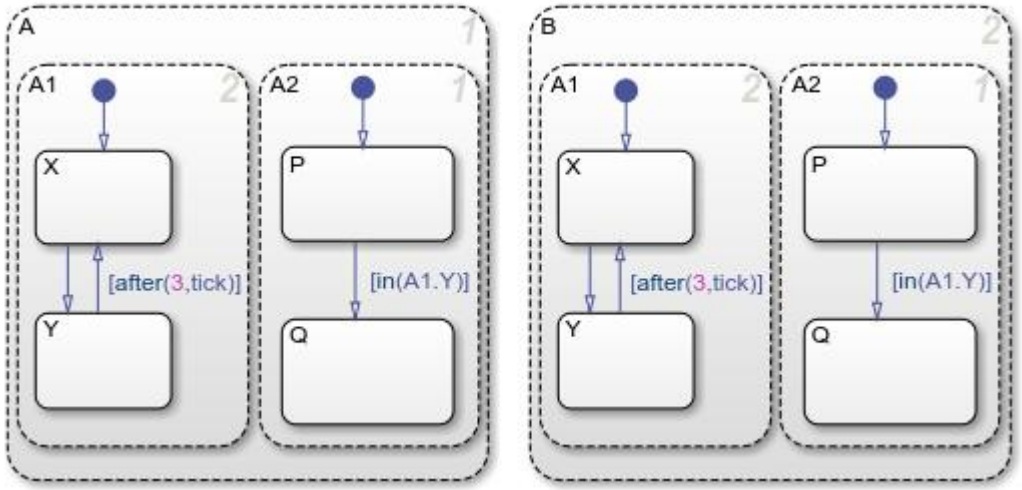
Check State Activity by Using in() Operator

- We can use in() operator to reference status of other parallel states
  - Return 1 if the referenced state is also active
- Starting point
  - If in state action, start from the containing state
  - If on transition, start from the parent state
- Search up the state hierarchy until the chart level is reached



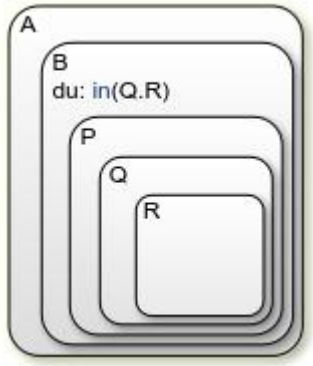
In() operator example

- In(A1.Y) in both A and B only find local copies of A1.Y
- Because at the chart level, there is no A1.Y

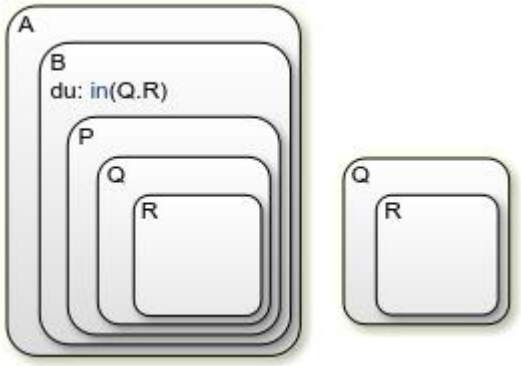


In() operator example (cont.)

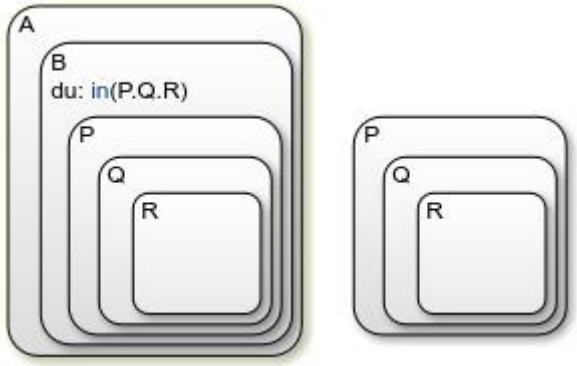
No match



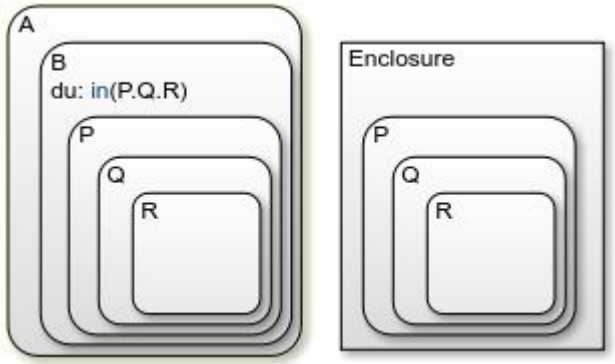
In(P.Q.R) will do
Wrong match



In(B.P.Q.R) will do
Multiple match

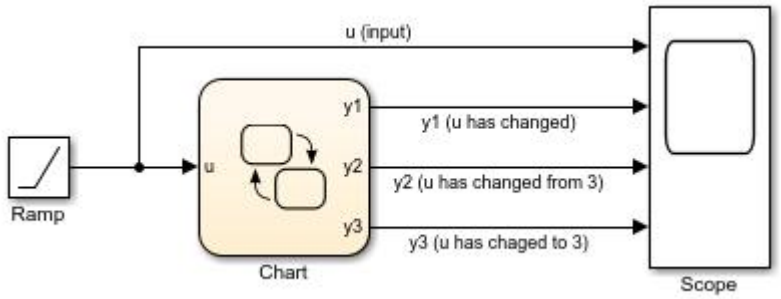


Use enclosure to ensure local match

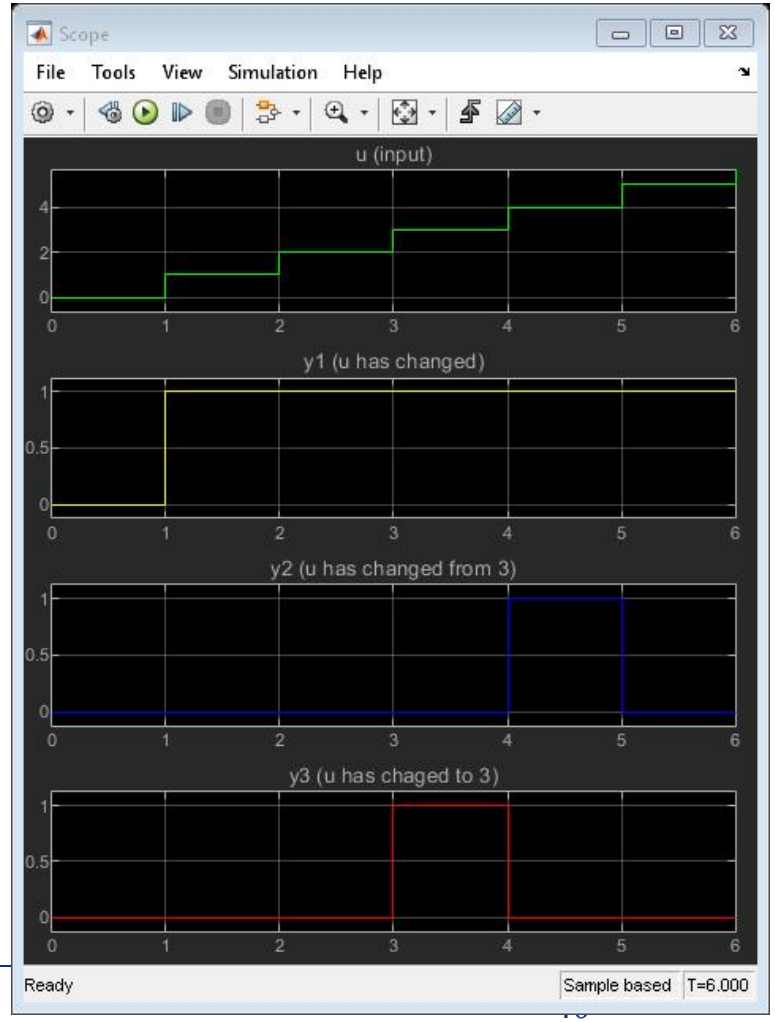


Detect data change

- hasChanged(u)
  - Detects changes in data value from the beginning of the last time step to the beginning of the current time step.
- hasChangedFrom(u,v)
  - Detects changes in data value from a specified value at the beginning of the last time step to a different value at the beginning of the current time step.
- hasChangedTo(u,v)
  - Detects changes in data value to a specified value at the beginning of the current time step from a different value at the beginning of the last time step.

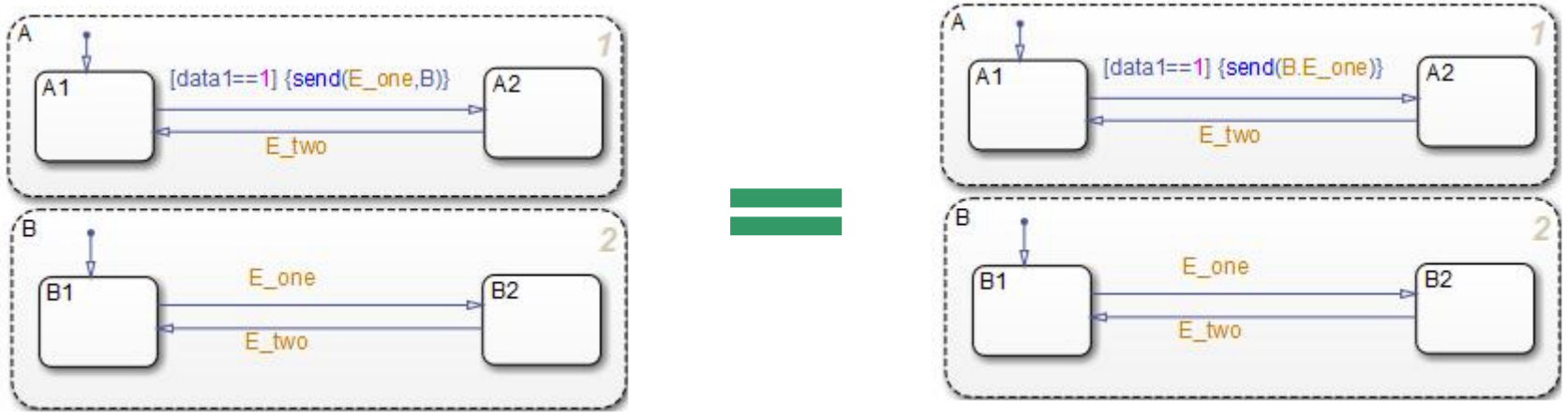



```
{
y1 = hasChanged(u);
y2 = hasChangedFrom(u,3);
y3 = hasChangedTo(u,3);
}
```



Broadcast Local Events to Synchronize Parallel States

- send(event_name,state_name)
- event_name is broadcast to its owning state (state_name) and any offspring of that state in the hierarchy.
- The receiving state must be active during the event broadcast.
- An action in one chart cannot broadcast events to states in another chart.

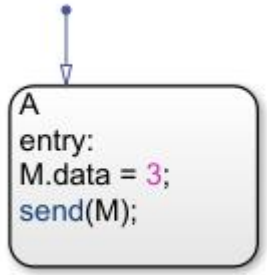


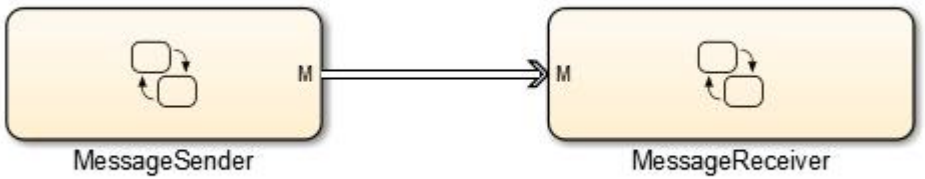
Implicit Events

- change(data_name) or chg(data_name)
  - generates a local event when writing a value to the variable data_name
  - Data_name has to be at chart level or lower
- enter(state_name) or en(state_name)
  - generates a local event when the specified state_name is entered
- exit(state_name) or ex(state_name)
  - generates a local event when the specified state_name is exited
- Tick/wakeup
  - generates a local event when the chart of the action being evaluated awakens

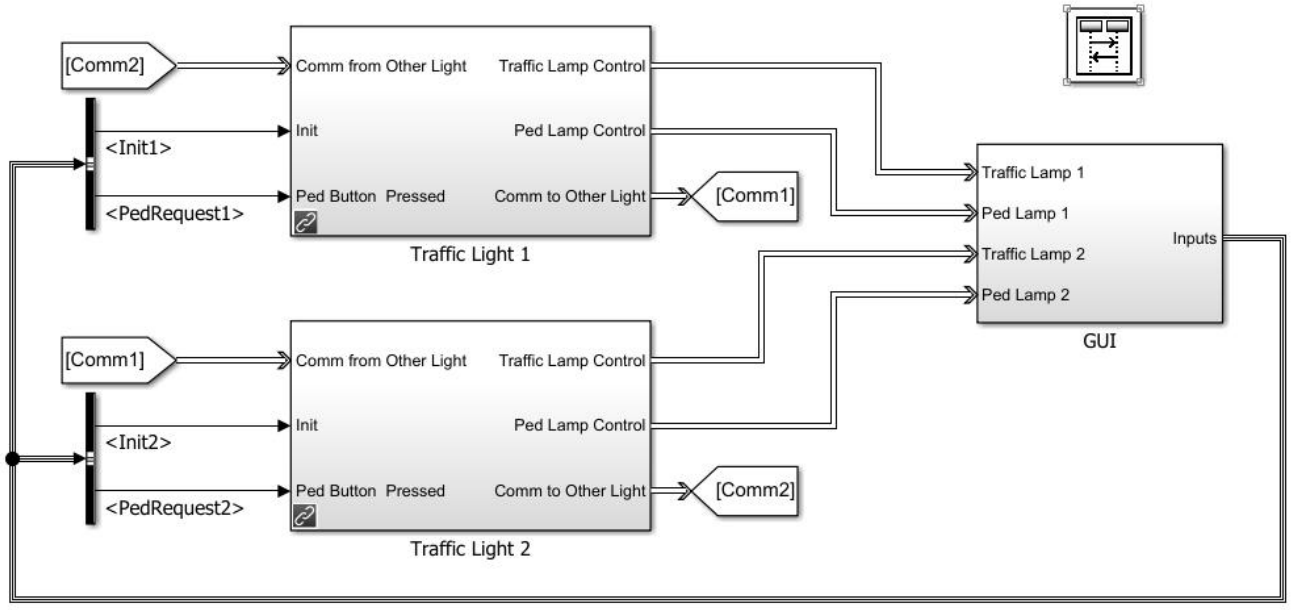
Message

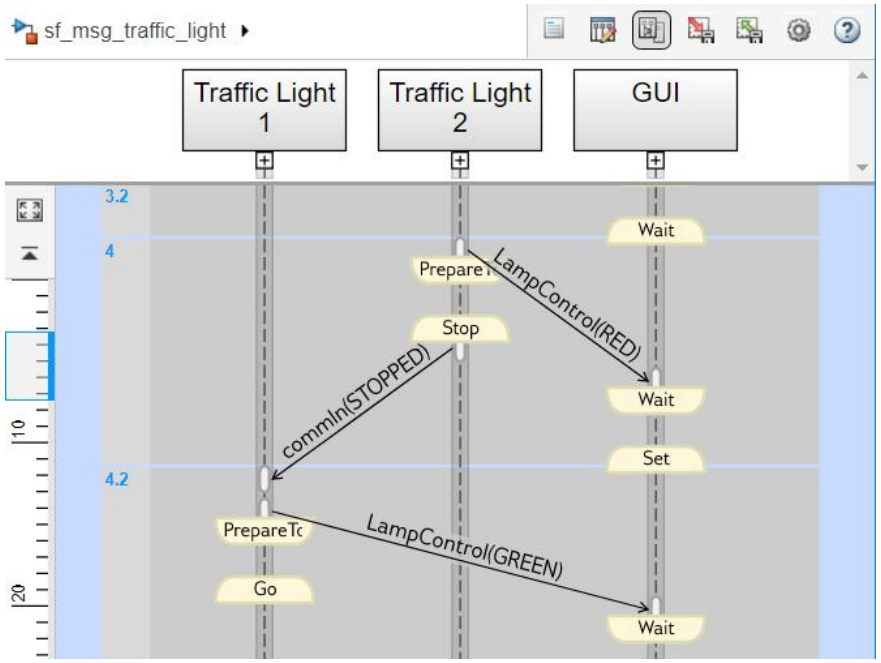
- Contains data: Message_name.data
- Receiver has a queue for each input message
- send(message_name)
- receive(message_name)
- discard(message_name)
- forward(input_message_name, output_message_name)
- isvalid(message_name)
  - if the chart has removed it from the queue and has not forwarded or discard



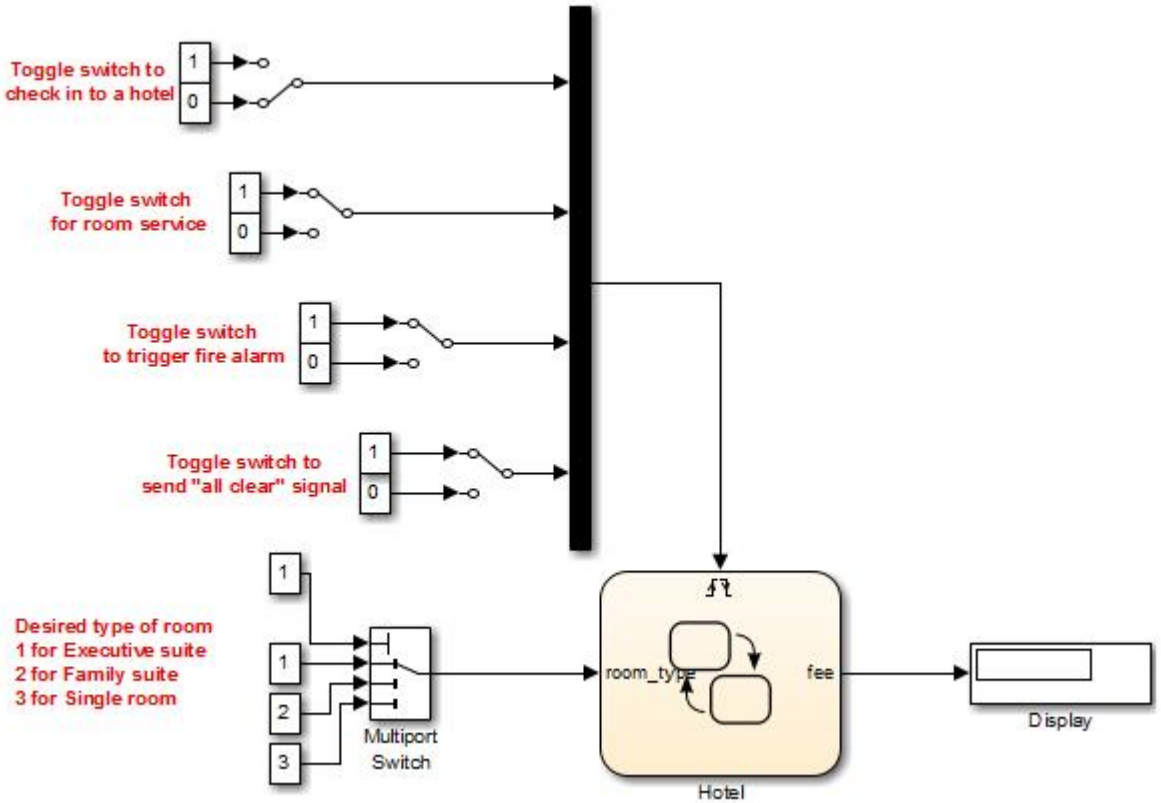


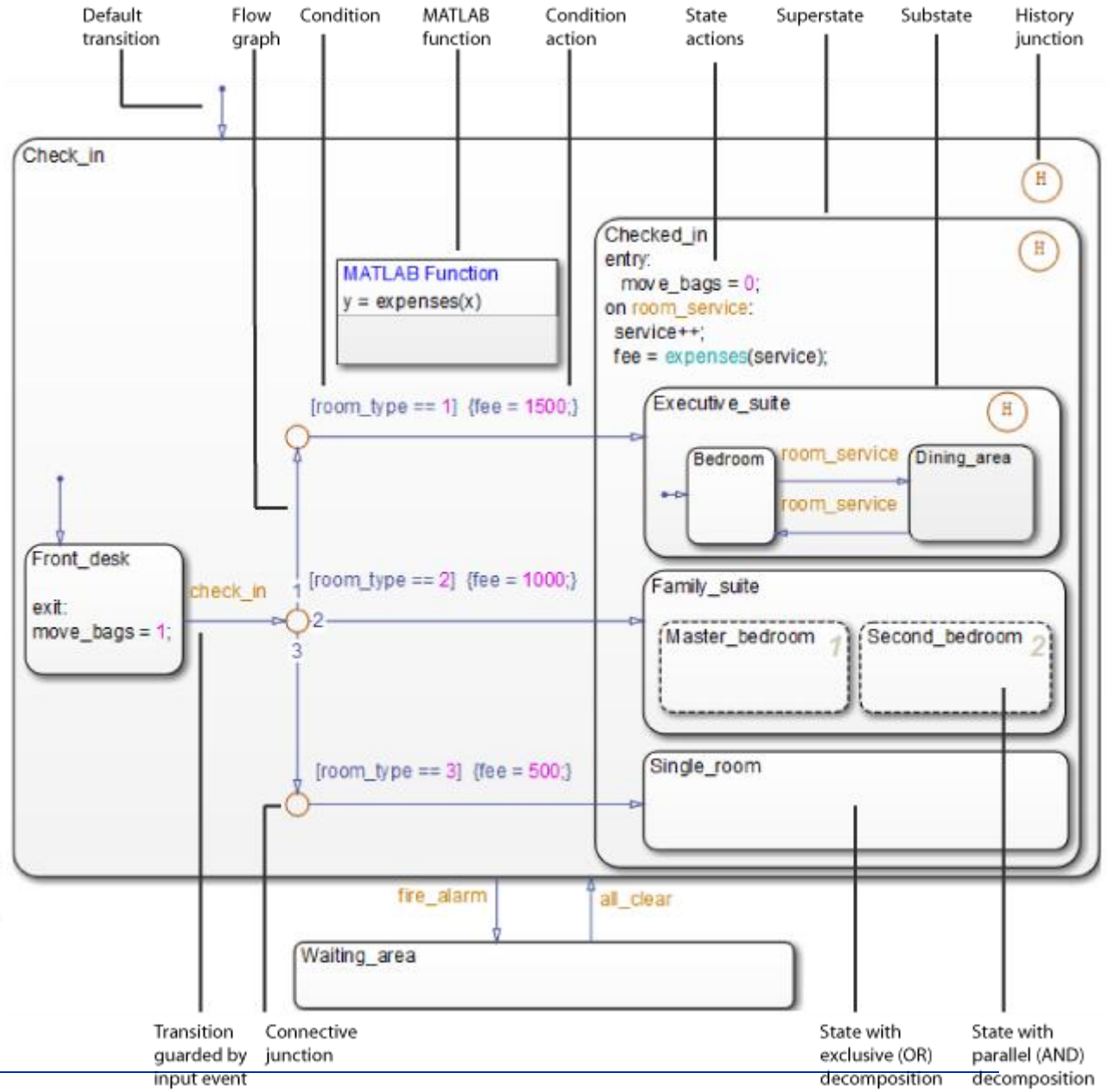
Visualizing messages/events





Example


Toggle switch to check in to a hotel
Toggle switch for room service
Toggle switch to trigger fire alarm
Toggle switch to send "all clear" signal
Desired type of room
1 for Executive suite
2 for Family suite
3 for Single room
Multiport Switch
room_type
fee
Hotel
Display


Default transition
Flow graph
Condition
MATLAB function
Condition action
State actions
Superstate
Substate
History junction
Check_in
MATLAB Function
y = expenses(x)
Checked_in
entry:
move_bags = 0;
on room_service:
service++;
fee = expenses(service);
[room_type == 1] {fee = 1500;}
Executive_suite
Bedroom
room_service
Dining_area
room_service
Front_desk
exit:
move_bags = 1;
check_in
[room_type == 2] {fee = 1000;}
Family_suite
Master_bedroom
Second_bedroom
[room_type == 3] {fee = 500;}
Single_room
fire_alarm
all_clear
Waiting_area
Transition guarded by input event
Connective junction
State with exclusive (OR) decomposition
State with parallel (AND) decomposition

Modeling Tips

- Use signals of the same data type for input events
- Use a default transition to mark the first state to become active among exclusive (OR) states
- Use condition actions instead of transition actions whenever possible
- Use explicit ordering to control the testing order of a group of outgoing transitions
- Use MATLAB functions for performing numerical computations in a chart

Discussion: First Consultation

- UML
- What to discuss?
- Interactions during consultation