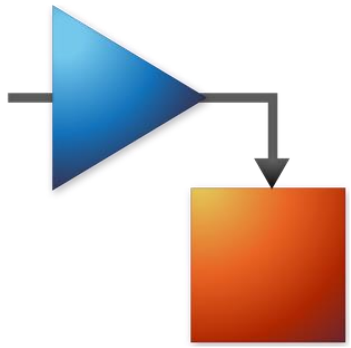
Lecture 17: Simulink & Stateflow

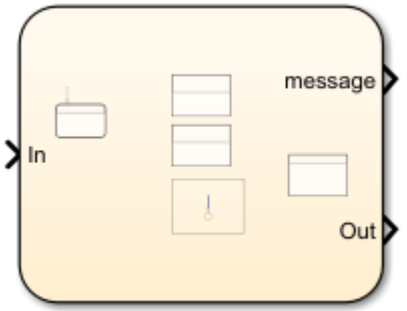


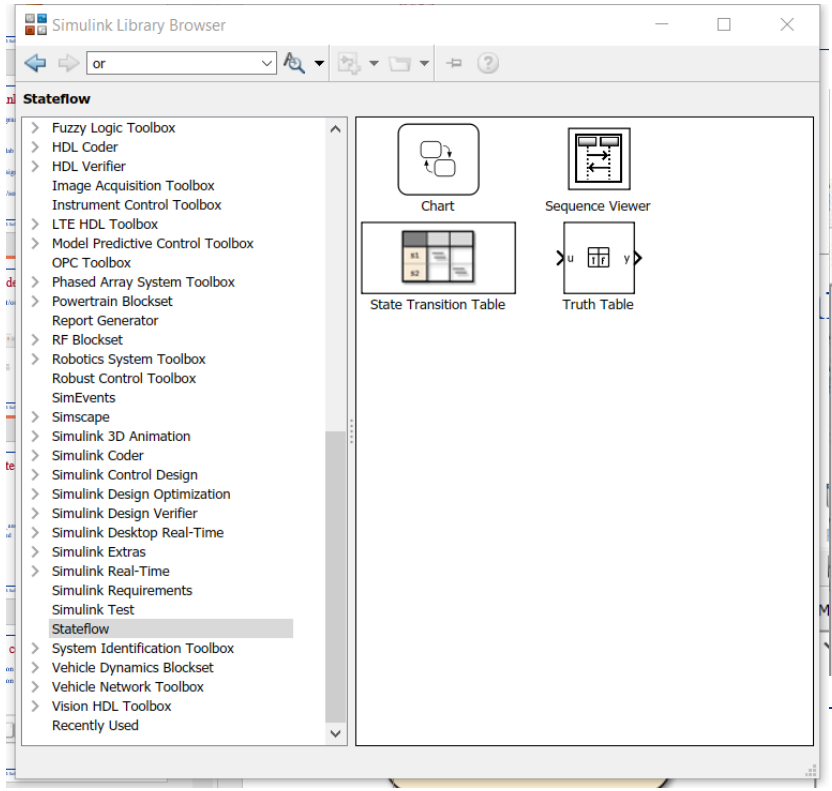
Simulink & Stateflow

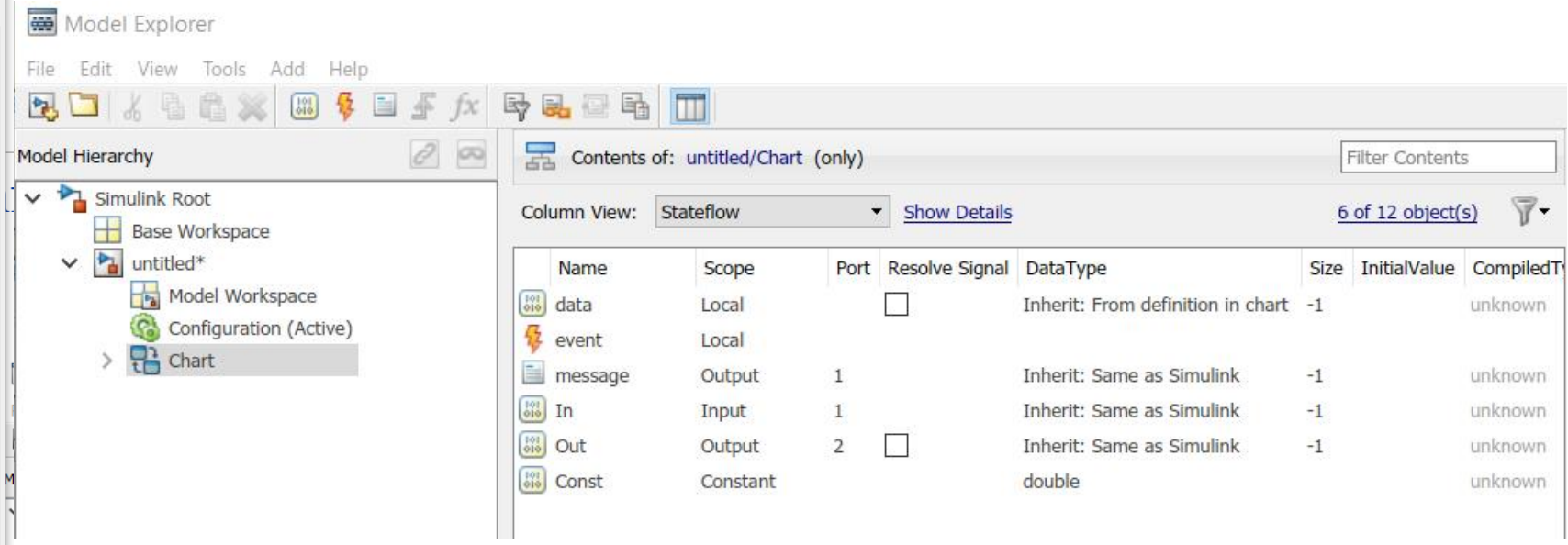
- A graphical modeling/programming language
- Developed by Mathworks
  - Highly integrated with Matlab
- Has a full model-based design toolchain
- Widely adapted by system/software developers
- Rich expressiveness

Model Explorer

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

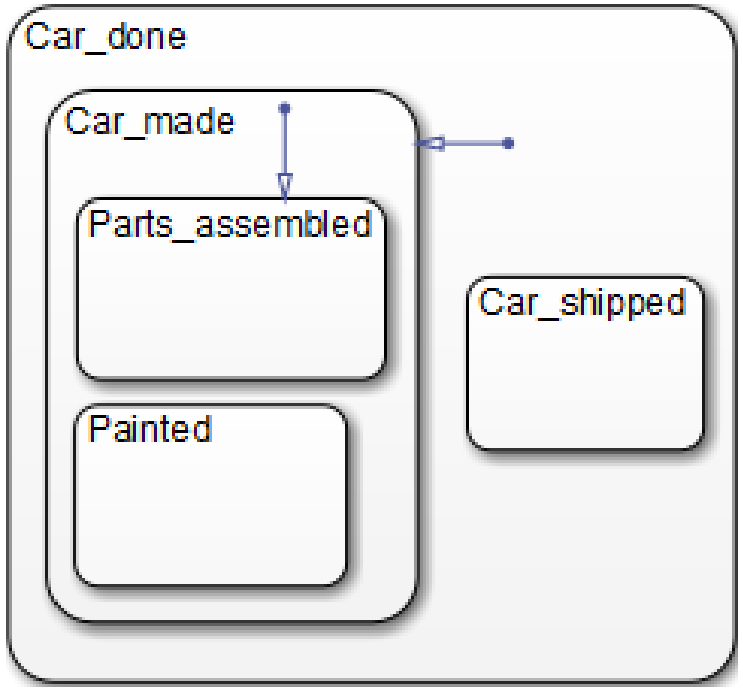






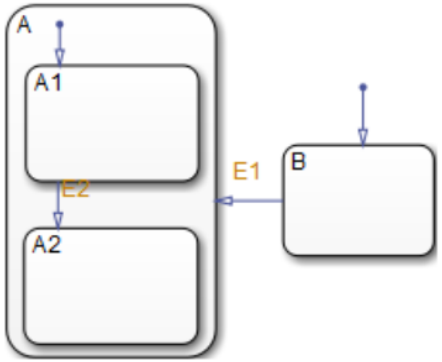
State Hierarchy

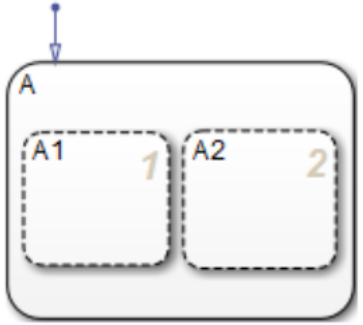
- Object oriented modeling
  - /Car_done
  - /Car_done.Car_made
  - /Car_done.Car_shipped
  - /Car_done.Car_made.Parts_assembled
  - /Car_done.Car_made.Painted

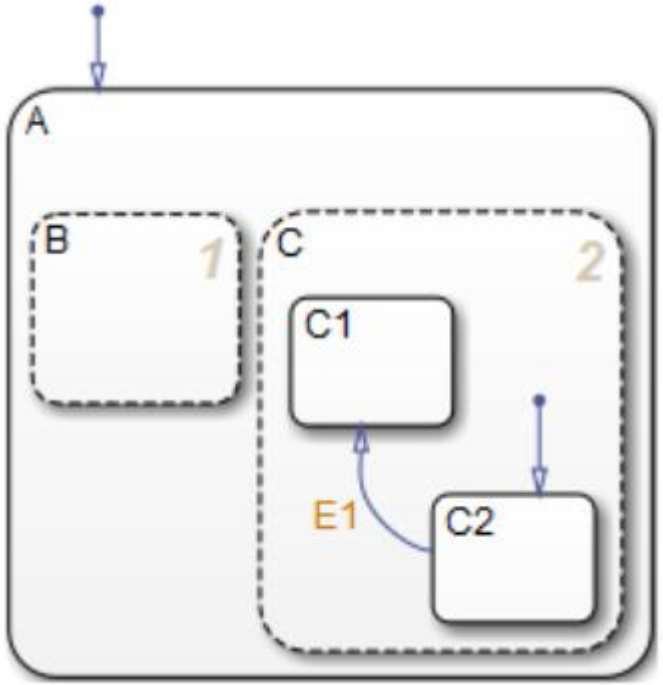


State compositions

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

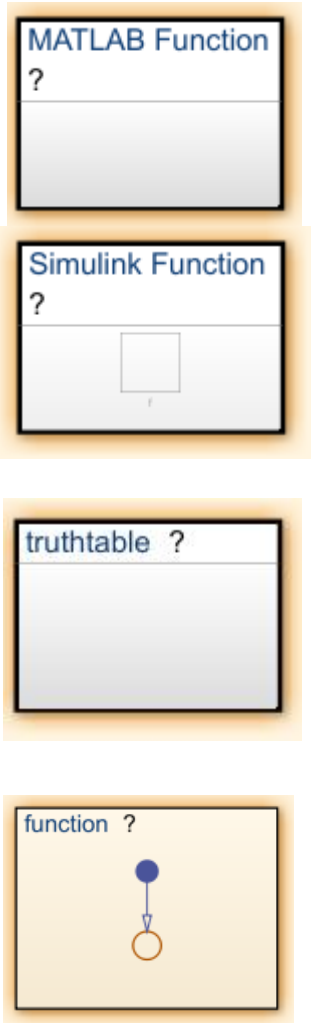






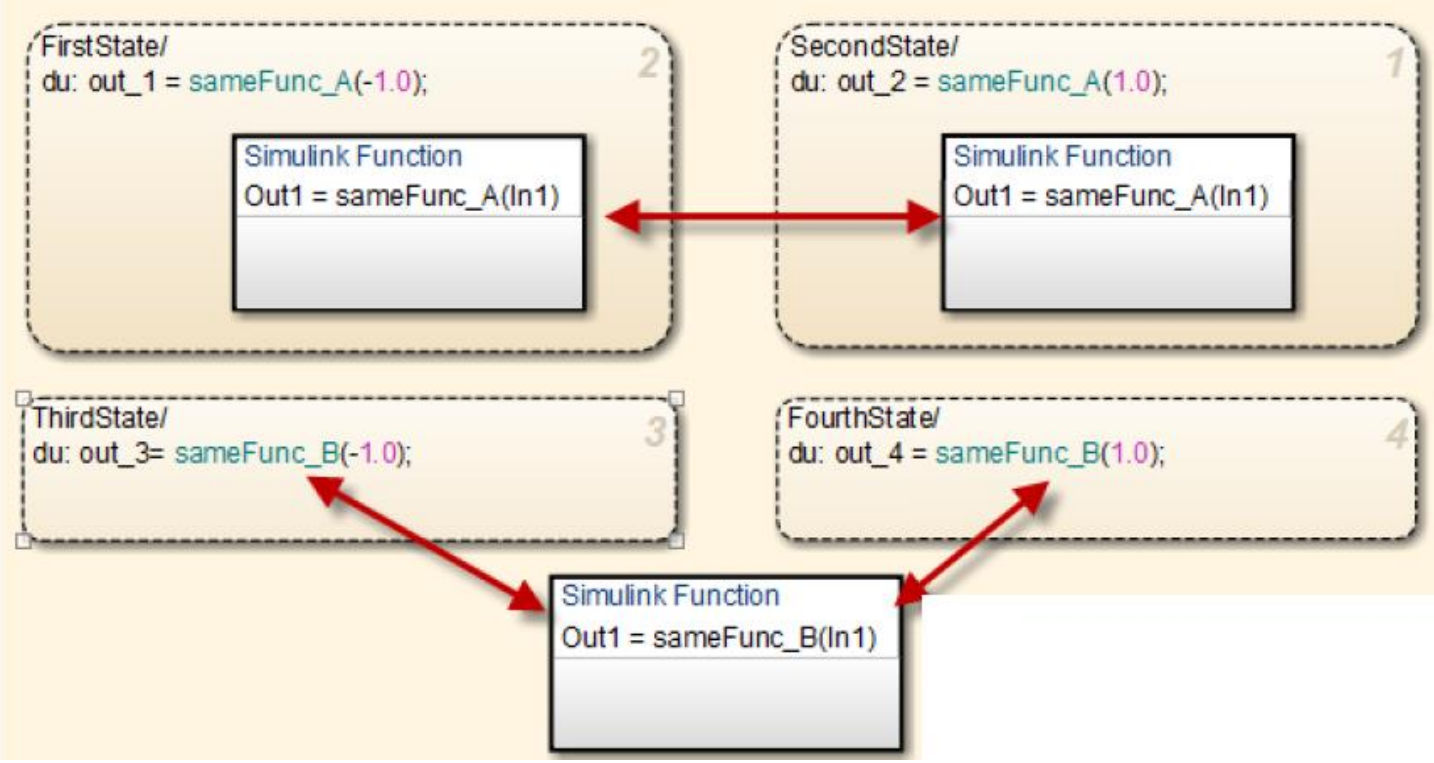
Embedded Functions

- Matlab function
- Simulink function
- Truthtable
- Graphical function



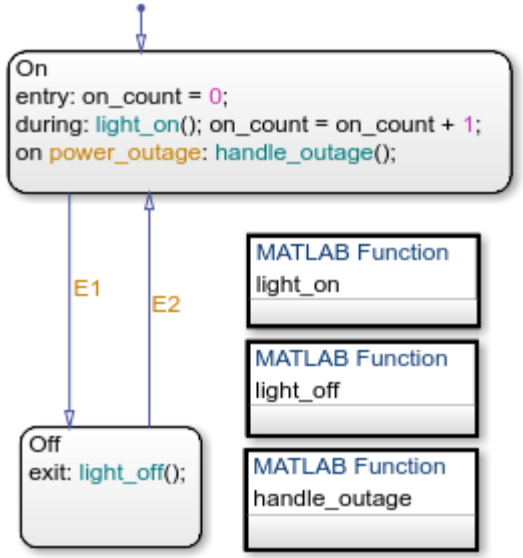
Function availability

- 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();
```

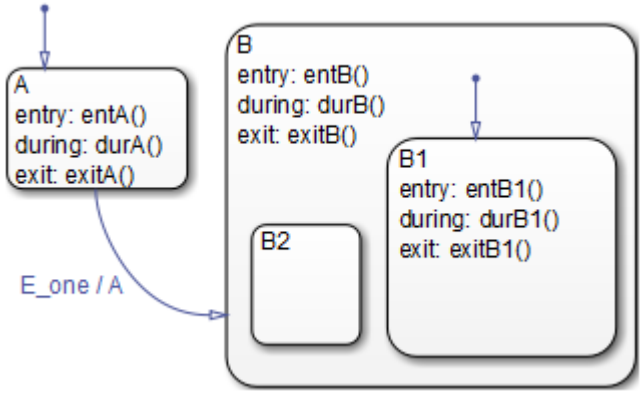
```
en, du, ex: fcn1();
en: fcn2();
```

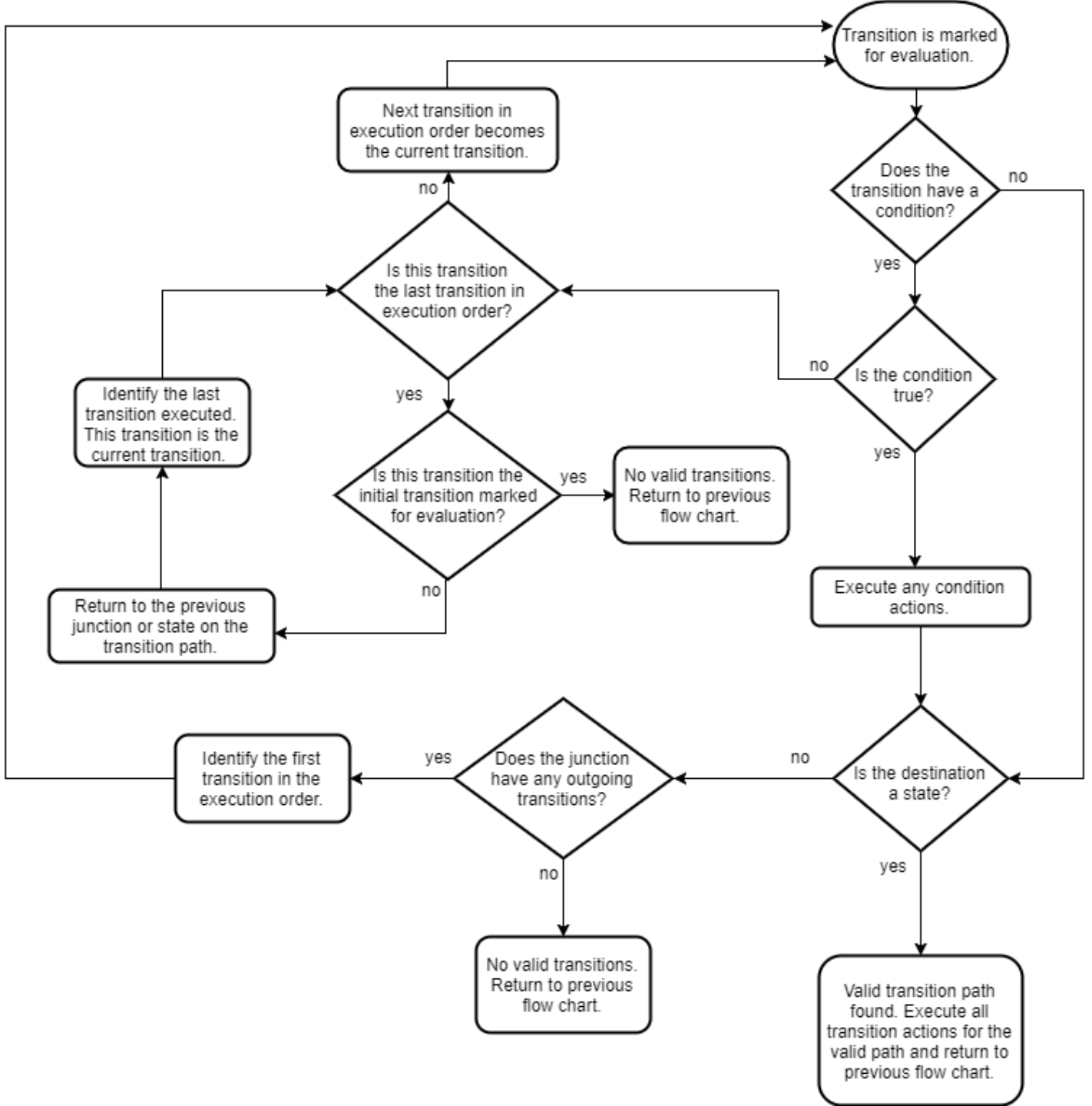
Transition

- 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

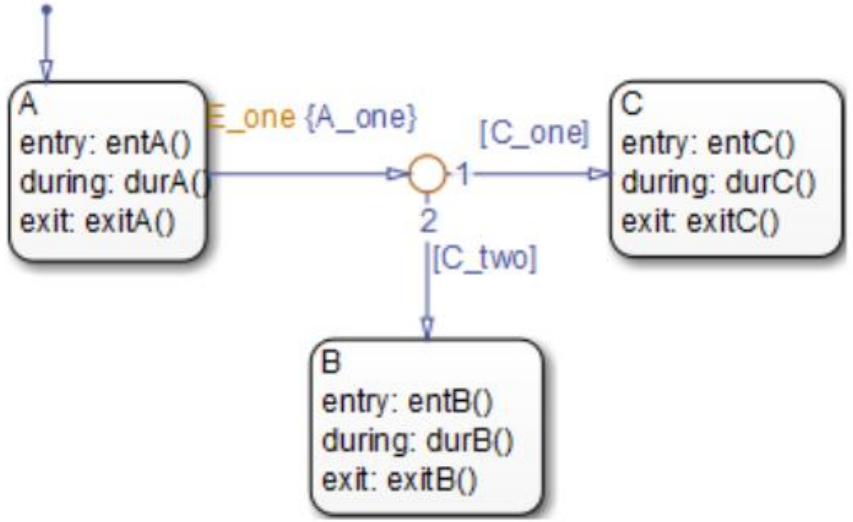
- State A is active. Event E_one occurs and awakens the chart
- State A exit actions (exitA()) execute and complete.
- State A is marked inactive.
- The transition action, A, is executed and completed.
- State B is marked active.
- State B entry actions (entB()) execute and complete.
- State B detects 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
no
Identify the first transition in the execution order.
No valid transitions. Return to previous flow chart.
Next transition in execution order becomes the current transition.
no
Is this transition the last transition in execution order?
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.

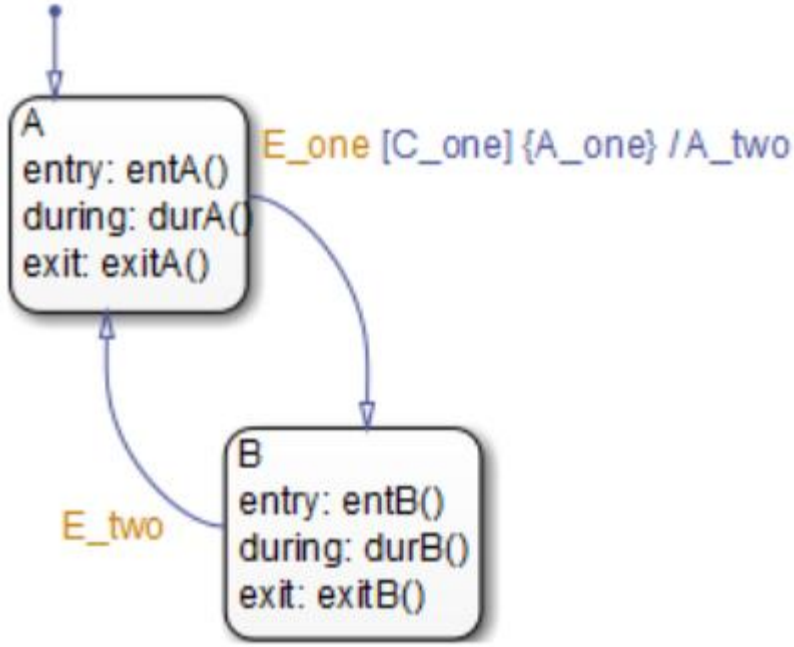
Condition Action Behavior

- 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

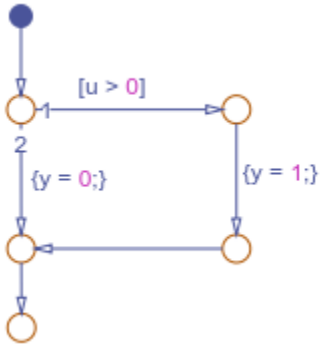
- E_one happened and awaked the chart.
- The condition C_one is true. The condition action A_one is immediately executed.
- State A is still active.
- State A exit actions (ExitA()) execute and complete.
- State A is marked inactive.
- The transition action A_two is executed.
- State B is marked active.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



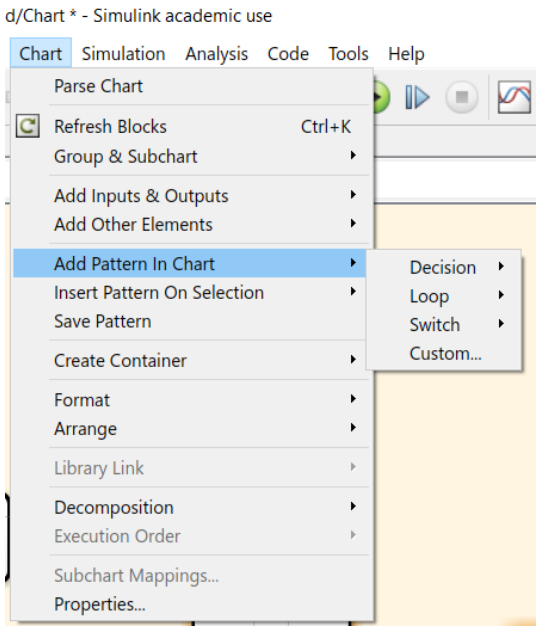
Flow chart

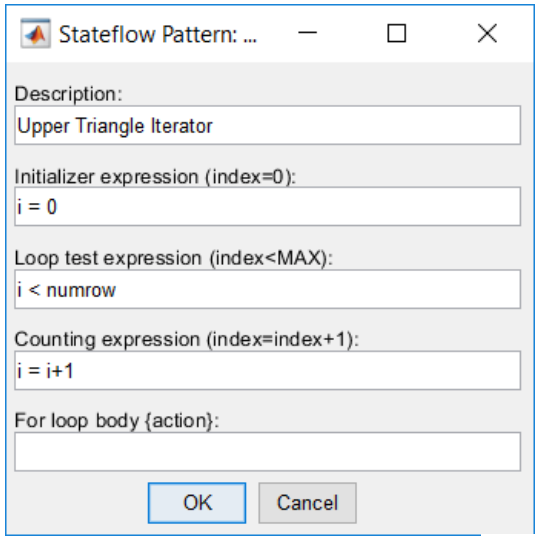
- No time consumption during execution
- Can be used for graphical function definition

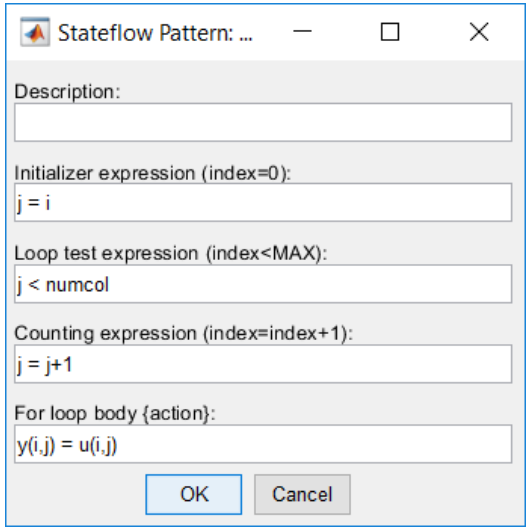
```
if u > 0
   y = 1;
else
   y = 0;
end
```

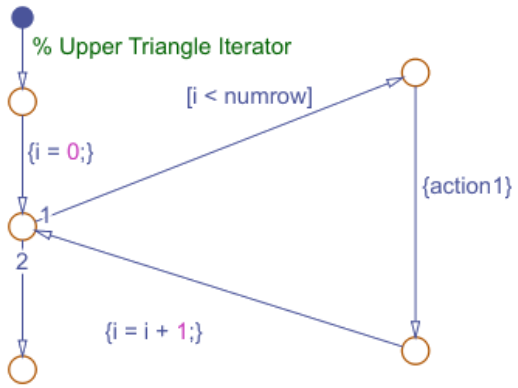


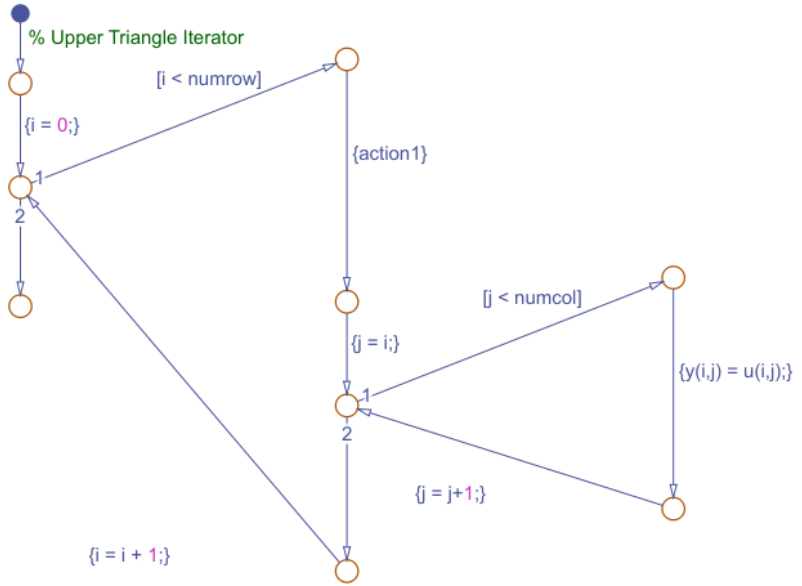
Add pattern in chart







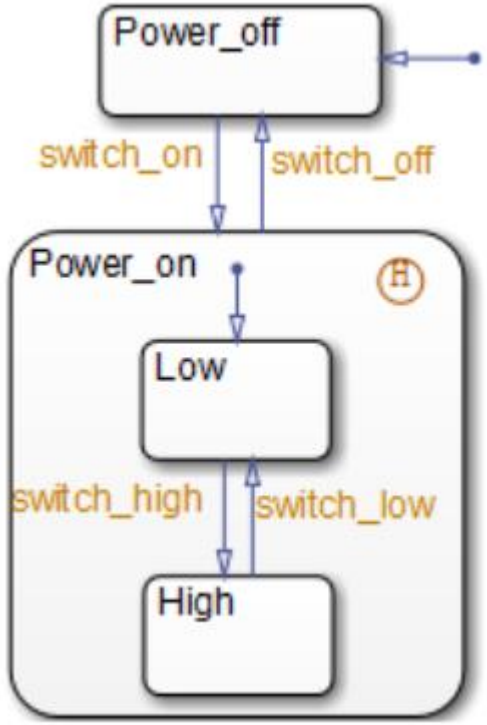




History Junction

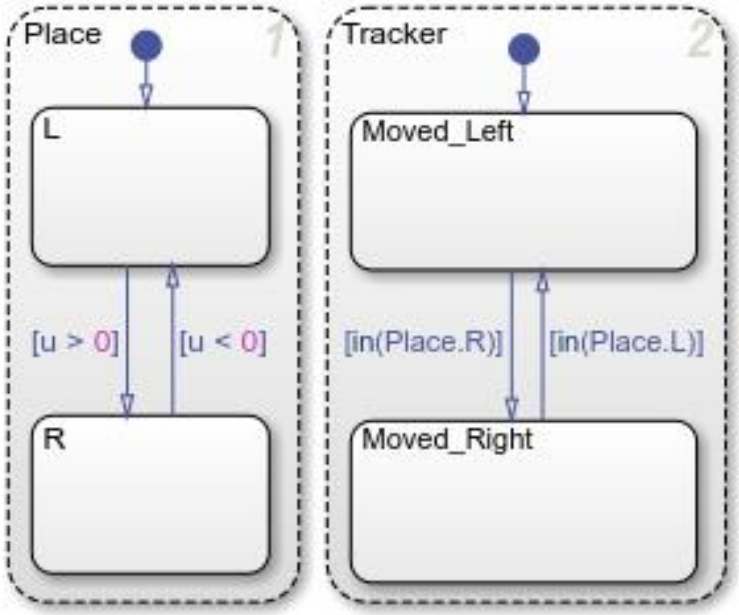
- Restores the state that is on the same level of the composite state as the history state itself

- 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



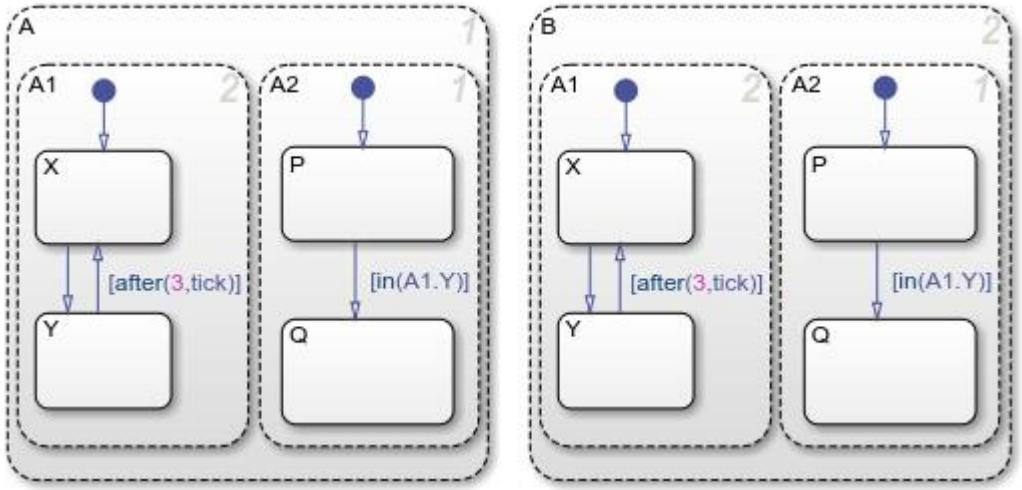
Check State Activity by Using in() Operator

- 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



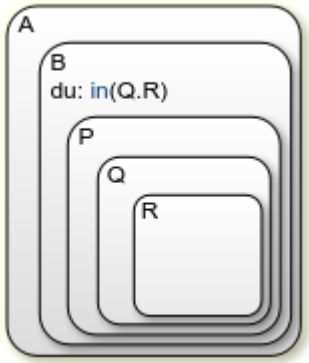
In() operator example

- 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

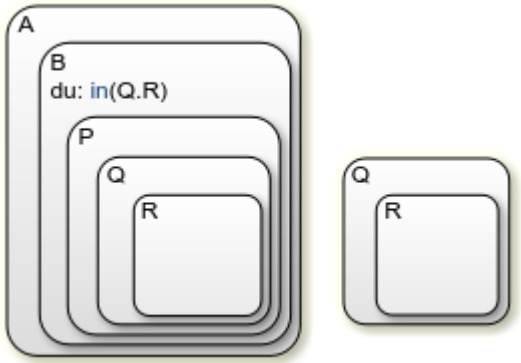


In() operator example (cont.)

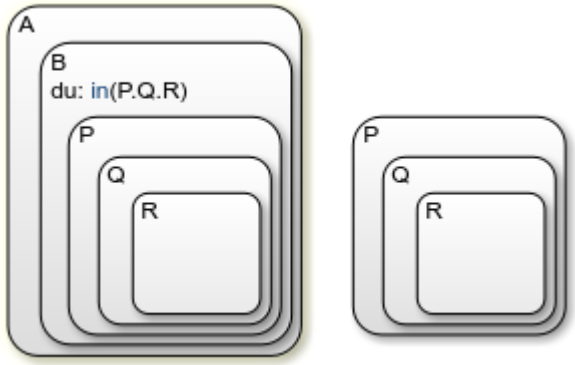
No match



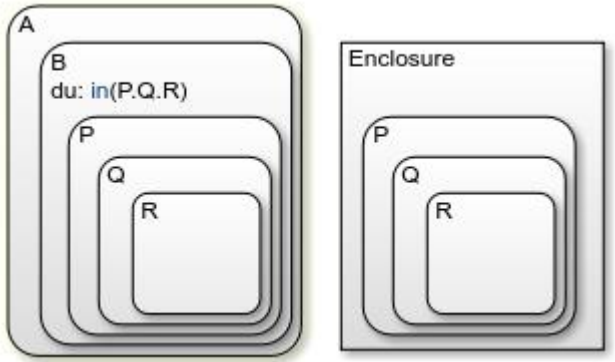
In(P.Q.R) will do
Wrong match



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

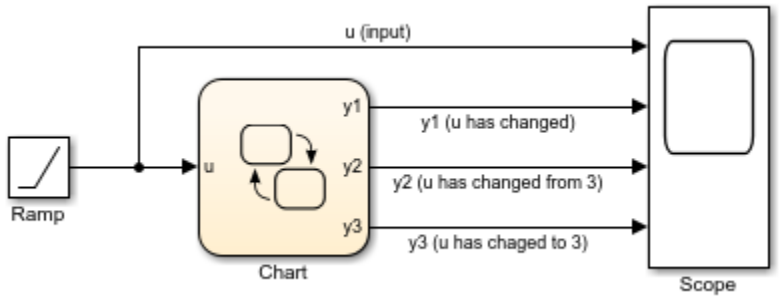


Use enclosure to ensure local match

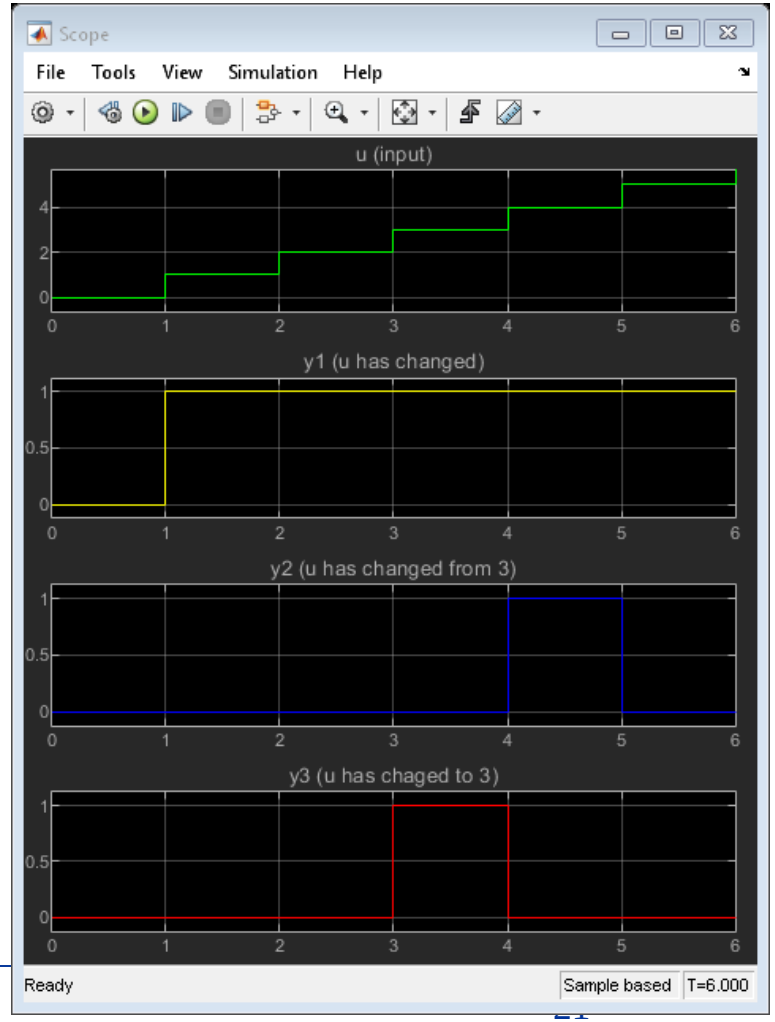


Detect data change

- 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.

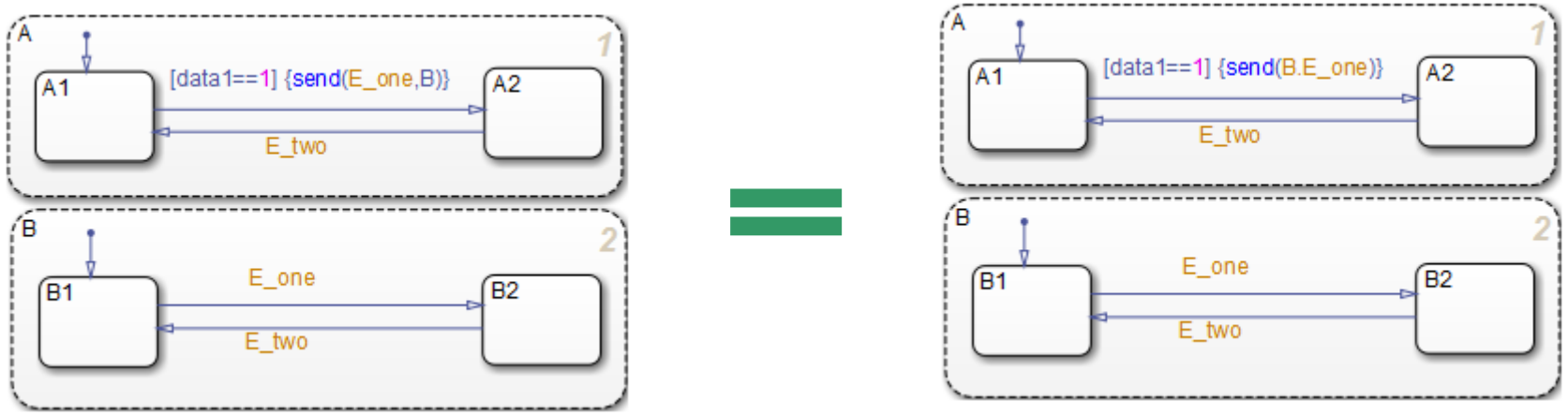



```
{
y1 = hasChanged(u);
y2 = hasChangedFrom(u,3);
y3 = hasChangedTo(u,3);
}
```



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.

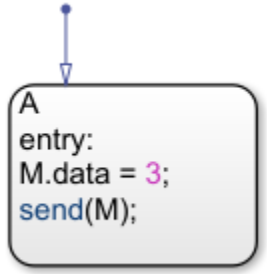


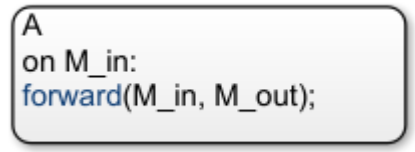
Implicit Events

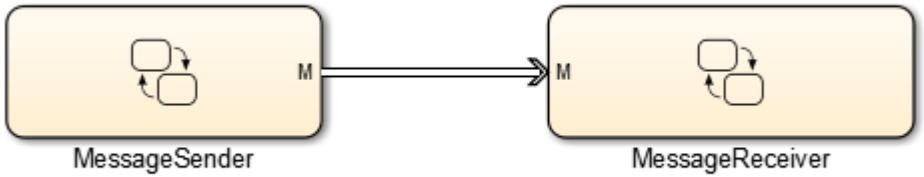
- change(data_name) or chg(data_name)
  - generates a local event when writing a value to the variable data_name
  - Data_name has to be at chart level or lower
- enter(state_name) or en(state_name)
  - generates a local event when the specified state_name is entered
- exit(state_name) or ex(state_name)
  - 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

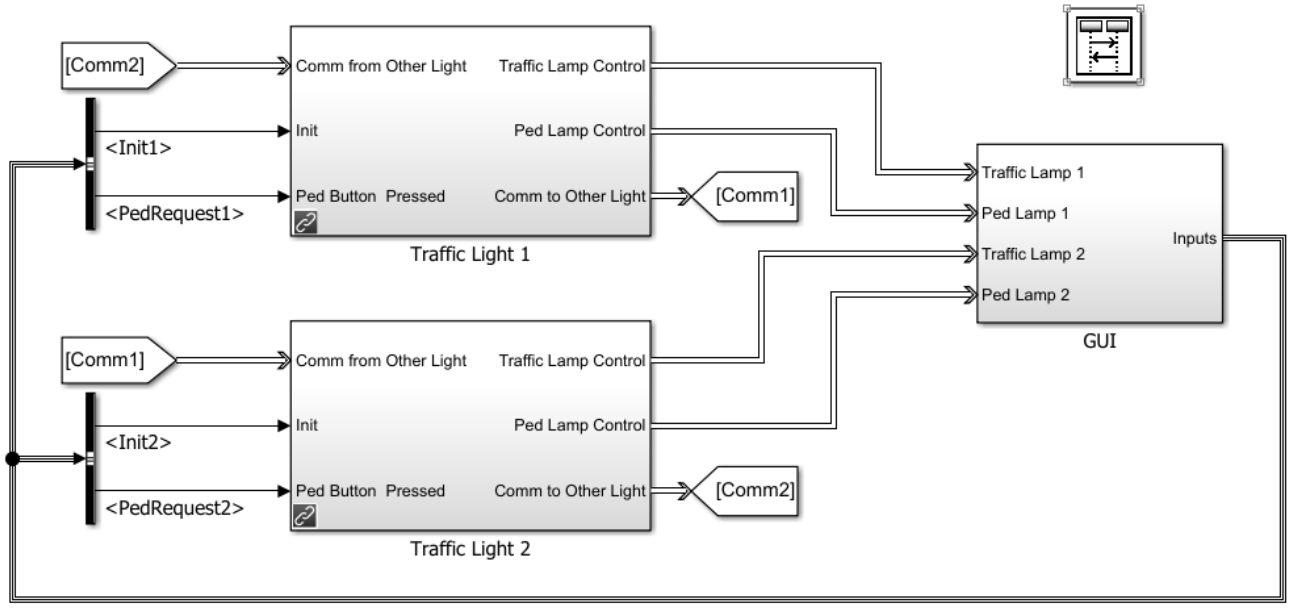
- Contains data: Message_name.data
- Receiver has a queue for each input message
- send(message_name)
- receive(message_name)
- discard(message_name)
- forward(input_message_name, output_message_name)
- isvalid(message_name)
  - if the chart has removed it from the queue and has not forwarded or discarde

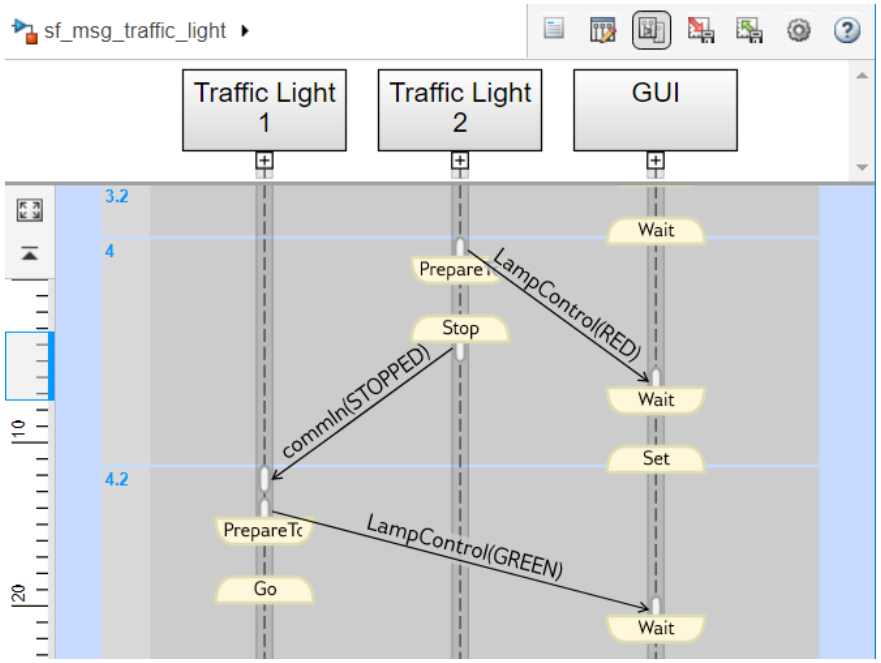




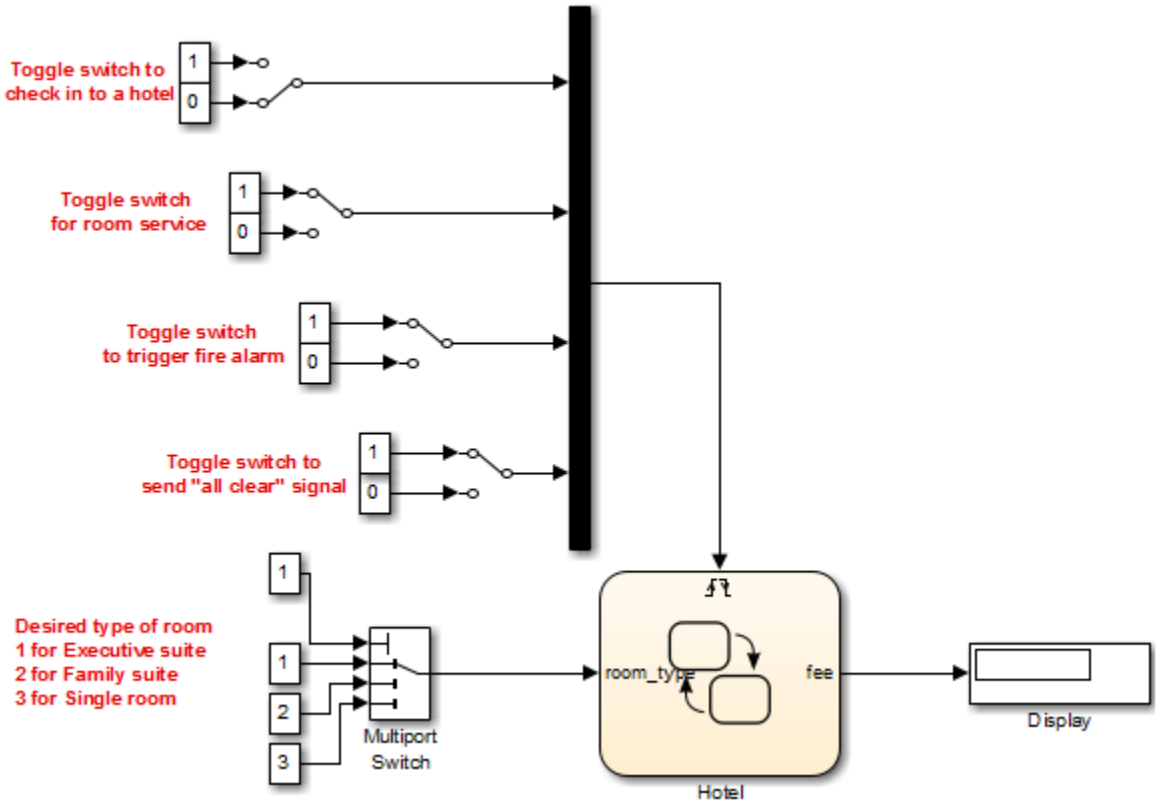


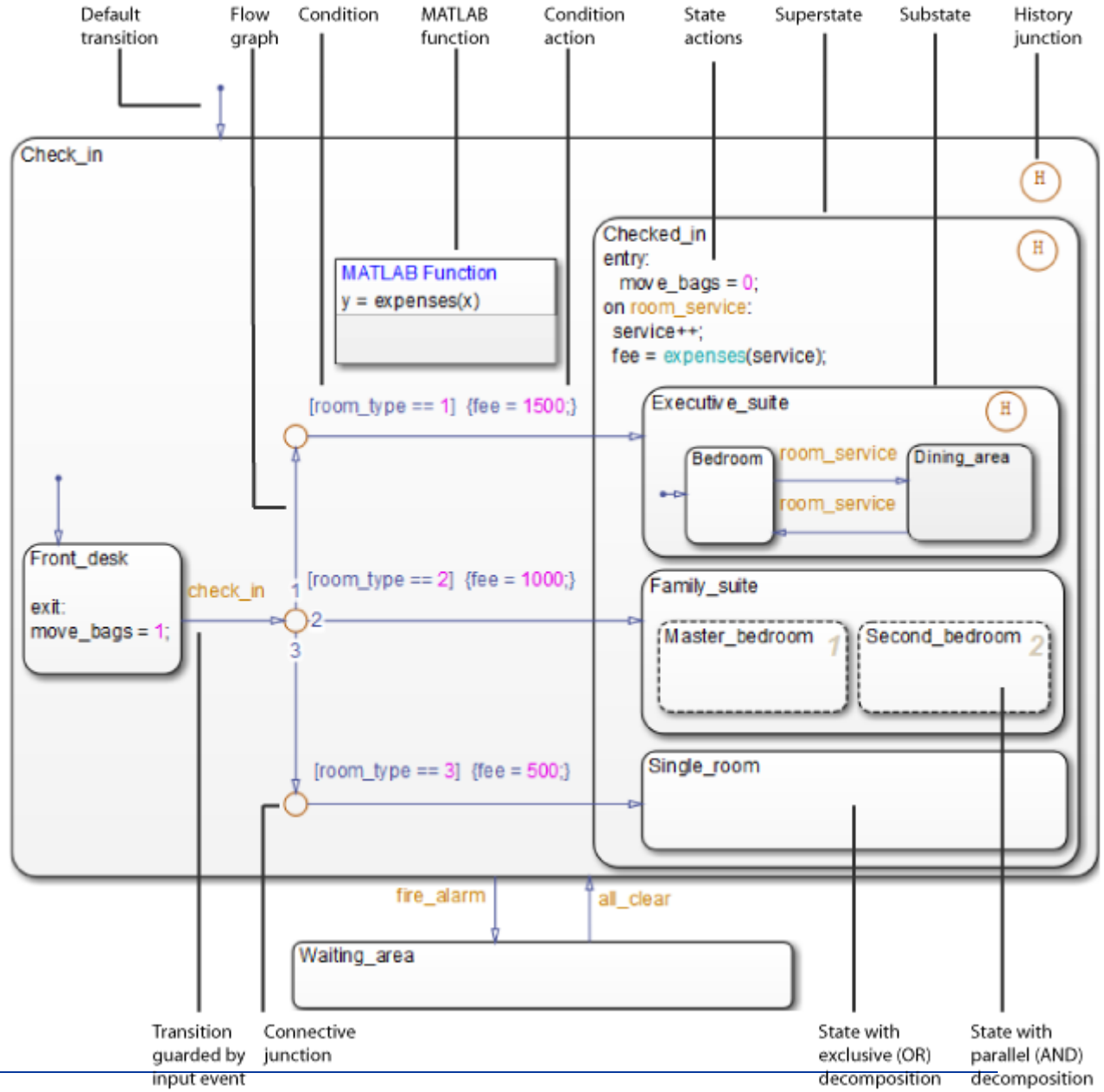
Visualizing messages/events





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

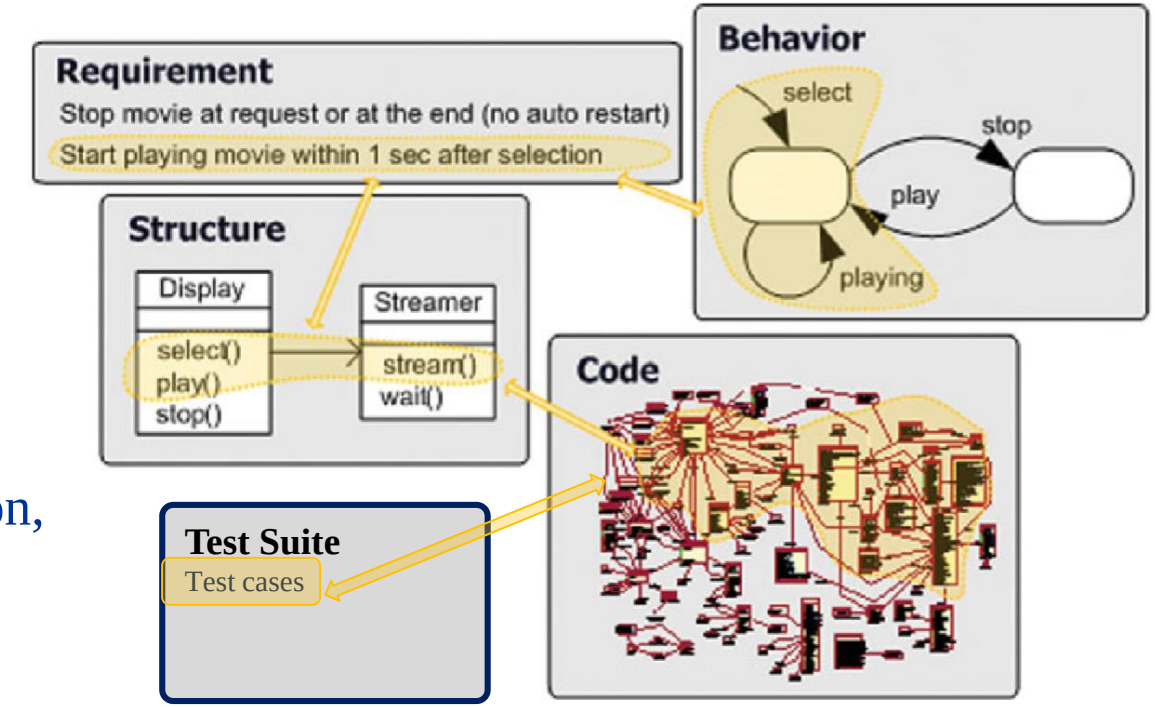
Modeling Tips

- 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

Traceability

What is traceability?

- We would like to make sure that
  - All requirements are implemented
  - All implementations are necessary
- Trace artifacts
  - Requirements, models, code, etc.
- Trace link
  - Association between two trace artifacts
  - Type: Refinement, Abstraction, Implementation, etc.
- Trace granularity: component level, statement level, etc.
- Trace quality: completeness, correctness, etc.

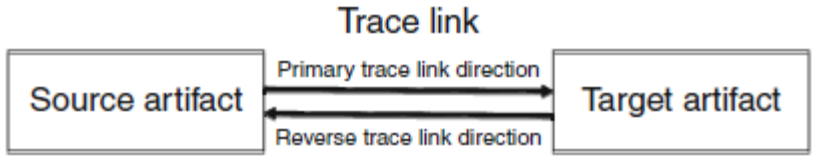


Objectives of Traceability

- Software lifecycle involves more than one person
- Within the team
  - Make sure the requirements are faithfully translated to code
- For the customers and regulation agencies
  - Part of validation evidence

Traceability Activities

- Trace Creation
  - Establish *trace link* between a *source artifact* and a *target artifact*
  - Traceability document



- Trace Validation
  - Between requirements and model: Model checking
  - Between concept model and implementation model: Model translation
  - Between model and code: Conformance testing
- Trace Maintenance
  - Update trace when modification happened

Class-based Testing in Matlab

- testCase.verifyEqual

```
%% Test Class Definition
classdef MyComponentTest < matlab.unittest.TestCase

    %% Test Method Block
    methods (Test)

        %% Test Function
        function testASolution(testCase)
            %% Exercise function under test
            % act = the value from the function under test

            %% Verify using test qualification
            % exp = your expected value
            % testCase.<qualification method>(act,exp);
        end
    end
end
```

Testing APP

```
classdef TestPatientsDisplay < matlab.uitest.TestCase
    properties
        App
    end

    methods (TestMethodSetup)
        function launchApp(testCase)
            testCase.App = PatientsDisplay;
            testCase.addTeardown(@delete,testCase.App);
        end
    end

    methods (Test)
        function test_plottingOptions(testCase)
            % Press the histogram radio button
            testCase.press(testCase.App.HistogramButton)

            % Verify xlabel updated from 'Weight' to 'Systolic'
            testCase.verifyEqual(testCase.App.UIAxes.XLabel.String,'Systolic')

            % Change the Bin Width to 9
            testCase.choose(testCase.App.BinWidthSlider,9)

            % Verify the number of bins is now 4
            testCase.verifyEqual(testCase.App.UIAxes.Children.NumBins,4)
        end

        function test_tab(testCase) ...

    end
end
```

Component	`matlab.uitest.TestCase` Gesture Method				
	`press`	`choose`	`drag`	`type`	`hover`
Button	✓				
State button	✓	✓			
Check box	✓	✓			
Switch	✓	✓			
Discrete knob		✓			
Knob		✓	✓		
Drop-down		✓		✓	
Edit field				✓	
Text area				✓	
Spinner	✓			✓	
Slider		✓	✓		
List box		✓			
Button group		✓			
Tab group		✓			
Tab		✓			
Tree node		✓			
Menu	✓				
Date Picker				✓	
Axes	✓				✓
UI Axes	✓				✓
UI Figure	✓				✓