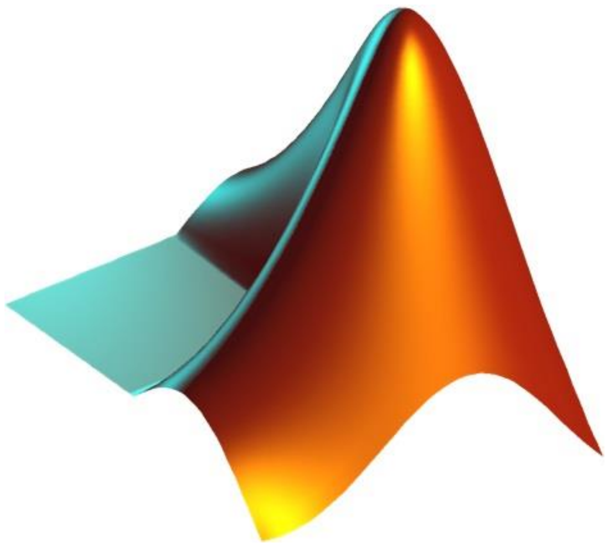
MATLAB / Simulink Lab Course
Stateflow

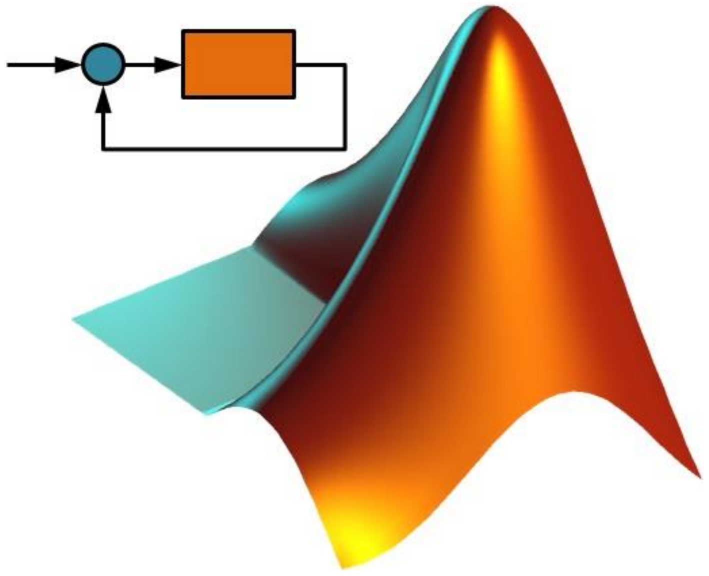


Objectives & Preparation “Stateflow"

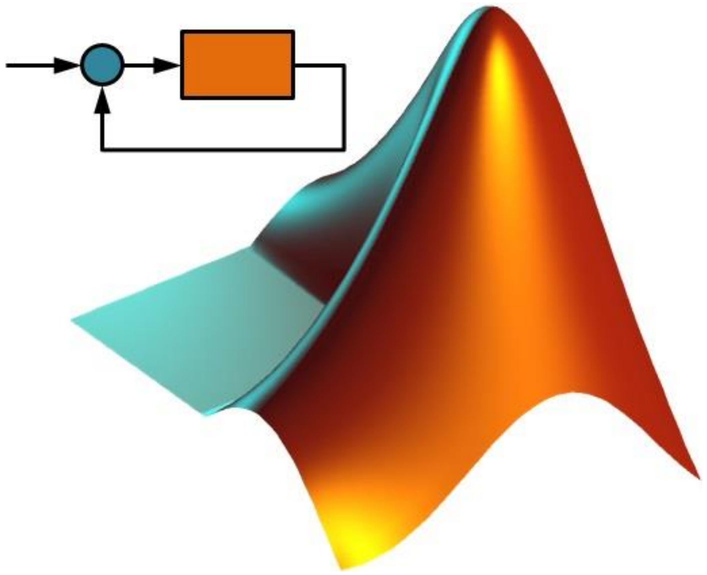
- Which MathWorks products are covered?
  - ⇨ Stateflow Toolbox
- What skills are learnt?
  - ⇨ Stateflow Elements
  - ⇨ Transition Actions and Transition Conditions
  - ⇨ Stateflow Action Languages
  - ⇨ Data Management in Stateflow
- How to prepare for the session?
  - ⇨ Mathworks Documentation
    - ⇨ https://www.mathworks.com/help/stateflow/ug/finite-state-machine-concepts
    - ⇨ https://www.mathworks.com/help/stateflow/ug/states.html
    - ⇨ https://www.mathworks.com/help/stateflow/ug/transitions.html
  - ⇨ MathWorks Videos
    - ⇨ https://www.mathworks.com/videos/stateflow-overview-61210.html
    - ⇨ https://www.mathworks.com/videos/getting-started-with-stateflow-70063.htm

Outline

1. Introduction
2. Elements in Stateflow
3. Data Management
4. Action Language
5. Structure and Hierarchy
6. Coding Guidelines / Lessons Learned



1. Introduction

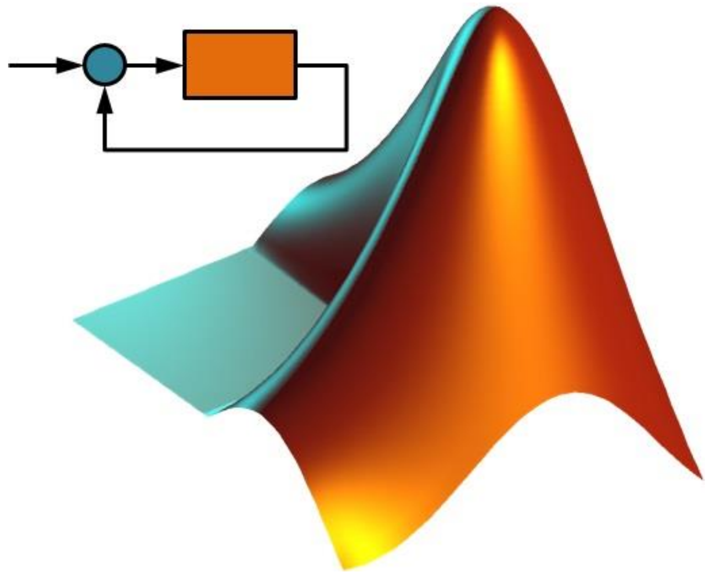


Introduction

- What is Stateflow?
  - Environment for modeling and simulating decision logic
    - State charts
    - Flow charts
  - Logic for
    - Supervisory control and monitoring
    - Task scheduling
    - Fault management applications

- Examples
  - Coffee machine
  - Beverage Vending Machine
  - ATMs
  - Traffic Lights
  - …

2. Elements in Stateflow



Elements in Stateflow

- Where to find Stateflow Elements:
  - Simulink Library Browser – Stateflow
- Chart
- Message Viewer*
  - interchange of messages between charts
- State Transition Table*
  - modal logic in tabular format
- Truth Table*
  - logical decision-making behavior with conditions, decisions, and actions

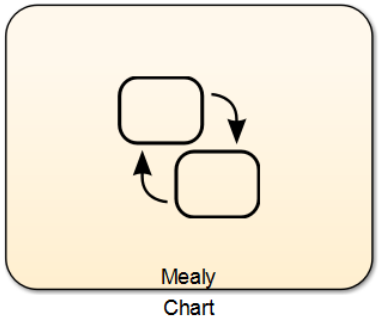


Chart Elements

- State
- Junction
- Default Transition
- Box
- Simulink State*
- Simulink Function*
- Graphical Function
- MATLAB Function*
- Truth Table*
- History*
- Annotation*
- Image*

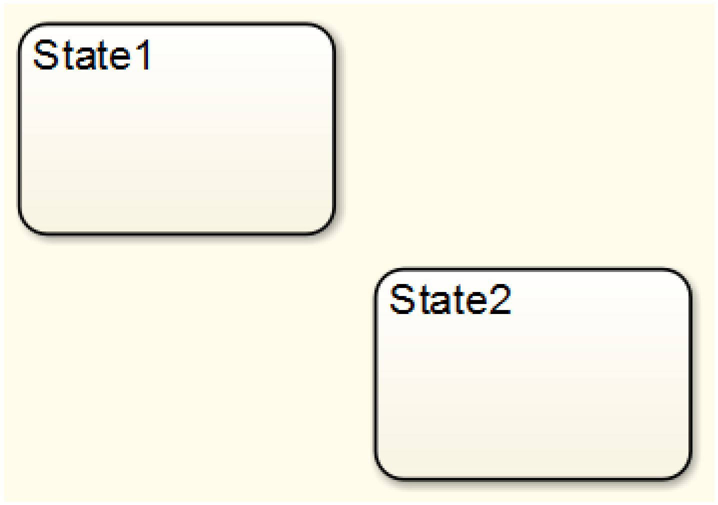
* will not be covered

Elements in Stateflow

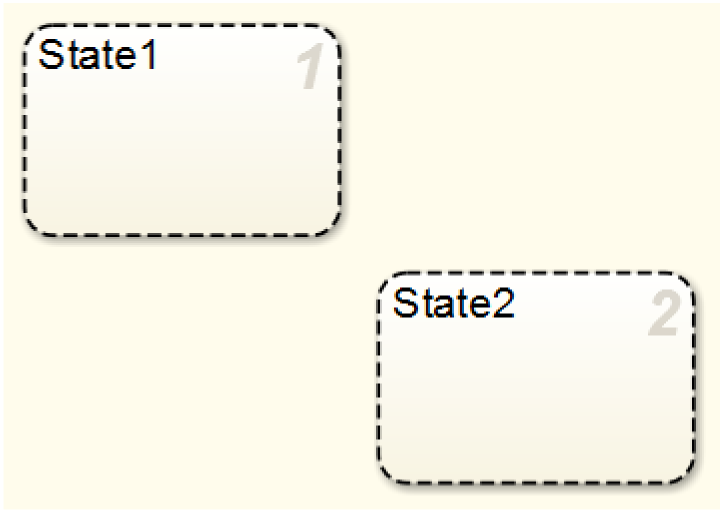
States

- Exclusive States: Only one state can be active
- Parallel States: Two or more states are active simultaneously

Exclusive



Parallel

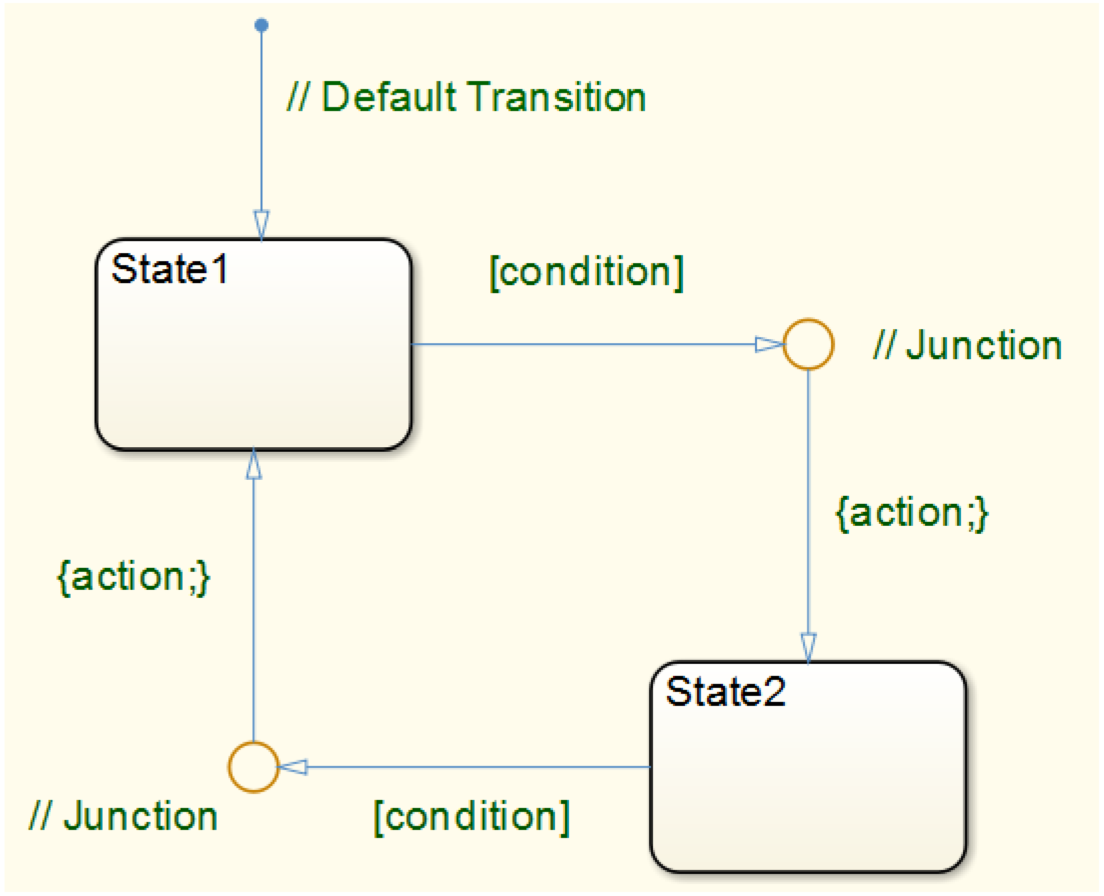


- Right Click on Parent Chart/State – Decomposition – OR (Exclusive) / AND (Parallel)

Elements in Stateflow

Default Transition / Transition / Junction

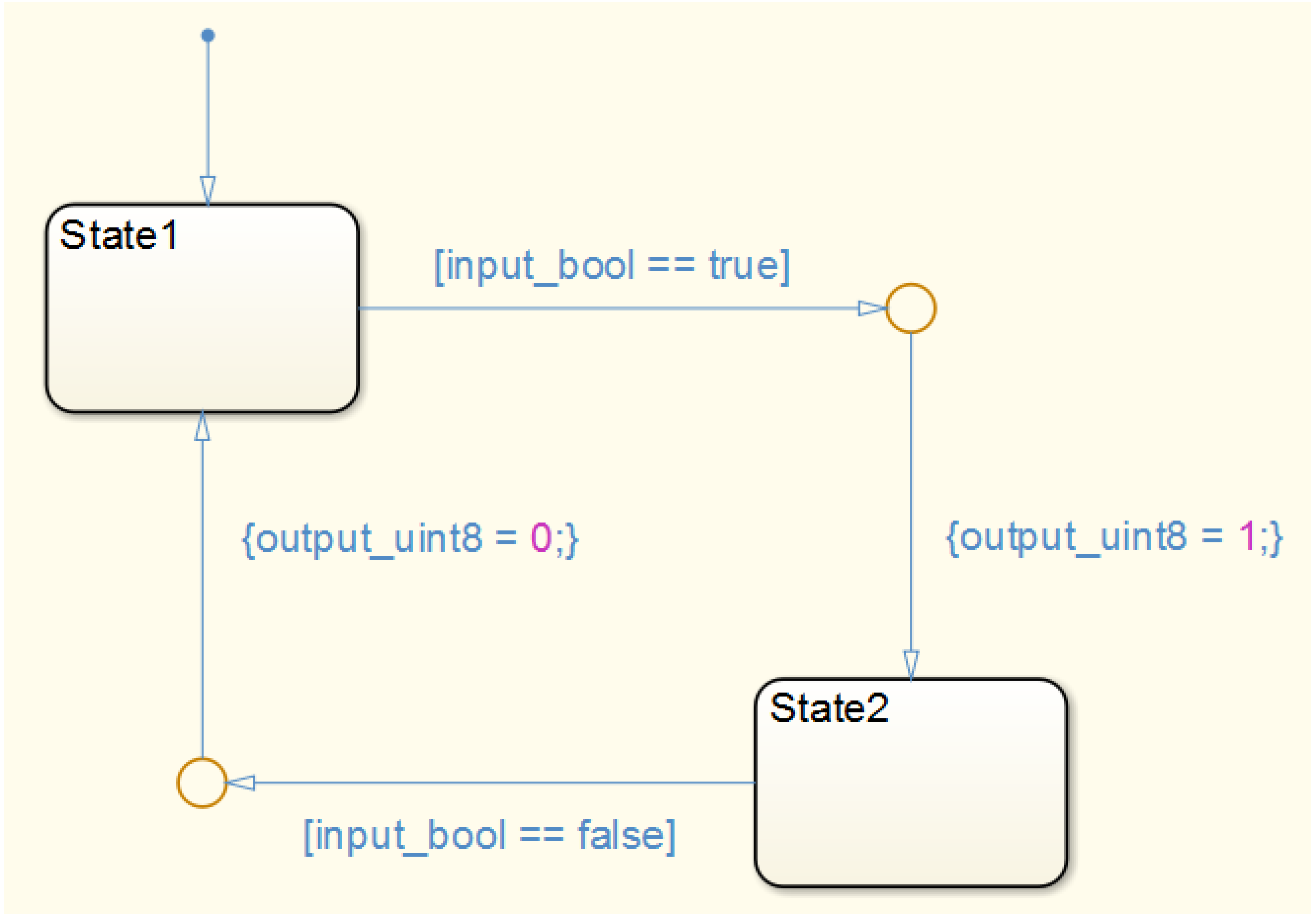
- Default Transition:
  - The Transition which is used at the activation of the chart
- Transition
  - A connection between one state and another
  - Condition
  - Action
- Junction
  - A way to combine conditions / actions
  - Graphical Reasons (horizontal / vertical) lines



Elements in Stateflow

Transitions

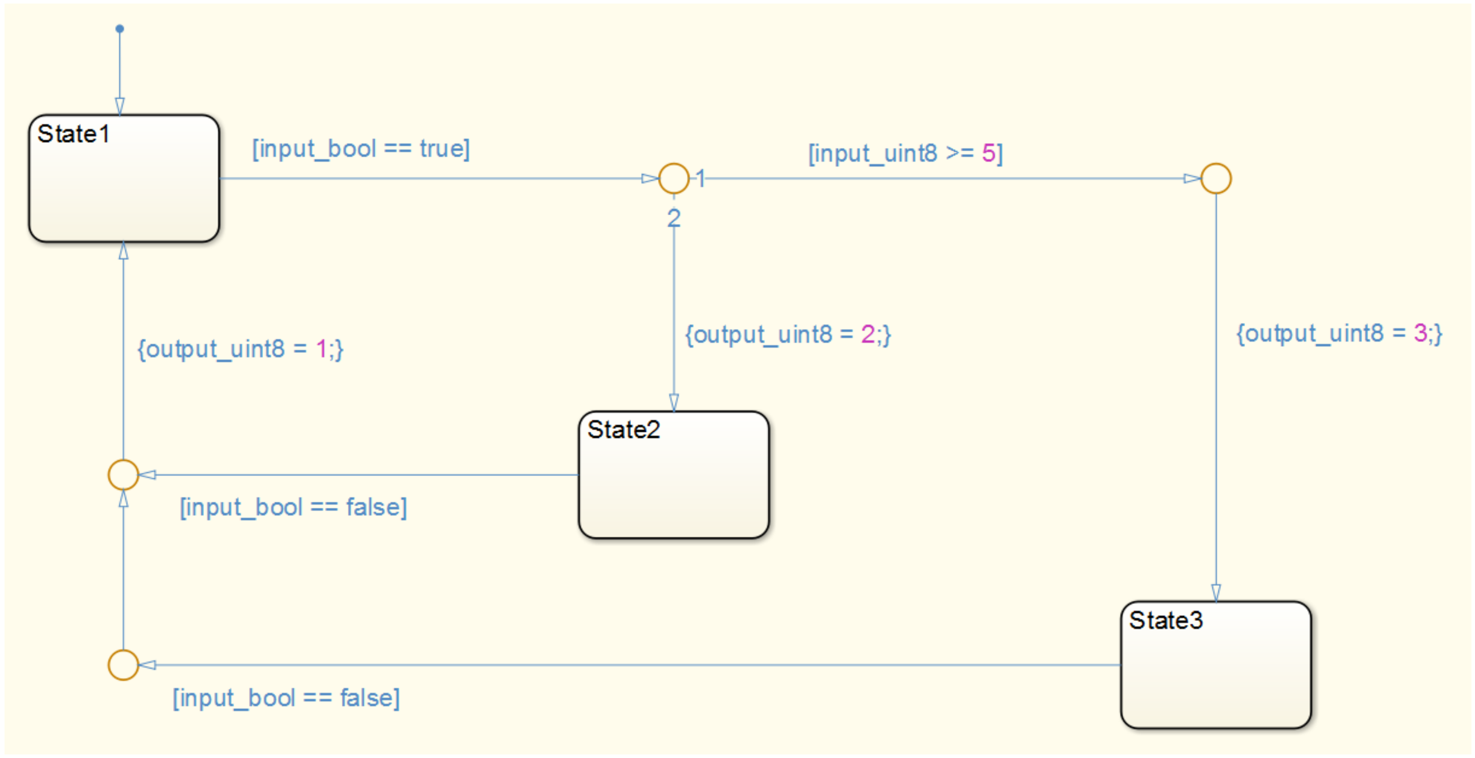
- Conditions
  - One or more statements which can be true or false
- Action
  - An Action assigning a specific value to an output variable
- Event or Message Trigger
  - Additional object that can trigger a transition



Elements in Stateflow

Execution Order

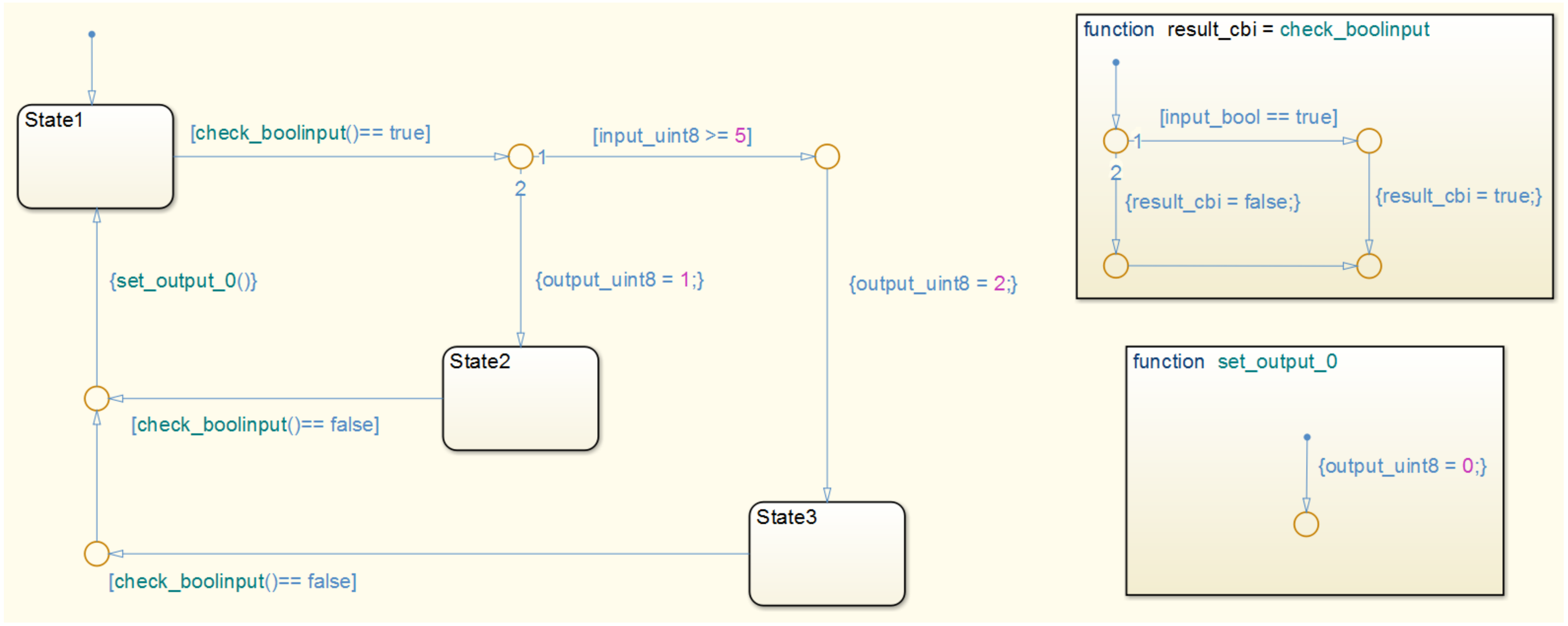
- Stating the order in which the following conditions are examined



Elements in Stateflow

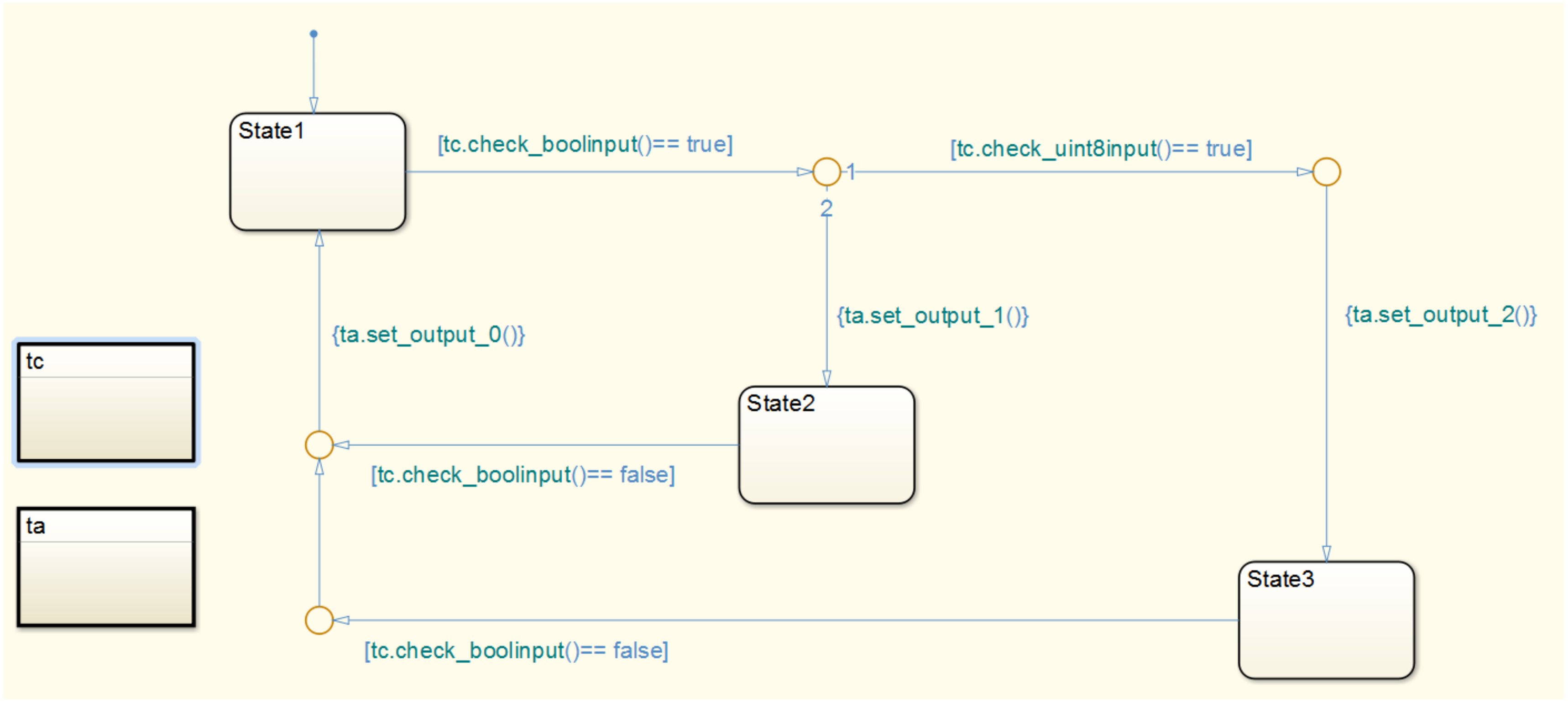
Graphical Function

- Functions which can be used multiple times whithin a chart
- Increase of Clarity

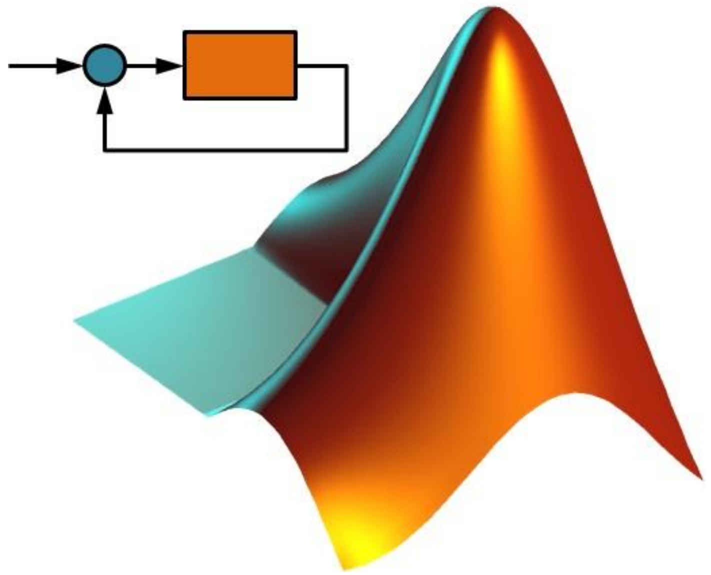


Box

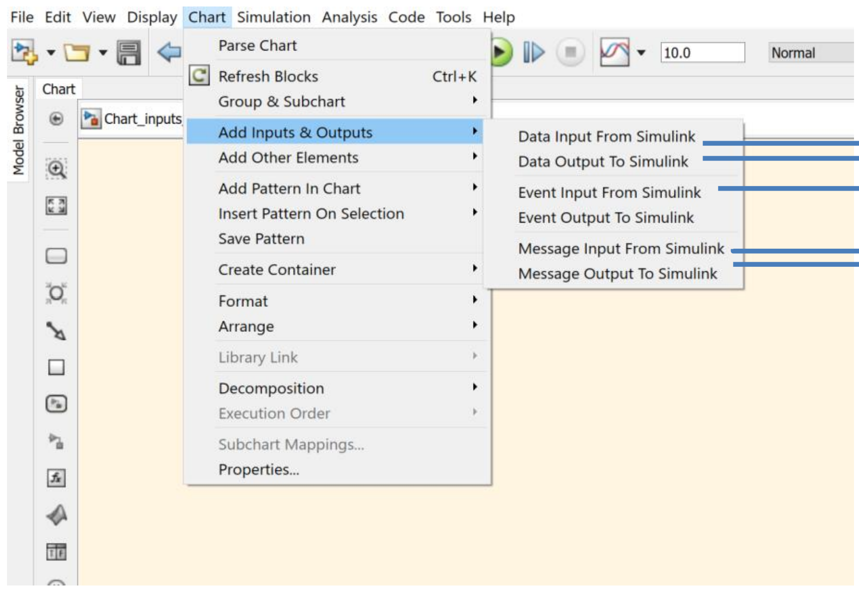
- A graphical Element to combine multiple graphical functions


State1
[tc.check_boolinput()== true]
1
2
[tc.check_uint8input()== true]
{ta.set_output_0()}
{ta.set_output_1()}
{ta.set_output_2()}
tc
ta
State2
[tc.check_boolinput()== false]
State3
[tc.check_boolinput()== false]

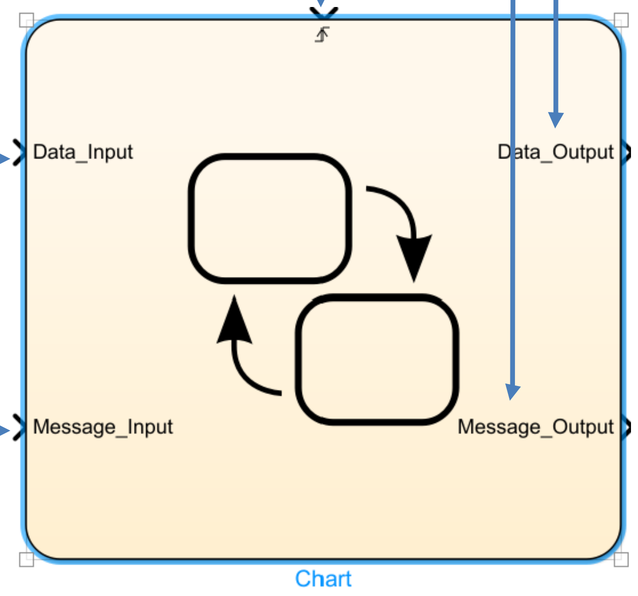
3. Data Management



Data Management: Adding Inputs or outputs to state charts



Inputs for a statechart over
Chart → Add Inputs & Outputs



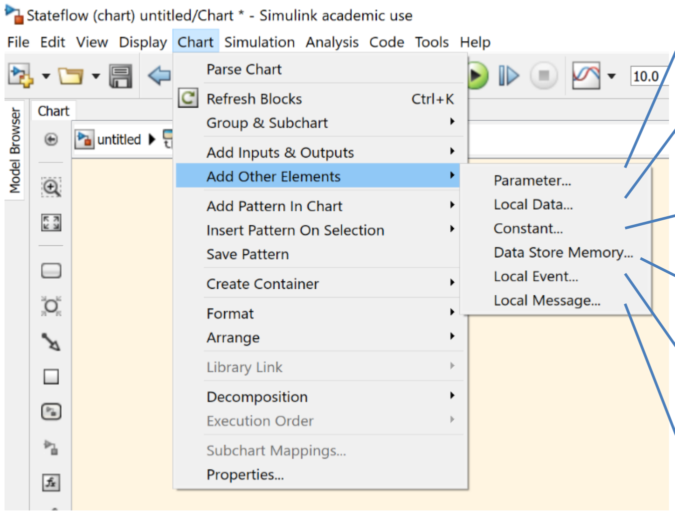
Events:
Used for triggers, which are change based
Data:
Used for checking absolute values

! Each event can be replaced by a data input, using a change checking before the chart

Message:
Used for queued messages between state charts

Data Management: Adding other elements to state charts

- Besides data input/output, stateflow uses a number of local data, which can be defined over ***Chart → Add Other Elements***



Parameter: Sharing a simulink parameter, given in Workspace, with the statechart.

Local Data: Variables only used within the chart, e.g. a temporary variable, which is only used during a transition.

Constant: Values, which shall be constant over the whole chart. e.g. „PI = 3.14146“

Data Store memory: Variables used within the chart and being stored in the central data store memory

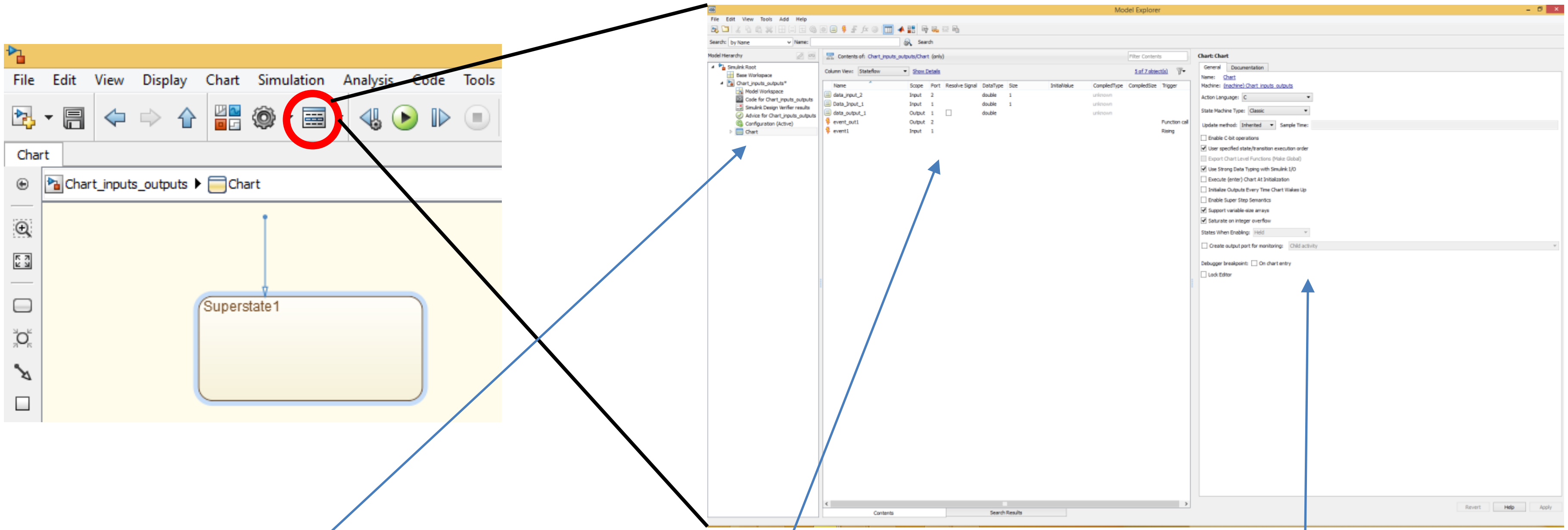
Local Event: Events, only used within the chart, e.g. a temporary event, which is only used during a transition.

Local Message: Messages which are queued, only used within the chart, e.g. to trigger a transition

Lehrstuhl für Flugsystemdynamik

Data Management: Editing signals of a statechart

Existing signals can be edited using the model explorer



Contents of: Chart_inputs_outputs/Chart (only)

Column View: Stateflow Show Details 5 of 7 object(s)

Name	Scope	Port	Resolve Signal	DataType	Size	InitialValue	CompiledType	CompiledSize	Trigger
data_input_2	Input	2		double	1		unknown		
Data_Input_1	Input	1		double	1		unknown		
data_output_1	Output	1	☐	double			unknown		
event_out1	Output	2							Function call
event1	Input	1							Rising

Chart: Chart

General | Documentation

Name: Chart
Machine: (machine) Chart_inputs_outputs
Action Language: C
State Machine Type: Classic
Update method: Inherited Sample Time:
- ☐ Enable C-bit operations
- ☑ User specified state/transition execution order
- ☐ Export Chart Level Functions (Make Global)
- ☑ Use Strong Data Typing with Simulink I/O
- ☐ Execute (enter) Chart At Initialization
- ☐ Initialize Outputs Every Time Chart Wakes Up
- ☐ Enable Super Step Semantics
- ☑ Support variable-size arrays
- ☑ Saturate on integer overflow

States When Enabling: Held
☐ Create output port for monitoring: Child activity

Debugger breakpoint: ☐ On chart entry
☐ Lock Editor

Area 1: Administration of all subsystem in the model

Area 2: Administration of all signals (which port, which datatype etc,…). Many Data of area 3 also editable here

Area 3: Signal details Administration of the signal, activated in area 2 (signal name, dimension etc.)

4. Action Language

Action Language: Chart preferences

- Action language forms commands for the statechart.
- General language properties have to be set in the chart properties
  → ***„File/Model Properties/Chart Properties“***
- The following settings are important for the course:

Select the basis language for the action language: „C“ and „MATLAB“ is possible. For the course, we will use „C“ language

Choose „Mealy“ or „Moore“ state machine types → detailed explanation on the following slides

If bit-operations shall be used (bitwise „AND“,“OR“,etc,…), the option has to be activated

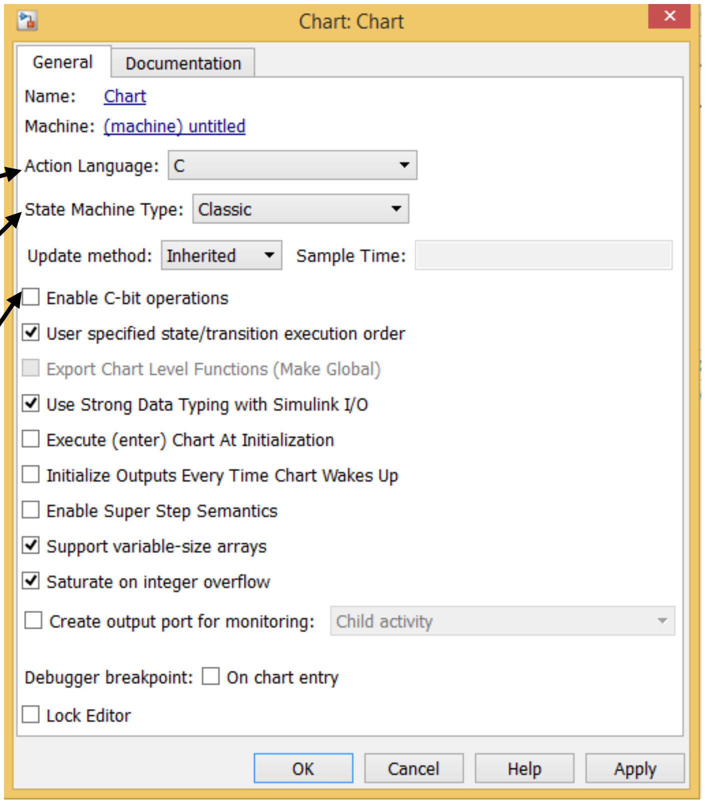


Chart: Chart

General | Documentation

Name: Chart
Machine: (machine) untitled
Action Language: C
State Machine Type: Classic
Update method: Inherited Sample Time:

- [ ] Enable C-bit operations
- [x] User specified state/transition execution order
- [ ] Export Chart Level Functions (Make Global)
- [x] Use Strong Data Typing with Simulink I/O
- [ ] Execute (enter) Chart At Initialization
- [ ] Initialize Outputs Every Time Chart Wakes Up
- [ ] Enable Super Step Semantics
- [x] Support variable-size arrays
- [x] Saturate on integer overflow
- [ ] Create output port for monitoring: Child activity

Debugger breakpoint: [ ] On chart entry
- [ ] Lock Editor

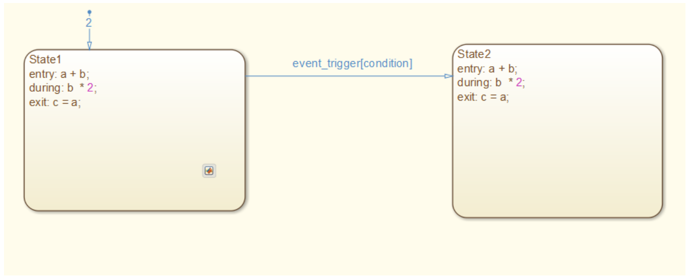
OK Cancel Help Apply

Action Language: Mealy vs. Moore

Moore charts: The output of the chart is only dependent on the actual active state of the chart.

The Matlab documentation specifies key aspects for Moore Charts:

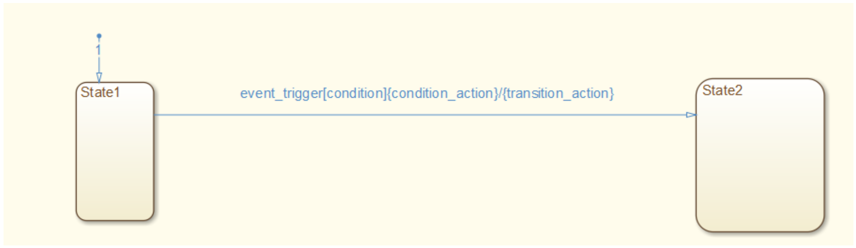
- Outputs depend only on the current state, not the next state.
- Outputs do not depend on previous outputs.
- Chart must compute outputs only in states, not in transitions.
- Chart must compute outputs before updating state.



Mealy charts: The output of the chart is dependent on the actual state AND on the current input

The Matlab documentation specifiedkey aspects for Mealy Charts:

- Outputs never depend on previous outputs.
- Outputs never depend on the next state.
- Chart wakes up periodically based on a system clock.

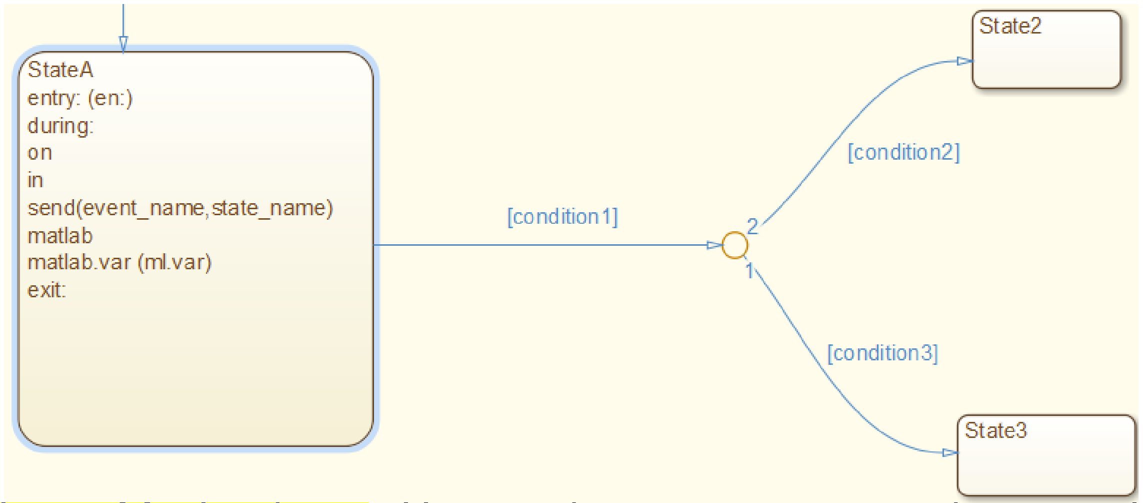

1
State1
event_trigger[condition]{condition_action}/{transition_action}
State2

Action language: Moore Chart example

The following lists show a set of commands, which can be used within a state.

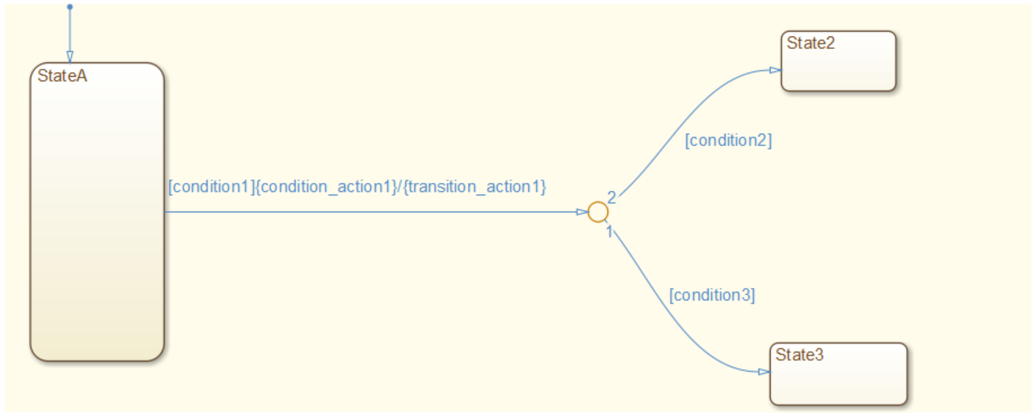
Command	Description
in (state_name)	True, if the state „state_name“ is active.
on event_name action_name	Action is executed, if even „event_name“ is true.
Change (data_name)	Sends an event, if value in „data_name“ changes

Command	Description
entry: (or en:) action_name	Action is executed once at state entry
during: (or du:) action_name	Action is executed each time step during state state is active
exit (or ex:) action_name	Action is executed, before exit of the state



Within this practical course, we will use Mealy charts. Hence, the upper commands are only explained very shortly here.

Action language: Mealy chart example



Label name	Description
(Event_trigger)	The event_trigger is true, if the specified „event_input“ is true → *not displayed here since it is not covered within this course*
Condition	If a condition is specified, the condition has to be true additionally to the event trigger
Condition action	is evaluated, if the condition is true
	Upper example: ***Condition_action1*** *is evaluated, if* ***„condition1“*** *is true, independent on „condition2“ and „condition3“*
Transition action	is evaluated, if the transition is executed.
	Upper example: ***„transition_action1“*** *is only evaluated, if* ***„condition1“*** *is true, and* ***„condition2“*** *or* ***„condition3“*** *is true, so that a transition is executed.*

Action language: Numerical operators for action language

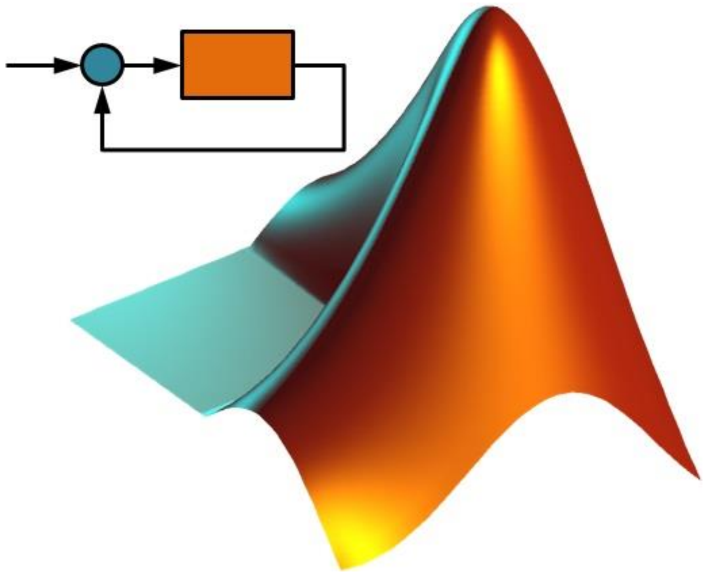
The following list shows the most important operations, which can be used as conditions and actions. The lists do not contain all possible operations

Numeric operations	
a + b or a- b	Addition /Subtraction
a * b or a / b	Multiplication /Division
a %% b	Modulo

Logical operations (conditions)	
a == b	a is equal b
a ~= b or *a!=b*	a is unequal b
a > b or *a < b*	a is greater /less than b
a > =b or *a =< b*	a is greater or equal / less or equal than b
a && b or *a || b*	a AND b (logical AND) / a OR b (logical OR)

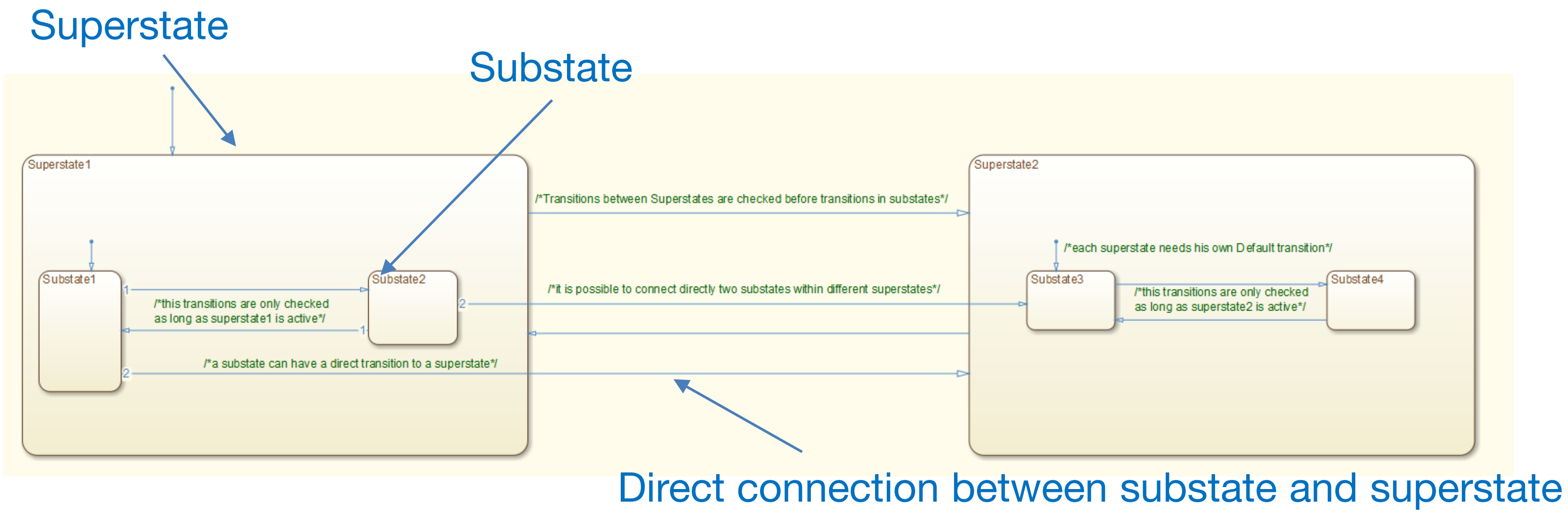
Logical operations (actions)	
a = b	*Set a equal b*
a++ or a--	*Equal to a = a+1 / a = a-1*
a = value or *a:= value*	*Set a equal a value*
a + = value or *a -= value*	*Equal to a = a+value / a = a-value*
*a * = value* or *a /= value*	*Equal to a = a*value / a = a/value*

5. Structure and Hierarchy



Structure and Hierarchy: Overview Stateflow structure

- A state can be organized itself in a number of states.
- The states are therefore in general called „Superstates“ and „Substates“
- A Substate can directly be connected to another superstate and vice versa
  - If an substate gets inactive caused by such a connection, the corresponding superstate gets inactive, too.
- Be careful: Usage of substates often becomes difficult to understand !!

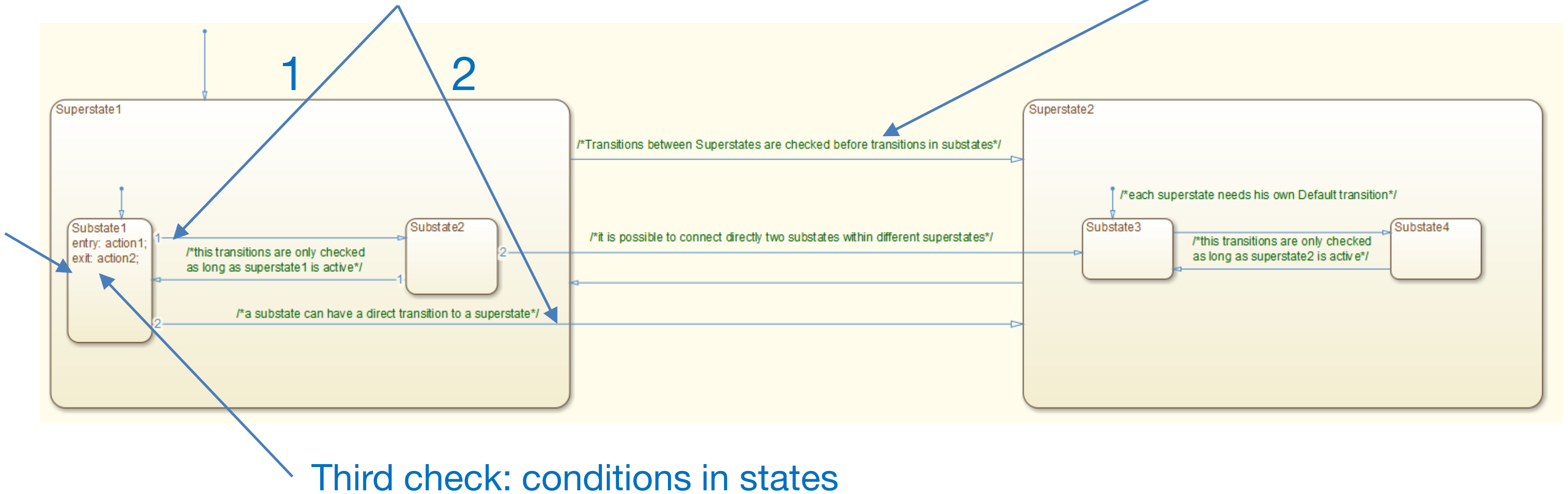

Superstate
Substate
Superstate1
Substate1
Substate2
1
/*this transitions are only checked as long as superstate1 is active*/
1
2
2
/*a substate can have a direct transition to a superstate*/
/*Transitions between Superstates are checked before transitions in substates*/
/*it is possible to connect directly two substates within different superstates*/
Superstate2
/*each superstate needs his own Default transition*/
Substate3
Substate4
/*this transitions are only checked as long as superstate2 is active*/
Direct connection between substate and superstate

Structure and Hierarchy: Checking order within Statecharts

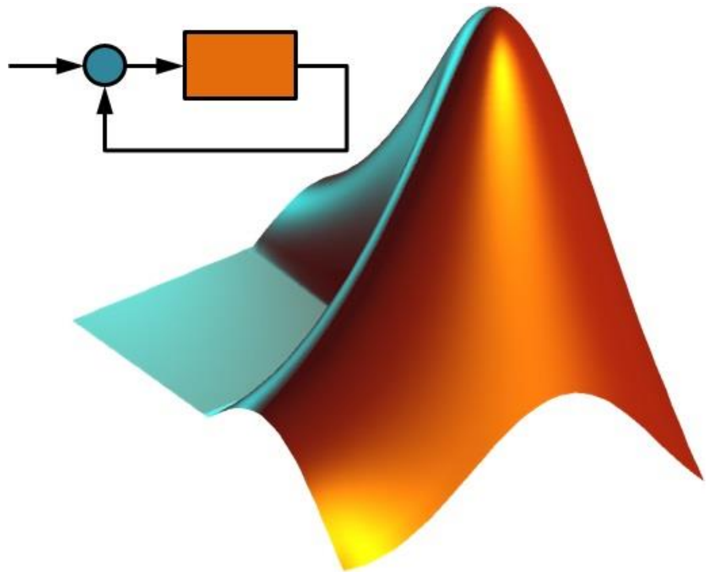
Each statechart has a strict order, which conditions are checked successively

1. Transitions beginning at Superstates are checked before transitions beginning at substates
2. Transitions are checked using the execution order, independently on the destination of the transition (to other substate or to superstate)
3. Transition conditions are checked before conditions within states (Moore charts or classical charts)

Assumption: Substate1 in Superstate1 is active


Second check: Transitions in active superstate, According to execution order
First check: transition between superstates
1
2
Superstate1
Substate1
entry: action1;
exit: action2;
Substate2
/*this transitions are only checked as long as superstate1 is active*/
/*a substate can have a direct transition to a superstate*/
/*Transitions between Superstates are checked before transitions in substates*/
/*it is possible to connect directly two substates within different superstates*/
Superstate2
/*each superstate needs his own Default transition*/
Substate3
Substate4
/*this transitions are only checked as long as superstate2 is active*/
Third check: conditions in states

6. Coding Guidelines / Lessons Learned



Coding Guidelines / Lessons Learned

- Coding Guidelines
  - MAAB (Mathworks Automotive Advisory Board)
    - Conditions horizontal, Actions vertical
  - Mathworks High Integrity Guidelines
  - Mathworks Code Generation Guidelines

- Lessons Learned
  - Simulink Level Structure
  - No Busses used for Input/Output
  - No Subcharts
  - Mealy Charts with C-Language
  - Enumerations for internal logics
  - High Usage of Design Verifier
    - Coverage Analysis (Find missing coverage objectives)
    - Design Error Detection (Dead Logic / Divide by Zero, Array Index)
    - Property Proving