Stateflow

30 January 2021 01:29 AM

1) Combination logic ?
   - Combinational Logic Circuits are memoryless digital logic circuits whose output at any instant in time depends only on the combination of its inputs

2) Sequential logic ?
   - sequential logic is a type of logic circuit whose output depends not only on the present value of its input signals but on the sequence of past inputs, the input history as well.

3) In parallel Stateflow state execution depends on each other or not?
   - No, the activity within parallel states is essentially independent on each other.

4) What is object ?
   - Object is a Identifier.
   - in computer science, an object can be a variable, a data structure, a function, or a method, and as such, is a value in memory referenced by an identifier.

5) Temporal logic operator.
   - After(n,sec)
     After(n,msec)
     After(n,usec)
     - Returns true if the specified time have elapsed since the associated state became active. Otherwise, the operator returns false
     - Eg. Set the temp variable to LOW every time that the chart wakes up, starting when the associated state is active for at least 12.3 seconds.

       ```
       on after(12.3,sec):
         temp = LOW;
       ```
   - at(n,sec)
     - Returns true if exactly specified time have elapsed since the associated state became active. Otherwise, the operator returns false.
     - Eg. Set the temp variable to HIGH if the state has been active for exactly 12.3 seconds.

       ```
       on at(12.3,sec):
         temp = HIGH;
       ```
   - before(n,sec)
     before(n,msec)
     before(n,usec)
     - Returns true if fewer than n units of time have elapsed since the associated state became active. Otherwise, the operator returns false.
     - Eg. Set the temp variable to MED every time that the chart wakes up, but only if the associated state has been active for fewer 12.3 seconds.

       ```
       on before(12.3,sec):
         temp = MED;
       ```
   - every(n,sec)
     - Returns true every n seconds since the associated state became active. Otherwise, the operator returns false
     - Eg. Increment the temp variable by 5 every 12.3 seconds that the state is active.

       ```
       on every(12.3,sec):
           temp = temp+5;
       ```
   - temporalCount(sec)
     temporalCount(msec)
     temporalCount(usec)

- Returns the length of time that has elapsed since the associated state became active.
- Eg. Store the number of milliseconds since the state became active.
  en,du:
  y = temporalCount(msec);

- elapsed(sec)
  - Returns the length of time that has elapsed since the associated state became active. Equivalent to temporalCount(sec).
  - Eg. Store the number of seconds since the state became active.
    en,du:
    y = elapsed(sec);
- count(C)
  - Returns the number of times that the chart has woken up since the conditional expression C became true and the associated state became active.
  - The Stateflow chart resets the value of the count operator if the conditional expression C becomes false or if the associated state becomes inactive.
  - Eg. Transition out of the associated state when the variable x has been greater than or equal to 2 for longer than five chart executions.
    [count(x>=2) > 5]
- duration(C)
  - Returns the number of seconds since the conditional expression C became true and the associated state became active.
  - The Stateflow chart resets the value of the duration operator if the conditional expression C becomes false or if the associated state becomes inactive.
  - Eg. Transition out of the state when the variable x has been greater than or equal to 0 for longer than 0.1 seconds.
    [duration(x>=0) > 0.1]

6) How many scope data we have in Stateflow?
   - Input, Output, Local, Parameter, Constant and Data Store Memory.

7) What is Event in Stateflow and is there any limit for events ?
   - An event is a Stateflow® object that can trigger actions in one of these objects:
   - A parallel state in a Stateflow chart.
   - Another Stateflow chart.
   - A Simulink® triggered or function-call subsystem.
   - For simulation purposes, there is no limit to the number of events in a Stateflow chart. However, for code generation, the underlying C compiler enforces a theoretical limit of 2^31-1 events.

8) What is state in Stateflow ?
   - A state describes an operating mode of a reactive system(reactive system is a event driven system). In a Stateflow® chart, states are used for sequential design to create state transition diagrams.
   - States can be active or inactive. The activity or inactivity of a state can change depending on events and conditions. The occurrence of an event drives the execution of the state transition diagram by making states become active or inactive. At any point during execution.

9) State Decomposition ?
   - Every state has decomposition that dictate what types of sub state the state can contain
     State decomposition can be exclusive (OR) or parallel (AND).
   - Exclusive (OR) State Decomposition :-
     - Substates with solid borders indicate exclusive (OR) state decomposition. Use this decomposition to describe operating modes that are mutually exclusive. When a state has exclusive (OR) decomposition, only one substate can be active at a time.
   - Parallel (AND) State Decomposition :-
     - Substates with dashed borders indicate parallel (AND) decomposition. Use this decomposition to describe

concurrent operating modes. When a state has parallel (AND) decomposition, all substates are active at the same time.

10) What is super step semantics in Stateflow ?
    - By default Stateflow chart execute once for each input event or time step
    - When you enable super step semantics, a Stateflow chart executes multiple times for every active input event or for every time step
    - If you are modelling a system that must react quickly to input you can enable super step semantics by right click in the chart block space go to properties and check the enable super step semantics option
    - In a super step semantics your chart respond faster to input but performs more computation in each time step.
    - When generating code for and embedded system target make sure that the chart can finish the computation in a single time step.
    - For simulation targets, specify whether the chart goes to the next time step or generates an error if it reaches the maximum number of transitions prematurely. However, in generated code for embedded targets, the chart always goes to the next time step after taking the maximum number of transitions.

11) State Action Types
    - Entry Action :- Entry actions are executed when a state becomes active.
    - During Action :- During actions are executed when a state is active, an event occurs, and no valid transition to another state or the current state is available.
    - Exit Action :- Exit actions are executed when a state is active and there is a valid transition from current state to other state.
    - Bind Action :- we can bind the data and events to a state by using a bind action. A bind action consists of the prefix bind followed by a colon (:) and one or more events or data. To separate multiple events and data, use semicolons or commas. You can also enter the events and data on separate lines.
      Only a state and its children can change data or broadcast events bound to that state. Other states can read the bound data or listen for the bound event, but they cannot change the bound data or send the bound events.
    - On Action :- On actions are executed when the state is active and it receives an event or message. On actions consist of the prefix on followed by a unique event event_name or message message_name, a colon (:), and one or more actions. To separate multiple on actions, use semicolons or commas. You can also enter the actions on separate lines.
      You can specify actions for more than one event or message. For example, if you want different events to trigger different actions, enter multiple on action statements in the state action label:
      Eg. on after(12.3,sec):
      temp = LOW;
      If multiple events occur at the same time, the corresponding on actions execute in the order that they appear in the state action label. For instance, in the previous example, if events ev1 and ev2 occur at the same time, then action1() executes first and action2() executes second. See Execution of a Stateflow Chart.

16) Data Store Memory in Stateflow.
    - Stateflow® charts share global data with Simulink models by reading from and writing to data store memory symbolically.
    - To access global data from a chart, bind a Stateflow data object to a Simulink data store. After we create the binding, the Stateflow data object becomes a symbolic representation of the Simulink data store memory. You can then use this symbolic object to store and retrieve global data
    - After binding a Stateflow data object to a Simulink data store, you can store and retrieve global data in state and transition actions. The data object acts as a global variable that you reference by its symbolic name. When you store numeric values in this variable, you are writing to the Simulink data store memory. When you retrieve numeric values from this variable, you are reading from the data store memory.

17) Data Store Memory in Stateflow.
    - We can share data between different region of model using data store memory block.
    - Reading and writing data is totally depend on where we added data store memory block.
    - If the Data Store Memory block is in the top-level system, Data Store Read and Data Store Write blocks anywhere

in the model can access the data store.
- If the Data Store Memory block is in a subsystem, Data Store Read and Data Store Write blocks in the same subsystem or in any subsystem below it in the model hierarchy can access the data store.
- If we want to access the data from the atomic subsystem then we can use the data store memory block.

18) Stateflow Chart
- A Stateflow® chart is a graphical representation of a finite state machine. States and transitions form the basic elements of the system. You can also represent stateless flow charts.
- A finite state machine is a representation of an event-driven (reactive) system. In an event-driven system, the system responds to an event by making a transition from one state (mode) to another. This transition occurs if the condition defining the change is true.
- You can also use a state machine to represent the automatic transmission of a car. The transmission has these operating states: park, reverse, neutral, drive, and low. As the driver shifts from one position to another, the system makes a transition from one state to another, for example, from park to reverse.
- A Stateflow chart can use MATLAB or C as the action language to implement control logic

19) Temporal Logic
- Temporal logic controls the execution of a chart in terms of time. In state actions and transitions, you can use two types of temporal logic:
  - Absolute time temporal logic
    Absolute-time temporal logic tracks the elapsed time since a state became active.
  - Event base temporal logic
    Event-based temporal logic tracks recurring events.

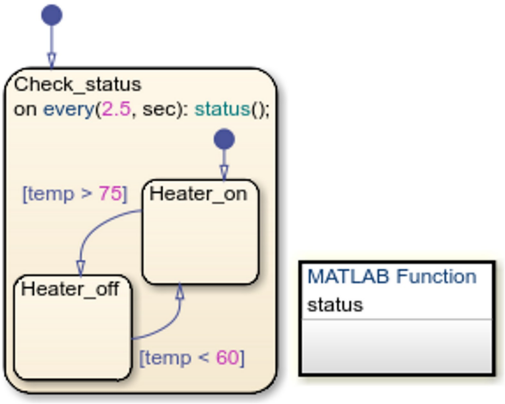
- Best Practices for Temporal Logic
- Do Not Use Temporal Logic on Transition Paths Without A Source State
  - The value of a temporal logic operator depends on when its associated state became active. To ensure that every temporal logic operator has a unique associated state, only use these operators in:
    - State on actions
    - Actions on transition paths that originate from a state
  - Do not use temporal logic operators on default transitions or on transitions in graphical functions because these transitions do not originate from a state.

- Use Absolute-time Temporal Logic Instead of tick in Charts in Simulink Models
  - In charts in a Simulink model, the value of delay expressions that use absolute-time temporal logic are semantically independent of the sample time of the model. In contrast, delay expressions that use temporal logic based on the implicit event tick depend on the step size used by the Simulink solver.
  - Additionally, absolute-time temporal logic is supported in charts that have input events. The implicit event tick is not supported when a Stateflow chart in a Simulink model has input events.

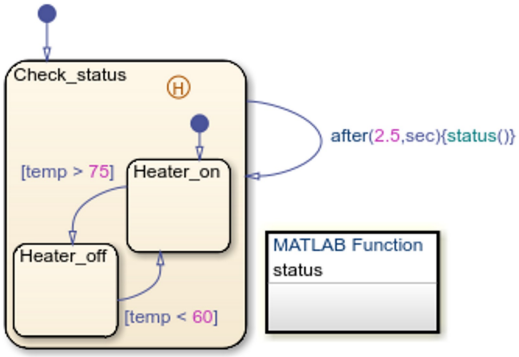
- Do Not Use at for Absolute-Time Temporal Logic in Charts in Simulink Models
  In charts in a Simulink model, using at as an absolute-time temporal logic operator is not supported. Instead, use the after operator.

To prevent a run-time error, change the transition label to after(5.33, sec).

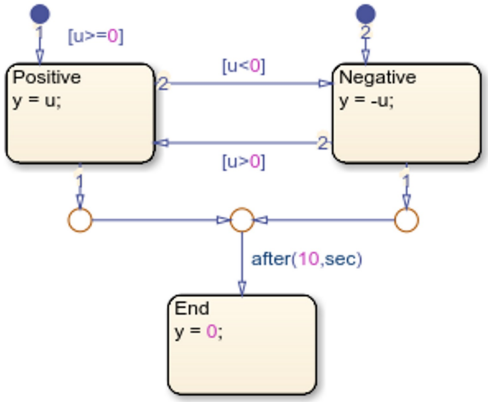
- Do Not Use every for Absolute-Time Temporal Logic in Charts in Simulink Models
  - In charts in a Simulink model, using every as an absolute-time temporal logic operator is not supported. Instead, use an outer self-loop transition with the after operator. For example, suppose that you want to print a status message for an active state every 2.5 seconds during chart execution.



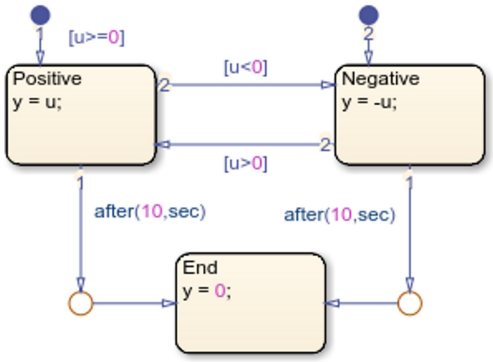
To prevent a run-time error, replace the state action with an outer self-loop transition



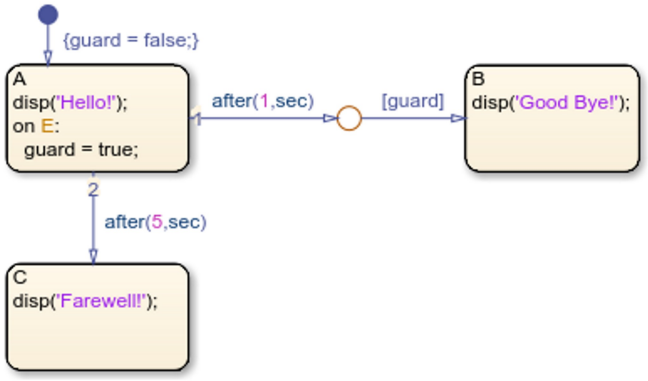
- Do Not Use Temporal Logic in Transition Paths with Multiple Sources in Standalone Charts in MATLAB
  - Standalone charts in MATLAB do not support the use of temporal logic operators on transition paths that have more than one source state. For example, this standalone chart produces a run-time error because the temporal logic expression after(10,sec) triggers a transition path that has more than one source state.



- To resolve the issue, use temporal logic expressions on separate transition paths, each with a single source state.



- Avoid Mixing Absolute-time Temporal Logic and Conditions in Transition Paths of Standalone Charts in MATLAB
  - In standalone charts in MATLAB, the operators after, at, and every create MATLAB timer objects that generate implicit events to wake up the chart. Combining these operators with conditions on the same transition path can result in unintended behaviour:
  - If a condition on the transition path is false when the timer wakes up the chart, the chart performs the during and on actions of the active state.
  - The chart does not reset the timer object associated with the operators after and at. If the condition on the transition path becomes true at a later time, the transition does not take place until another explicit or implicit event wakes up the chart.
  - For example, in this chart, the transition path from state A to state B combines the absolute-time temporal logic trigger after(1,sec) and the condition [guard]. The transition from state A to state C has the absolute-time temporal logic trigger after(5,sec). Each transition is associated with a timer object that generates an implicit event. Initially, the local variable guard is false



- When you execute the chart, state A becomes active. The chart performs the entry action and displays the message 'Hello!'. After 1 second, the timer associated with the transition from A to B wakes up the chart. Because the transition is not valid, the chart executes the during action in state A and displays the message 'Hello!' a second time.
- Suppose that, after 2 seconds, the chart receives the input event E. The chart executes the on action in state A and changes the value of guard to true. Because the chart does not reset the timer associated with the operator after, the transition from A to B does not take place until another event wakes up the chart.
- After 5 seconds, the timer associated with the transition from A to C wakes up the chart. Because the transition from A to B is valid and has a higher execution order, the chart does not take the transition to state C or display the message 'Farewell!'. Instead, state B becomes active and the chart displays the message 'Good bye!'.

- Use Charts with Discrete Sample Times for More Efficient Code Generation
  - The code generated for discrete charts that are not inside a triggered or enabled subsystem uses integer counters to track time instead of the time provided by Simulink. This behaviour allows for more efficient code generation in terms of overhead and memory, and enables this code for use in software-in-the-loop (SIL) and processor-in-the-loop (PIL) simulation modes. For more information, see SIL and PIL Simulations (Embedded Coder).

20) What is dynamic system ?
- If a system depends upon the past and future value of the signal at any instant of the time then it is known as dynamic system.
- Unlike static systems, these are not memory less systems. They store past and future values. Therefore, they require some memory.

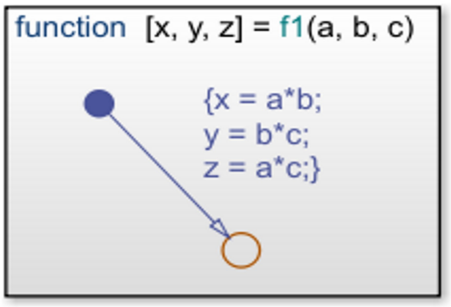
21) What is junction ?
- Junction divide a transition into transition segment.
- A connective junction represents a decision point in a transition path we can combine transition and connective junction to create path from common source to multiple destination or from multiple sources to a common destination.

22) What is transition ?
- A transition is a line with arrowhead that links one graphical object to another.
- In most cases a transition represents the passage of the system from one mode to another.
- A transition typically connects a source and a destination object. The source object is where the transition begin and the destination object is where the transition ends.

23) What is condition ?
- Condition specifies a Boolean expression that when it gets true it validates the transition.
- Enclose the condition in square brackets ([]) if no condition is specified an implied condition evaluates to true.

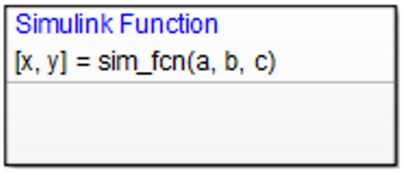
24) What is history junction ?
- It represent historical junction decision points in the Stateflow chart.
- The decision points are based on historical data relative to state activity.
- Placing a history junction in a superstate indicate that historical state activity information is used to determine the next state to become active.
- The history junction applies to only level of hierarchy in which it appears.

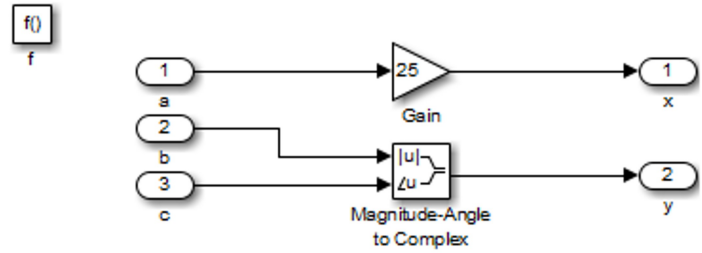
25) Graphical function in Stateflow.
- A graphical function in a Stateflow® chart is a graphical element that helps you reuse control-flow logic and iterative loops.
- You create graphical functions with flow charts that use connective junctions and transitions.
- You can call a graphical function in the actions of states and transitions.
- If you want to call the function within one state or subchart and its substates, put your graphical function in that state or subchart. That function overrides any other functions of the same name in the parents and ancestors of that state or subchart.
- If you want to call the function anywhere in a chart, put your graphical function at the chart level.
- For example, this graphical function has the name f1. It takes three arguments (a, b, and c) and returns three output values (x, y, and z). The function contains a flow chart that computes three different products of the arguments.



26) Matlab Function in Stateflow.
- To write the matrix oriented algorithms in stateflow then used this function.
- This type of function are useful for coding algorithm that are easily expressed using matlab instead of graphical stateflow.
- if provides optimization for generation efficient production quality c code for embedded applications.

27) Simulink Function In Stateflow.
- A Simulink® function is a graphical object that enables you to call a Simulink subsystem in the states and transitions actions.
- Simulink functions can improve the efficiency of your design and increase the readability of your model.
- A Simulink function can reside anywhere in a chart, state, or subchart. The location of a function determines the set of states and transitions that can call the function.
-



- 

28) Truth Table
   - Truth tables implement combinatorial logic design in a concise, tabular format. Typical applications for truth tables include decision making for:
     - Fault detection and management
     - Mode switching
   - Truth tables are supported only in Simulink®.
   - You can add a Truth Table block directly to your Simulink model, or you can define a truth table function in a Stateflow® chart, state, or subchart. Truth Table blocks in a Simulink model execute as a Simulink block, while truth table functions in a Stateflow chart execute only when you call the truth table function. The location of the function determines the set of states and transitions that can call the function.

29) Super State and Sub State
   - A state within the boundaries of another state indicate that inner state is a substate or child of the super state
   - The outer state is a parent of inner state or we can say it a supper state.

30) Discrete Time Signal
   - Discrete time signals are “the signals or quantities that can be defined and represented at certain time instants of the sequence.”
   - These are countable sets of number sequences. They are also called digitalized signals.

31) What is the difference between Stateflow and flowchart?

Stateflow	Flowchart
• Stateflow is for logical and state based system	• Flowchart is a graphical construct that models logic patterns by using connective junctions and transition
• in case of Stateflow default transition execute only single time.	• In flowchart the execution of the default transition condition take place every single time
• Stateflow contain the states	• the flowchart have the transition path and do not have the states
	• In flowchart junction provide decision branches between alternate transitions paths.