

Latest Features in Stateflow 7.3

R2009a

What's New in R2009a

- Simulation state snapshot
- Unintended backtracking diagnostic
- State inlining control
- New keywords: `true` and `false`

Simulation State Snapshot for Stateflow

Challenge

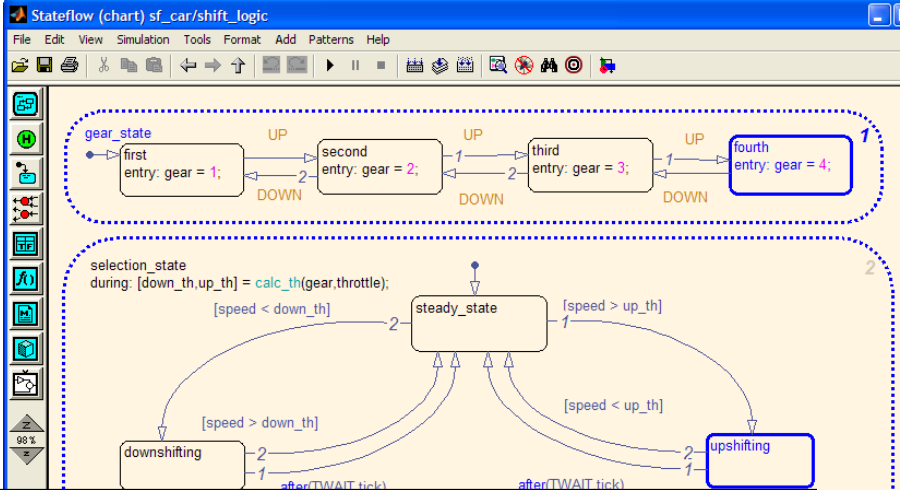
- Users could not efficiently conduct what-if analysis from any point within a simulation.
- Saved “final states” could not be used to restore simulation.

Solution

- Complete set of simulation states (SimState) can be saved in final states.

Benefit

- Save time by starting simulations from an important point
- Increase robustness by separating long simulations into smaller pieces
- Facilitate testing of different state configurations



```

>> c = xFinal.getBlockSimState('sf_car/shift_logic')
Block: "shift_logic" (handle) (active)
Path: sf_car/shift_logic

Contains:

+ gear_state "State (AND)" (active)
+ selection_state "State (AND)" (active)
gear "Block output data" double [1, 1]
up_th "Loca. scope data" double [1, 1]
down_th "Loca. scope data" double [1, 1]

>> c.gear.Value = 2;
>> c.gear
Description: 'Block output data'
DataType: 'double'
Size: '[1, 1]'
Range: [1x1 struct]
InitialValue: [1x0 double]
Value: 2

>> c.selection_state.upshifting.isActive
1
>> c.selection_state.upshifting.open
>> c.selection_state.upshifting.setActive
>> c.highlightActiveStates
>> c.checkStateConsistency
States are consistent!

>> xInitial = xFinal.setBlockSimState('sf_car/shift_logic', c);
  
```

Unintended Backtracking Diagnostic

Challenge

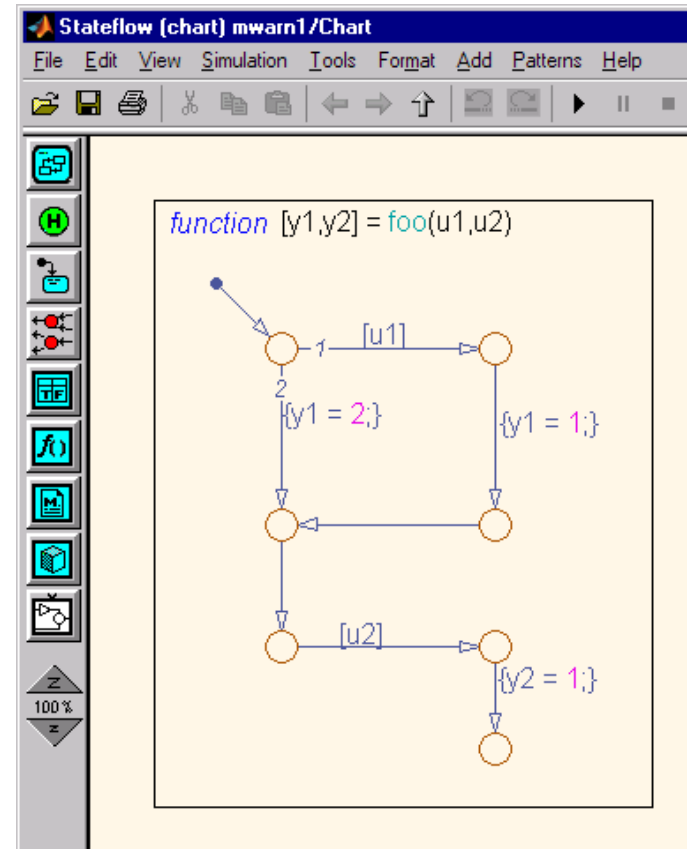
- Some flow graph constructs produce unexpected behavior due to backtracking semantics.

Solution

- Provide a detailed compile-time diagnostic to detect when flow graphs with possible unintended backtracking are constructed

Benefit

- Identify non-intuitive behavior at compile time rather than run time



This flow chart looks like a chain of two if-else statements, but actually has completely different meaning.

State Inlining Control

Challenge

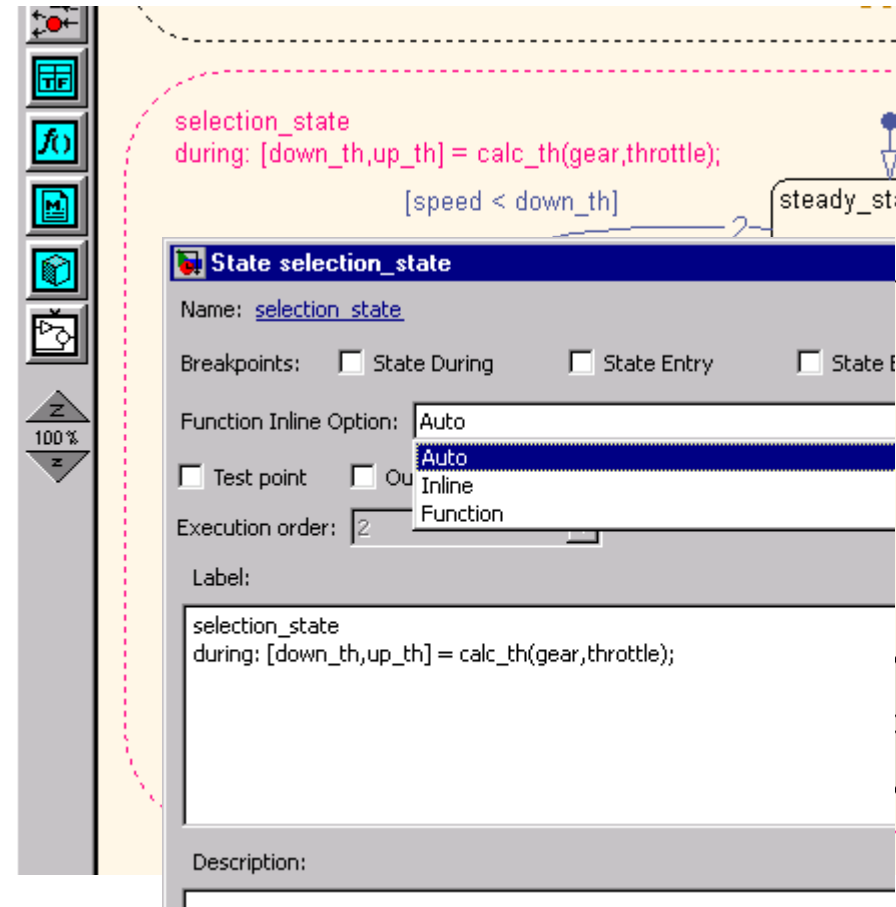
- Users could not control the structure of code generated from states.

Solution

- Include option to specify whether a state should be inlined in the generated code

Benefit

- Small perturbations to chart do not cause major changes to structure of generated code.
- Testing and validation tools are better supported.
- The one-to-one relationship between model and code makes it easier to manually inspect generated code.



New Keywords: `true` and `false`

Challenge

- Additional constants were necessary to define Boolean values for “true” and “false.”

Solution

- Provide `true` and `false` as Stateflow language keywords

Benefit

- No need to define redundant constants
- Uniform look and feel across charts

