Model-based Test Generation for Simulink/Stateflow
Meng Li and Ratnesh Kumar, ECE, Iowa State Univ. (mengl,rkumar@iastate.edu)
Phone: 515-294-8523; Fax: 515-294-8432

Simulink/Stateflow, a graphical model-based development tool by Mathworks, is commonly used for the design, simulation and code generation of reactive software and systems. The semantics of Simulink/Stateflow are able to be formalized and therefore can be targeted for static and dynamic analysis techniques. We have developed a recursive method that translates a Simulink/Stateflow model into an Input/Output Extended Finite Automata (I/O-EFA) which is suitable for formal analysis (at present model-based test-generation).

In this translation, Simulink primitives are considered "atomic" and each is uniquely translated into a canonical I/O-EFA model consisting of 4 locations. The compositions of the primitive blocks are translated using the rules of composition (for connection and conditioning) to preserve the semantics of the source model. Stateflow charts are also translated as canonical entities: Each state is treated as an atomic element with its own atomic model of 3 locations, and the hierarchical organization of states is recursively composed following the rules of parallel (AND) and sequential (OR) composition that we develop. Additionally rules are developed for refining the overall model to capture the more exotic features of Stateflow, such as event broadcasting and history junctions. Correctness of the translated model is proven following the discrete-time semantics of Simulink/Stateflow. This translation approach scales linearly with the size of the source model.

The resulting I/O-EFA model drives a test generation approach guaranteed to fully cover all computation paths. Other coverage criteria such as MCDC can also be met. The generated test suites can be used to test the generated code obtained through Real-Time Workshop, the code generation utility provided by Matlab. The test-generation at present is based on model-checking and it also reports the computation paths that are infeasible for which no tests can be found. To maintain decidability, the test generator takes into account the precision-level of arithmetic computations and also upper bounds the length of the test-inputs.

Our model translation has been implemented in Matlab and given a Simulink/Stateflow model, the input/output variables and sample time it outputs a semantically equivalent I/O-EFA model as a “flat” Stateflow chart that can itself be simulated in Matlab. The test generation approach has also been implemented in Matlab and interfaces with NuSMV to search for test inputs that activate each of the computation paths. The figure above shows a sample Simulink/Stateflow model (left top), its translated I/O-EFA model (right), and the test-inputs (left bottom). To further enhance the test-generation capabilities, we plan to interface our translation tool with more powerful mathematical programming tools such as CPLEX.