

Scoping and Testing in Statechart Based Models

Moshe Cohen
i-Logix, Inc.
22 3rd Avenue, Burlington, MA 01803
(617) 272-8090
Email: mc@ilogix or uunet!ilogix!mc

Abstract

A behavioral model described as a statechart is based not only on modes and transitions, but on events, conditions and different types of data items too. Although statecharts resolve the blow-up phenomena associated with state transition diagrams, a realistic system is not described as a single statechart. This paper introduces scoping of elements in a model based on several statecharts and its impact of testing.

A statechart is hierarchical in nature; It allows one to decompose states into sub-states. In addition to the hierarchy within a chart, one can use the mechanism of activity-charts to describe the data flow in the system as well its functional decomposition. In this case, the whole system is decomposed to a hierarchy of activities, the data flow between the activities is specified, and the behavior of each of these activities is described using statecharts. This in effect provides a mechanism of decomposing a system into a tree of statecharts, while at any given time, only some of them are active.

The visibility rules associated with statechart models are similar to those found in other top-down structured methods. Elements defined in an activity, can be referred to by any of its sub-activities and statecharts. It is also legal to define an element in some low level with the same name of another element in a higher level, and the semantics of such a case is as one expects to find in any method that supports scoping.

Statecharts, or behavioral models in general, can be used to describe more than just a model of the system being designed. One can design a "watchdog" statechart to observe the system during its operation and monitor its behavior. This is not part of the designed system; it may describe the environment's behavior, some global properties or scenarios derived from the requirements. As a watchdog, it can be executed in parallel to the system model, either as a Statechart or after being translated into VHDL - as a testbench or as a functional test-vectors generator.

A Statechart watchdog can operate on the inputs and outputs of the whole system. However, it is often very useful to monitor internal values or even inject faults by modifying them. To allow for this kind of testing, Statecharts watchdogs can "violate" the scoping rules, and elements used in the watchdogs are automatically resolved to those used in the system model. However, preserving this capability in the VHDL code generated for the testbench is not straight-forward.

Automatic code generation for the testbench must take into account the fact that VHDL is strongly typed and scoped. Therefore, if one needs to access local signals from the testbench, these signals must be made part of the inputs and outputs for the whole model. By using a statechart to model the system, the model does not have to be modified; The code generation process automatically produces the appropriate code for the system being modeled according to the watchdogs needs. This may involve both the generation and handling of signals (that are needed solely for testing purposes) and changing some of the local signals to global signals.