Specifying and Testing Software Components with Spec#

Wolfram Schulte

Microsoft Research, Redmond, Washington, USA
schulte@microsoft.com

Abstract

Spec# is a formal language for API contracts, based on AsmL (a wide spectrum formal specification language), which extends C# with constructs for preconditions, postconditions, object invariants, and model programs (behavioral contracts that take the history of the entire run into account). Spec# is the input to a suite of correctness tools. These tools include static and dynamic verification, a test case generator and a model checker. Our goal is that Microsoft product teams will be able to write Spec# contracts as simply or richly as they like and then drive all of their checking tools from this common contract.

We discuss Spec# and its use for testing. Our SpecExplorer tool provides a rigorous, systematic way to generate behavioral test cases. With SpecExplorer testers can search the space of all possible sequences of method invocations that 1) do not violate the pre- and postconditions of the system's contracts and 2) are relevant to a user-specified set of test properties. The resulting traces are algorithmically traversed to produce behavioral tests that cover all explored transitions. Spec#'s Runtime Conformance Checker is used for test verification. It uses a specially instrumented build to dynamically enforce contractual constraints that cannot be statically verified. More than just "assertion checking" familiar to all programmers, the conformance tool handles nondeterministic actions (such as user input and the handling of events) and the mapping of abstract state given in an interface's contract with concrete state defined by the implementer.

Key words: Formal specification, test-case generation, runtime monitoring, refinement.

This is a preliminary version. The final version will be published in Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs