Automatic Test Case Generation with NuSMV

Grygoriy Bunin

Rücker GmbH
Domagkstraße 13-15
D-80807 München
grygoriy.bunin@ruecker.de

Axel Schneider

Lucent Technologies Network Systems GmbH
Thurn-und-Taxis-Str. 10
D-90411 Nürnberg
aschneider@lucent.com

Christian Haubelt

University of Erlangen-Nuremberg, Germany
Am Weichselgarten 3
D-91058 Erlangen
haubelt@cs.fau.de

Jan Langer, Ulrich Heinkel

Chemnitz University of Technology, Germany
Reichenhainer Str. 70
D-09126 Chemnitz
{laja,heinkel}@infotech.tu-chemnitz.de

Abstract: Formal verification has become a key technology to ensure the quality of complex (hardware/software) systems. Although formal verification is widely used in the hardware design phase, it plays a minor role in software or even system development. In this paper, we describe how we used the model checker NuSMV to automatically generate executable test cases from an abstract system description.

For verification with model checking the properties of a system are described in temporal logic formulas (e.g. CTL, LTL) and proven against a formal system model. In case the property is not fulfilled, the model checker provides a counterexample violating the property. If used at all in industrial practice, formal techniques usually stop at this point and test cases for simulation or test are still specified manually. The idea of the approach described herein is: We first generate a formal model from an abstract system description, then take a well-known method of generating stimuli and responses using model checking [CSE1996] and finally convert the raw model checker output into automatically executable test scripts for a particular test environment.

As real-world telecommunication example we use the so-called Timing Link Switch functionality. This is a mechanism to select, based on several criteria, the most appropriate input signal for synchronization of an optical data transmission system. The functionality is described in a text document and formalized using the specification language ADeVA [HHG2003]. From the abstract ADeVA model an SMV model is generated with our code generator. Negated properties are used to generate test stimuli and responses [WGHH2004]. However, the raw model checker output is not yet in a suitable format for test execution. The NuSMV output needs to be transformed into an executable test script by extracting the relevant signals and mapping them to function calls of the system's user interface. The whole process can be summarized as follows:

- Describe the functionality with ADeVA, generate a SMV system model
- Specify the system behavior, which should be tested, as CTL or LTL property
- Verify the property with NuSMV, correct the specification until the property holds
- Negate the property and run NuSMV again – the negated property fails
- Extract relevant signals from the counterexample and map to system function calls

Not all kinds of properties can be used to generate a test case. The following LTL template is an example, which may be used to specify an appropriate property: *G (<assumptions about system state> → <expected output values>)* where *<assumptions about system state>* contain the current values of internal or output signals (i.e. current system state) and actions applied to the system (i.e. input signal values). The second part *<expected output values>* contains the expected values of the output signals. This property expresses: if the first part is fulfilled, always (G = globally) the second part must be fulfilled. Negating a property means: every commitment in the second part is negated, the assumptions in the first part stay unchanged. In the negated property we claim that system states specified in the second part of the original property are never reachable, which is obviously false. Thus, NuSMV will always provide a counterexample in terms of input, internal and output signals. Considering only the input signals, we get the complete input sequence that leads from the initial system state to the desired state, as specified in the second part of the original property. The output signals define the expected result of a test. To map these signals from the counterexample to real function calls, we use a mapping database, maintained with our MappingEditor tool.

Bounded Model Checking may be used to find the shortest possible counterexample and therewith generate the shortest possible test case, which verifies the desired behavior. However, it must be kept in mind that the generated test case verifies only one out of (in general) many system state transitions, in which the considered behavior occurs. In future work, we will extend the test case generation to optimize the test coverage.

[WGHH2004] Wang F., Gossens S., Haas W., Heinkel U.: "Generierung von Testvorschlägen aus tabellarischen Spezifikationen", GI/ITG/GMM Workshop "Testmethoden und Zuverlässigkeit von Schaltungen und Systemen", Dresden, Germany, March 2004

[HHG2003] Haas W., Heinkel U., Gossens S.: "Semantics of a Formal Specification Language for Advanced Design and Verification of ASICs (ADeVA)", 11th E.I.S.-Workshop, Erlangen, Germany, April 2003

[CSE1996] Callahan J., Schneider F., Easterbrook S.: "Automated Software Testing Using Model-Checking", Proceedings SPIN 1996 Workshop, Rutgers, NJ, August 1996