A Virtual Machine for State Space Generation

Michael Weber and Stefan Schürmans

Lehrstuhl für Informatik II
RWTH Aachen University
Aachen, Germany
{michaelw,stes}@i2.informatik.rwth-aachen.de

Abstract. The semantics of modelling languages are not always specified in a precise and formal way, and their rather complex underlying models make it a non-trivial exercise to reuse them in newly developed tools. We introduce a virtual machine-based approach for state space generation. The virtual machine's (VM) byte-code language is straight-forwardly implementable, facilitating reuse and making it an adequate target for translation of higher-level languages like the SPIN model checker's Promela. As added value, it provides precise operational semantics for modeling languages.

Common approaches in state-based model checking employ modeling languages like CSP, LOTOS, or Promela to describe of actual state spaces. These languages are usually non-trivial: in addition to concepts found in programming languages (scopes, variables, expressions etc.) they often provide features like process abstraction, non-determinism, guarded commands, synchronisation and communication primitives, timers, etc. Implementing an operational model of such languages for use in verification tools is consequently not straight-forward, even more so if the language is described informally only, and their static and operational semantics are incomplete at best, outdated, or entirely unavailable.

It comes at no suprise that researchers often find it easier to invent their own modeling language with informally specified semantics incompatible to existing tools which, as argued above, puts additional burden on end-users to switch tools, benchmark them, or consider them at all.

To remedy current shortcomings we propose a virtual machine-based approach to state space generation, in which high-level modeling languages are translated to byte-code instructions. Subsequent execution of such byte-code programs with our virtual machine yields state spaces for further use in model checkers. As additional benefit the translation process assigns precise operational semantics to given modeling languages.

The operational semantics of our virtual instruction set are straight-forward, and hence easy to implement from scratch. Moreover, our machine model can be augmented with additional constructs to handle timers, probabilities, etc. in a compositional way. Most byte-codes are simple operations, and based on preliminary tests we estimated a peak performance of around 500 operations per second per Megahertz, yielding about 1 MegaOps on a 2 GHz processor. For distributed state space generation our machine allows to be restarted from machine state snapshots.