Towards Expressive Specification and Efficient Model Checking

Jin Song Dong
National University of Singapore
dongjs@comp.nus.edu.sg

Jun Sun
National University of Singapore
sunj@comp.nus.edu.sg

System modeling is very important and highly non-trivial. The choice of specification language is an important factor in the success of the entire development. The language should cover several facets of the requirements and the model should reflect exactly (up to abstraction of irrelevant details) an existing system or a system to be built. The language should have a semantic model suitable to study the behaviors of the system and to establish the validity of desired properties. A formal model can be the basis for a variety of system development activities, e.g., system simulation, visualization, verification or prototype synthesis.

Over the last decade, many integrated formal specification languages have been proposed in order to precisely and concise model systems with not only complicated control flows but also complex data structures and operations. Examples include *Circus* [5] (i.e., an integration of CSP and the Z language), CSP-OZ [1] (i.e., an integration of CSP and Object-Z) and TCOZ [2] (i.e., an integration of Timed CSP and Object-Z). Integrated specification languages are very expressive, which makes mechanical analysis extremely difficult. In particular, all above languages incorporate (a large subset of) the Z language, which is based first-order logic and set theory and thus beyond the capability of mechanical verification techniques like model checking.

On the other hand, popular model checkers like SPIN, SMV and FDR are designed for specialized domains and are therefore based on restrictive modeling languages. For instance, Promela (supported by SPIN) is based on a subset of CSP for communicating network protocols. The input language of SMV is initially designed for specification of hardware circuits. A number of compositional operators which model common system behavior patterns are missing in both languages. FDR, which supports all operators of CSP, however, lacks support of shared variables or non-trivial data-types. Language limitations can be significant barrier to the practical verification of complex systems.

In this tutorial, we introduce our latest effort on combining the expressiveness of integrated formal specification languages with the power of mechanical system analysis method like model checking. We present a process analysis toolkit (PAT [3, 4], available at http://pat.comp.nus.edu.sg), which is a self-contained framework for system specification, simulation and verification. PAT supports a modeling language named CSP# (short for communicating sequential programs), which shares similar design principle with specification languages like TCOZ. Nonetheless, instead of relying on the Z language, CSP# mixes high-level modeling operators with low-level programs, for the purpose of flexible system modeling and efficient verification. In CSP#, data operations can be modeled as terminating sequential programs, which then can be composed using high-level compositional operators. *The idea is to treat sequential terminating programs, which may indeed be C# programs, as atomic events.* The result is a highly expressive modeling language which cover a wide range of application domains.

CSP# models are executable with complete operational semantics, and therefore subject to fully automated system verification techniques like model checking. PAT verifies CSP# models using state-of-art model checking techniques, e.g., on-the-fly explicit state model checking with partial order reduction. Besides new modeling techniques, PAT complements existing model checkers in a number of aspects. For instance, it supports an assertion language which allows LTL formulae constituted with propositions and events. It has dedicated algorithms for model checking under a variety of fairness constraints, which are often required for verification of liveness properties. CSP# and PAT have been applied a many systems including distributed algorithms, concurrent data objects, parameterized systems, etc. Previously unknown bugs have been identified.

References

[1] C. Fischer. CSP-OZ: a combination of object-Z and CSP. In *FMOODS'97*, pages 423–438. Chapman & Hall, Ltd., 1997.

[2] B. Mahony and J. S. Dong. Timed Communicating Object Z. *IEEE Trans. on Soft. Eng.*, 26(2):150–177, 2000.

[3] J. Sun, Y. Liu, J. S. Dong, and J. Pang. PAT: Towards Flexible Verification under Fairness. *CAV'09*, 2009. to appear.

[4] J. Sun, Y. Liu, J. S. Dong, and H. Wang. Specifying and Verifying Event-based Fairness Enhanced Systems. In *ICFEM'08*, volume 5256 of *LNCS*, pages 318–337. Springer, 2008.

[5] J. Woodcock and A. Cavalcanti. The Semantics of Circus. In *ZB 2002*, volume 2272 of *LNCS*, pages 184–203, 2002.