Automatic Analysis and Verification of Dependable Systems with Formal Methods

Li Xin, li-xin@jaist.ac.jp
Japan Advanced Institute of Science and Technology

1 Research Aim

Current practiced methods for software and system validation are mostly based on simulation and testing. The fundamental problem for these methods is that they can not cover all possible scenarios of system runs. Thus it can find errors but cannot guarantee. A promising alternative approach is formal verification, of which some successful methods are type system, model checking, theorem proving, etc. In particular, Model checking is a technique for automatic verification of reactive and concurrent systems and has been successfully used in hardware and protocol design. This research aims to extend the application of formal methods, model checking techniques in particular, to automatic analysis and verification of dependable systems in general. More efforts will be devoted to methodologies of model checking infinite state spaces and its applications to program analysis and domain-specific problems.

2 Approaches

Our work has the following specific focus:

- Interprocedural program analyses will be re-examined based on model checking approach, for possibly more automation, efficiency and effectiveness.

- More sophisticated model checker tailored for infinite state spaces based on SP Term. The empirical observation says that most of control flow graphs (CFG) in practical programs have bounded tree width. SP Term is an algebraic construction of a control flow graph that has bounded tree width, which enables control flow analysis to be carried out in an iterative-free manner. Model checking methodologies on SP Term and an extension to current model checkers tailored for infinite state spaces are expected.

- Verification of security properties for software or protocols based on pushdown model checking. Process calculi as spi-calculus and our previous proposal for Grid Computing, so-called Proxy Certificate Trust List, will be examined as case studies.
3 Work in the Academic Year of 2005

- Interprocedural Program Analysis Based on WPDS
  Based on the observation that “program analysis is abstraction plus model checking”, we investigate pushdown model checking based approach on interprocedural program analyses for mono-thread Java. The running example is an interprocedural dead code detection under PER (partial equivalence relation) based abstraction. The prototype implementation combines SOOT as preprocessing to convert Java to Jimple and the Weighted PDS (pushdown system) library as the back-end model checking engine. With these existing tools, we developed an interprocedural dead code analyzer for mono-thread Java. This analysis framework enables us a rapid prototyping for an interprocedural analysis design.

- Strictness Analysis for Haskell
  Previous approaches to strictness analysis are based on abstract interpretation and iteration of finding fixed points. In our approach, the easy implementation with ready-made fixed-point calculator, that is pushdown model checking, enables us rapid prototyping. The final aim is possibly easier implementation with reasonable efficiency for strictness analysis. We also made an effort to come up with solutions for Higher-order functions and non-flat domain.

4 Future Work

- More case studies of interprocedural program analyses; the first target will be call graph generation.

- Applying similar methodology to other languages, such as ML. We are planning to cooperate with Interoperable ML project \(^1\).

- More sophisticated model checker tailored for infinite state spaces based on SP Term.

A Publications

- Interprocedural Program Analysis for Java based on Weighted Pushdown Model Checking, submitted to AVIS2006.

B System Development

- Interprocedural Program Analyzer for mono-thread Java based on Weighted Pushdown Model Checking

\(^1\)http://www.pllab.riecl.tohoku.ac.jp/~ohori/research/iml-e.html