Program Analysis based on Weighted Pushdown Model Checking

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

1 Research Aims

The increasing complexity of software and hardware systems nowadays makes their validation more challenging. Current practiced methods for system validation are mostly based on simulation and testing. The fundamental problem for these methods is that they cannot cover all possible scenarios of system runs. A promising alternative to system validation is formal verification, of which popular approaches are theorem proving, model checking, etc. Theorem proving is a deductive approach, and the use of it usually need expertise and enough experience. In contrast, model checking, the so-called “push-button technique”, is a fully automatic algorithmic technique for verification on temporal safety of reactive and concurrent systems. In particular, if model checking once fails, counterexamples are provided as evidences for the failure and clues for fixing the problem. This research aims at applying formal verification methods, model checking in particular, to automatic system validation.

2 Approaches

B. Steffen and D.A. Schmidt observed that data flow analysis can be regarded as model checking of abstract interpretation. This view enables separation of the design (abstraction) and implementation (back-end model checking) of program analyses. Popular model checkers, such as SPIN, NuSMV/SMV, are model checkers on finite state space. Recently, practical algorithms for (weighted) pushdown model checking have been developed, which enables design of interprocedural and context-sensitive program analysis. For prototype implementations, an universal analysis engine is advantageous to clarify whether analysis design is correct wrt language semantics based on abstract interpretation.

Our start is an interprocedural extension of BANDERA-like approach, i.e. design and prototype implementation with a model checker as the analysis engine. Program analysis for Java is explored based on weighted pushdown model checking, following to the methodology that program analysis = abstract interpretation + model checking. In Java, call graph generation and points-to analysis are mutually dependent. Since they are the starting point and basis of most other program analyses, we choose points-to analysis as our first target. Our idea can be applied to other object-oriented programming languages as well.
3 Progress

We propose context-sensitive points-to analysis algorithms for Java based on weighted pushdown model checking. These four algorithms are designed by taking into account primary design choices, which are orthogonally two dimensional: on-the-fly vs ahead-of-time call graph construction, exploded supergraph vs control flow graph model design. Furthermore, the relatively unexplored problem of parametrization is explored. Parameterized flow-sensitivity is obtained in our algorithms based on either model reduction or simplifying the weight design.

The above design choices are formalized as either weight designs or model constructions in the framework of weighted pushdown model checking. Most of combinations, expect for the one with ahead-of-time call graph construction on the exploded supergraph based model, are also supported with implementations. A prototype framework is presented, which combines SOOT compiler as preprocessing to convert Java to Jimple and the Weighted PDS (pushdown system) library as the back-end model checking engine. Our study also shows that:

- Only ahead-of-time call graph construction with a control flow graph based model design can perform a sound analysis with one run model checking. The study of precision/efficiency trade-off on it will be our future work.
- An ahead-of-time construction needs to explicitly give a bound on tracing nesting of field access (we choose 1), whereas an on-the-fly construction automatically bounds it (up to the number of abstract heap locations).

4 Future Work

This is just a beginning. Our future work will be:

- For field insensitive points-to analyses based on a control flow graph, it is an interesting theoretical question whether the ahead-of-time points-to analysis is equally precise as on-the-fly points-to analysis.
- Weighted PDS library calls a user-defined semiring module on each semiring operations. The efficiency of current implementations can be easily improved if more efficient data structures, such as BDD, are applied.
- Further research is to cover concurrent behaviors, since pushdown model checking with more than two stacks is undecidable.

A Publications

Li Xin, Mizuho Ogawa. Catalog of Context-sensitive Points-to Analyses for Java, pending submission to the 21st European Conference on Object-Oriented Programming 2007 (ECOOP’07), Berlin.

B System Development

Prototype implementation of points-to analysis for Java based on SOOT compiler and weighted PDS library