Bachelor Thesis: Speculated Loop Invariants for Proofs

TECHNISCHE UNIVERSITÄT DARMSTADT

Software Engineering Group – Jakob Laenge
laenge@cs.tu-darmstadt.de

Context

The KeY theorem prover can show the correctness of a given program with respect to its specification. This is accomplished using a combination of symbolic execution and deductive verification. KeY extends a sequent calculus for first-order logic to reason about formulas containing programs. While the effects of assignments and conditional statements can be captured precisely without user interaction, to reason about unbounded loops KeY requires specifications in the form of so-called *loop invariants*.

A loop invariant is a formula that allows KeY to reduce the correctness proof of a program containing a loop into three simpler tasks: (i) show that the loop invariant holds when the loop is visited for the first time, (ii) show that the loop invariant is preserved when the loop's body is executed, and (iii) show that the loop invariant is strong enough to prove the correctness of the remaining program. Not every formula is a suitable loop invariant, and hence using an unsuitable formula may prevent KeY from showing the correctness of the program.

Currently KeY requires the user to annotate every unbounded loop with a suitable loop invariant. This imposes a great burden on the user and hinders automated verification. While there exist a vast number of different techniques for automated invariant synthesis, not all of them are guaranteed to produce suitable invariants. Recently a very promising approach was developed that employs dynamic program analysis to discover polynomial invariants. It proceeds by running the program over a test suite and measuring the values of the program variables at the loop's head during different executions. These data samples can be interpreted as points in a coordinate system and a loop invariant can be interpreted as a polynomial that runs through all of these points. Thus the task of invariant generation can be reduced to standard mathematical problems from linear algebra.

While the approach sketched above is very efficient, it suffers from a major drawback: the computed invariants are merely *speculated invariants*, i.e. the algorithm provides no guarantee that the inferred formulas are suitable. Therefore, plugging this invariant generator directly into KeY may not allow KeY to prove correctness of the program.

Thesis

The goal of this thesis is: integrating an invariant generation algorithm which produces speculated invariants into a theorem prover that requires suitable invariants. To this end the invariant generation algorithm outlined above is to be implemented and integrated into KeY. The algorithm should be implemented in Java, using external libraries for equality solving. The integration into KeY should follow an *on-demand* style. This means that loop invariants are only computed as required by the theorem prover. To accomplish this, the original algorithm has to be slightly adapted to work on sequents instead of programs and specifications.

Additionally, the original algorithm should be extended such that it may provide a counterexample if the input sequent is unsatisfiable. Finally, a simple backtracking algorithms has to be devised that allows the prover to recover from an unsuitable invariant.

Approximate Work Distribution

Analysis
Programming
Literature

Contact

Jakob Laenge
Software Engineering Group
laenge@cs.tu-darmstadt.de
Office: S2|02/A202, Phone +49 6151 16-21366