Review 18

3/29/12

CS6V81.502

Seungtack Baek

**Quickly Detecting Relevant Program Invariants**

In this Paper, Ernst et al. presents how to detect “relevant” program invariants using a tool Daikon. They had introduced Daikon in their previous study, yet they envisioned some room for improvement of the relevancy of the output: inclusion of more interesting invariants and exclusion of uninteresting invariants. They improved Daikon using following four strategies:

1. Polymorphism elimination: By tracking polymorphic variables at runtime, they were able to find more interesting invariants from a program
2. Redundant invariants: They pruned implied invariants to suppress uninteresting invariants
3. Comparability: This enabled Daikon to consider fewer variables at a time. Using “coerced types” or “Lackwit types” decreased the number of possible pairs (that can have invariants) between variables.
4. Repeated values: They applied statistical confidence to eliminate invariants that only occur by chance.

The value of this research is in that without running complex & huge static analysis, a programmer can just run this dynamic procedure to find invariants hidden in the program. However, unlike other tools that report possible bugs, this tool does not explicitly tell the programmer what they should look at. Instead, it provides some useful information (I am not 100% sure on how useful of information can invariants provide) about the software.

I was actually surprised about the fact that I could understand quite a portion of this paper, without actually reading the first one (I guess the one that introduced & evaluated Daikon). Yet, I am still uncertain about some contents in this paper (and hoping those to be cleared tonight).

**Question:**

1. As I mentioned above, I am not sure how would programmer use “invariants”… What goods do they provide?