Technische Universität München
Institut für Informatik
Prof. Tobias Nipkow, Ph.D.
Andrei Popescu, Peter Lammich

WS 2012/13
18. 12. 2012

Semantics of Programming Languages

Exercise Sheet 10

Exercise 10.1 Forward Assignment Rule

Think up and prove a forward assignment rule, i.e., a rule of the form $\vdash \{P\}\ x::=a\ \{\dots\}$, where $\dots$ is some suitable postcondition. Hint: To prove this rule, use the completeness property, and prove the rule semantically.

Redo the proofs for *MAX* and *MUL* from the previous exercise sheet, this time using your forward assignment rule.

Exercise 10.2 More Hoare-Rules

Which of the following Hoare-rules are correct? Proof or counterexample!

lemma
"$[\![\vdash \{P\}\ c\ \{Q\}; \vdash \{P'\}\ c\ \{Q'\}]\!] \Longrightarrow \vdash \{\lambda s.\ P\ s \vee P'\ s\}\ c\ \{\lambda s.\ Q\ s \vee Q'\ s\}$"
lemma
"$[\![\vdash \{P\}\ c\ \{Q\}; \vdash \{P'\}\ c\ \{Q'\}]\!] \Longrightarrow \vdash \{\lambda s.\ P\ s \wedge P'\ s\}\ c\ \{\lambda s.\ Q\ s \wedge Q'\ s\}$"
lemma
"$[\![\vdash \{P\}\ c\ \{Q\}; \vdash \{P'\}\ c\ \{Q'\}]\!] \Longrightarrow \vdash \{\lambda s.\ P\ s \longrightarrow P'\ s\}\ c\ \{\lambda s.\ Q\ s \longrightarrow Q'\ s\}$"

Homework 10 Be Original!

Submission until Tuesday, 8 January 2012, 10:00am.

Think up a nice formalization yourself!

Here are some ideas:

- Add some new language features to IMP, and redo some proofs (e.g., compiler,typing,Hoare-Logic).
- A control flow graph (CFG) is a graph where edges are labeled by either an assignment or a boolean expression. An assignment causes a state change and a boolean expression restricts which states can traverse this edge.

  Formalize an operational semantics of control flow graphs and prove some nice results, e.g., compiler (IMP$\rightarrow$CFG or CFG$\rightarrow$STACK), Floyd-style correctness proofs.
- Compile commands to a register machine, and show correctness.
- Prove correct some non-trivial program, e.g., square roots using the bisection method. Hint: A modular approach of writing and proving programs may help, e.g., you may try to reuse a program for multiplication and its correctness proof, rather then inlining the program and the proof.

The following ideas require some amount of non-lecture related knowledge:

- Prove some interesting result about automata/formal language theory
- Implement (in Isabelle/ML) a translation from our WHILE-language to Java or C programs.
- Formalize β-reduction for (untyped) lambda calculus or a simple functional programming language.
- Formalize some results from mathematics

You should yourself set a time limit before starting your project. Also incomplete/unfinished formalizations are welcome and will be graded!