Technische Universität München
Institut für Informatik
Prof. Tobias Nipkow, Ph.D.
Sascha Böhme, Alexander Krauss

WS 2010/11
22. 11. 2010

Semantics of Programming Languages

Exercise Sheet 5

This exercise builds on theory `Big_Step`.
To save some typing, download the theory `Ex05_Template` and fill in the gaps.

Exercise 5.1 Program equivalence

Prove or disprove (by giving counterexamples) the following program equivalences.

(a) *IF And b1 b2 THEN c1 ELSE c2* $\sim$ *IF b1 THEN IF b2 THEN c1 ELSE c2 ELSE c2*

(b) *WHILE And b1 b2 DO c* $\sim$ *WHILE b1 DO WHILE b2 DO c*

(c) *WHILE And b1 b2 DO c* $\sim$ *WHILE b1 DO c*; *WHILE And b1 b2 DO c*

(d) *WHILE Or b1 b2 DO c* $\sim$ *WHILE Or b1 b2 DO c*; *WHILE b1 DO c*

Hint: Use the following definition for *Or*:

definition *Or* :: *"bexp ⇒ bexp ⇒ bexp"* **where**
"Or b1 b2 = Not (And (Not b1) (Not b2))"

Exercise 5.2 Alternative loop constructs

Many programming languages provide different loop constructs beyond our while loops, e.g., do-while and repeat-until.

(a) Prove the following congruence rules, which make it easier to prove program equivalences later:

lemma *Semi_cong*: "$[\![\, c1 \sim c1'; c2 \sim c2' \,]\!] \Longrightarrow c1;\, c2 \sim c1';\, c2'$"

lemma *If_cong*: "$[\![\bigwedge s.\ bval\ b\ s = bval\ b'\ s;\ c1 \sim c1';\ c2 \sim c2' \,]\!]$
$\Longrightarrow$ *IF b THEN c1 ELSE c2* $\sim$ *IF b′ THEN c1′ ELSE c2′*"

lemma *While_cong*: "$[\![\, \bigwedge s.\ bval\ b\ s = bval\ b'\ s;\ c \sim c' \,]\!]$
$\Longrightarrow$ *WHILE b DO c* $\sim$ *WHILE b′ DO c′*"

(b) Define a new loop construct *DO c WHILE b* in terms of the language *com*, where the condition is first checked after execution of the loop body. Define a translation which rewrites a *com* program into a program without while loops. Prove that your translation preserves equivalence.

Homework 5 Occurrence of variables and *REPEAT _ UNTIL _*

Submission until Wednesday, December 1, 2010, 12:00 (noon).

(a) Define a predicate *assigned* which tests if a program contains an assignment command for a given variable.

(b) Prove that if a program contains no assignment for a variable, then that variable is never modified by the program.

(c) Give a counterexample for the reverse implication.

(d) As in Exercise 5.2, define a new loop construct *REPEAT c UNTIL b*, where the loop is executed until *b* becomes true, but at least once. Define a program translation that expresses all WHILE loops in terms of REPEAT. Prove that it preserves equivalence.