Technische Universität München
Institut für Informatik
Prof. Tobias Nipkow, Ph.D.
Peter Lammich, Johannes Hölzl

WS 2013/14
4. 2. 2014

Semantics of Programming Languages

Exercise Sheet 14

1 Nondeterministic Choice

To the standard *com* datatype, add a command for nondeterministic choice:

datatype *com* = ... | *Or com com*

Augment the inductive definition of big-step semantics by adding two new rules:

$$\frac{(c_1,\ s) \Rightarrow s'}{(Or\ c_1\ c_2,\ s) \Rightarrow s'}\ \text{OR-LEFT} \qquad \frac{(c_2,\ s) \Rightarrow s'}{(Or\ c_1\ c_2,\ s) \Rightarrow s'}\ \text{OR-RIGHT}$$

Also extend the inductive definition of the Hoare calculus with a new rule:

$$\frac{\vdash \{P\}\ c_1\ \{Q\} \qquad \vdash \{P\}\ c_2\ \{Q\}}{\vdash \{P\}\ Or\ c_1\ c_2\ \{Q\}}\ \text{OR}$$

Show that the Hoare calculus remains sound, i.e., that we still have

$\vdash \{P\}\ c\ \{Q\} \Longrightarrow \models \{P\}\ c\ \{Q\}$

Note: You need not reproduce parts of the proof that remain unchanged w.r.t. the proof for the original while-language without *Or*.

2 Recursive Functions and Structural Induction

Consider this simple datatype of boolean formulae:

datatype *form* = *Var vname* | *Impl form form*

Write a recursive definition of a function *subst* that does *simultaneous substitution*. The application *subst* σ t simultaneously replaces *every* variable in t with a new sub-formula: Each occurrence of *Var* x is replaced with $\sigma(x)$.

$subst :: (vname \Rightarrow form) \Rightarrow form \Rightarrow form$

If we do simultaneous substitution with $(\lambda x.\ Var\ x)$, we should get the identity function on formulae:

$subst\ (\lambda x.\ Var\ x)\ t = t$

Prove this fact by structural induction on t.

3 Simplified Sign-Analysis

Design a simplified sign analysis, that only distinguishes between positive values and any values, i.e., the lattice has the elements $L = \{Pos,\ Any\}$.

Define the ordering ($\leq$), supremum $\sqcup$, and indicate the $\top$-element. Prove that your definitions yield a join semilattice (class *semilattice_sup_top* from the lecture), i.e., that $\leq$ is a preorder (reflexive, transitive) with greatest element $\top$, and that $\sqcup$ is the least upper bound.

Define the concretization function $\gamma_s :: L \Rightarrow int\ set$, and the abstract operations $num_s :: int \Rightarrow L$ and $plus_s :: L \Rightarrow L \Rightarrow L$. Show that they are sound abstractions, i.e.,

$n \in \gamma_s\ (num_s\ n)$

$n_1 \in \gamma_s\ a_1 \wedge n_2 \in \gamma_s\ a_2 \Longrightarrow n_1{+}n_2 \in \gamma_s\ (plus_s\ a_1\ a_2)$

Iterate the *step'* function for the following program until a fixed point is reached, and tabulate the annotations after each iteration:

```
x := 1 {A1};
y := 2 {A2};
IF z<1 THEN (
  {A3} x := x + y {A4}
) ELSE (
  {A5} x := x + (-1) {A6}
) {A7}
```

	0	1	2	3	4	5	6
A1	*None*						
A2	*None*						
A3	*None*						
A4	*None*						
A5	*None*						
A6	*None*						
A7	*None*						

Notes:

- The column numbered n shall contain the annotation after applying *step'* $\top$ n times. Hence, column 0 that we filled for you contains the initial annotation.
- Remember that the annotations produced by the *step'*-function have type $(vname \Rightarrow L)\ option$.

  Use some shortcut notations to represent annotations, e.g.,
  $\langle a_x, a_y, a_z \rangle$ for $Some\ \langle\ x := a_x,\ y := a_y,\ z := a_z\ \rangle$,
  where $a_x, a_y, a_z \in L$.
- Use the simplest version of *step'*, i.e., the one without analysis of boolean expressions.

4 Post-fixed points

Recall that a complete lattice is a type $'a$ with a partial order $\leq$ such that every set X $::$ $'a\ set$ has a greatest lower bound, denoted $\bigsqcap X$. This means that $\forall\ x \in X.\ \bigsqcap X \leq x$ and $\forall\ y.\ ((\forall\ x \in X.\ y \leq x) \longrightarrow y \leq \bigsqcap X)$.

Prove that, for a complete lattice and a monotone function f $::$ $'a \Rightarrow 'a$ on it, the set of post-fixed points of f is closed under $\bigsqcap$:

$$\forall\ X :: {'a}\ set.\ ((\forall\ x \in X.\ f\, x \leq x) \longrightarrow f\ (\bigsqcap X) \leq \bigsqcap X)$$