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:

\[
\frac{(c_1, s) \Rightarrow s'}{(Or \ c_1 \ c_2, s) \Rightarrow s'} \quad \text{OR-LEFT}
\]

\[
\frac{(c_2, s) \Rightarrow s'}{(Or \ c_1 \ c_2, s) \Rightarrow s'} \quad \text{OR-RIGHT}
\]

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

\[
\vdash \{P\} \; c_1 \{Q\} \quad \vdash \{P\} \; c_2 \{Q\} \quad \vdash \{P\} \; Or \; c_1 \; c_2 \{Q\} \quad \text{OR}
\]

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

\[
\vdash \{P\} \; c \{Q\} \implies \vdash \{P\} \; c \{Q\}
\]

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 \(\sigma\) \(t\) simultaneously replaces every variable in \(t\) with a new sub-formula: Each occurrence of \(Var\) \(x\) is replaced with \(\sigma(x)\).

\[
\text{subst} :: (vname \Rightarrow form) \Rightarrow form \Rightarrow form
\]

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

\[
\text{subst} (\lambda x. \ Var\ x) \; t = t
\]

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 \( L = \{ \text{Pos}, \text{Any} \} \).

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 \( \gamma_s :: L \Rightarrow \text{int set} \), and the abstract operations \( \text{num}_s :: \text{int} \Rightarrow L \) and \( \text{plus}_s :: L \Rightarrow L \Rightarrow L \). Show that they are sound abstractions, i.e.,

\[
n \in \gamma_s (\text{num}_s n) \\
n_1 \in \gamma_s a_1 \land n_2 \in \gamma_s a_2 \implies n_1 + n_2 \in \gamma_s (\text{plus}_s a_1 a_2)
\]

Iterate the \( \text{step}' \) function for the following program until a fixed point is reached, and tabulate the annotations after each iteration:

\[
x := 1 \ \{A1\} \\
y := 2 \ \{A2\} \\
\text{IF} \ z<1 \ \text{THEN} \ (\\n\quad \{A3\} \ x := x + y \ \{A4\} \\
\quad ) \ \text{ELSE} \ (\\n\quad \{A5\} \ x := x + (-1) \ \{A6\}\\n\quad ) \ \{A7\}
\]

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>None</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A2</td>
<td>None</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A3</td>
<td>None</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A4</td>
<td>None</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A5</td>
<td>None</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A6</td>
<td>None</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A7</td>
<td>None</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Notes:
- The column numbered \( n \) shall contain the annotation after applying \( \text{step}' \top \) \( n \) times. Hence, column 0 that we filled for you contains the initial annotation.
- Remember that the annotations produced by the \( \text{step}' \) function have type (\( \text{vname} \Rightarrow L \)) \( \text{option} \).
  Use some shortcut notations to represent annotations, e.g.,
  \( \langle a_x, a_y, a_z \rangle \) for \( \text{Some} \ \{ x := a_x, y := a_y, z := a_z \} \),
  where \( a_x, a_y, a_z \in L \).
- Use the simplest version of \( \text{step}' \), 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 ≤ 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 $\forall \ y. \ ((\forall \ x \in X. \ y \leq x) \rightarrow y \leq \bigsqcap X)$.

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 $\bigsqcap$:

$$\forall \ X :: 'a set. \ ((\forall \ x \in X. \ f x \leq x) \rightarrow f \ (\bigsqcap X) \leq \bigsqcap X)$$