Advanced Programming Languages — Assignment 4

CS 6110 S16 — Due Friday, 15 April 2016

Instructions: *You may complete this assignment with one other CS 6110 student. You must register with your partner on CMS before you begin working together on the assignment. With the exception of your CMS-registered partner you should not give or receive assistance on this assignment. If slip days are used, they must be used by both partners. In addition, please limit use of outside resources to the lecture notes and textbooks for this course. If you have any questions about what is allowed and what is not allowed, please contact the course staff.*

1. Hoare Logic

Suppose we extend IMP with arrays. We can do this by adding arithmetic expressions:

- $\mathsf{X}[a]$ to retrieve the integer stored in array X at the index denoted by a, and
- $\mathsf{len}(\mathsf{X})$ to obtain the length of the array X,

as well as a new command $\mathsf{X}[a_0] = a_1$ to update the array X.

If X is an array, we use the following "function update" notation, $\mathsf{X}[i \mapsto j]$, to indicate the array obtained by updating X so that $\mathsf{X}[i]= j$, provided i is in bounds for the array X. Arrays are zero-indexed, so legal array indices for X are in the range $0 \leq i < \mathsf{len}(\mathsf{X})$.

We can now attempt to extend the axiomatic semantics for IMP by adding a new Hoare Logic axiom to handle the array update command:

$$\frac{}{\{P[\mathsf{X}[a_0 \mapsto a_1]/\mathsf{X}]\}\ \mathsf{X}[a_0] := a_1\ \{P\}}$$

Unfortunately, this is not enough. As hinted at above, in order to prove the correctness of an assignment to an array, we must make sure that the array index is in bounds. Hence, we need to extend our proof rules to check that the "domain" of arrays is respected. In this exercise you extend the axiomatic semantics to account for these details.

The syntax for the extended version of IMP with arrays is as follows:

$$\begin{array}{rrcl}
\mathit{AExp}: & a & ::= & \overline{n} \mid x \mid a_0 \oplus a_1 \mid \mathsf{X}[a] \mid \mathsf{len}(\mathsf{X}) \\
\mathit{BExp}: & b & ::= & \mathsf{true} \mid \mathsf{false} \mid a_0 \odot a_1 \mid b_0 \oslash b_1 \mid \neg b \\
\mathit{Com}: & c & ::= & \mathsf{skip} \mid x := a \mid \mathsf{X}[a_1] := a_2 \mid c_0 \,;\, c_1 \mid \mathsf{if}\ b\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2 \mid \mathsf{while}\ b\ \mathsf{do}\ c \\
& \oplus & ::= & + \mid * \mid - \\
& \odot & ::= & \leq \mid = \\
& \oslash & ::= & \vee \mid \wedge
\end{array}$$

(a) Write inductively defined predicates $\mathsf{dom}_a(-)$ and $\mathsf{dom}_b(-)$ for arithmetic expressions and boolean expressions, respectively. The idea is that that for any arithemetic expression a_0, we have that $\models \mathsf{dom}_a(a_0)$ if and only if every array index operation in a_0 obeys the proper domain restrictions. Similarly, $\mathsf{dom}_b(-)$ asserts that boolean expressions don't have domain violations.

(b) Modify the Hoare Logic rules for to use the domain predicates from part (a). (You only need to give the rules that need to be changed.)

(c) The following program in the extended version of IMP performs an insertion sort on the array X.

```
j = 1;
while j < len(X) do
    x = X[j];
```

```
$i = j - 1$;
while $(i \geq 0) \wedge (X[i] < x)$ do
   $X[i+1] = X[i]$;
   $i = i - 1$
$X[i+1] = x$
$j = j + 1$
```

The desired post-condition, P, of the above loop specifies that X is sorted in descending order:

$$P = \{\forall i. 0 \leq i < (\mathsf{len}(\mathsf{X}) - 1) \Rightarrow \mathsf{X}[i] \geq \mathsf{X}[i+1]\}$$

i. Find suitable loop invariants for the two while loops in the above program.

ii. If A_0 is the loop invariant for the outer loop, prove that $A_0 \wedge \neg(j < \mathsf{len}(\mathsf{X})) \Rightarrow P$. That is, if the outer loop terminates with loop invariant A_0, then the post-condition holds.

iii. Let A_1 be the loop invariant for the inner while loop and let c be the body $\mathsf{X}[i+1] = \mathsf{X}[i]$; $i = i-1$. Prove that that A_1 is actually an invariant, that is the following judgement is derivable using the extended Hoare logic rules:

$$\{A_1 \wedge ((i \geq 0) \wedge (\mathsf{X}[i] < x))\}\ c\ \{A_1\}$$

2. Domain Constructions

We saw the disjoint sum domain construction in class. It's reasonable to wonder why we had to bother with tagging the elements from the two domains being summed. Consider these alternative sum-like constructions and either show that they always yield CPOs or that they don't:

(a) Given CPOs $(D, \sqsubseteq_D)$ and $(E, \sqsubseteq_E)$, show that the domain $(D \cup E, \sqsubseteq_D \cup \sqsubseteq_E)$ is or is not necessarily a CPO. (Recall that the relation $\sqsubseteq_D$ can be viewed as a subset of $D \times D$, so we can take a union of two relations).

(b) Given pointed CPOs $(D, \sqsubseteq_D)$ and $(E, \sqsubseteq_E)$ show that the following domain is or is not necessarily a pointed CPO:

$$(\{in_1(d) \mid d \in D \wedge d \neq \bot_D\} \cup \{in_2(e) \mid e \in E \wedge e \neq \bot_E\} \cup \{\bot\}, \sqsubseteq)$$

where $x \sqsubseteq y$ iff

$$\begin{array}{l} x = \bot \ \vee \\ (x = in_1(d) \wedge y = in_1(d') \wedge d \sqsubseteq_D d') \ \vee \\ (x = in_2(e) \wedge y = in_2(e') \wedge e \sqsubseteq_E e') \end{array}$$

(This is called a *smash sum*, and written $D \oplus E$.)

3. Continuous Partial Orders

Briefly characterize the continuous functions from an arbitrary pointed CPO to a discrete CPO. (Winskel 8.6)

4. Approximation

An element x of a CPO *approximates* another element y, written $x \ll y$, if all chains z_n whose supremum is at least y contain an element that is at least x:

$$y \sqsubseteq \bigsqcup_n z_n \implies \exists n.\ x \sqsubseteq z_n$$

An element of a CPO is *finite* (or *compact*) if it approximates itself.

(a) Show that $x \ll y \implies x \sqsubseteq y$.

(b) Describe the finite elements of the following domains:
   i. natural numbers $\mathbb{N}$ with discrete ordering
   ii. $\mathbb{N} \cup \{\infty\}$ with $\leq$ ordering $(\forall n \in \mathbb{N}.\ n \leq \infty)$
   iii. $\mathbb{Z} \to \mathbb{Z}$ with pointwise ordering
   iv. $\mathbb{Z} \to \mathbb{Z}_\perp$ with pointwise ordering

5. Debriefing

   (a) How many hours did you spend on this assignment?
   (b) Would you rate it as easy, moderate, or difficult?
   (c) Did everyone in your study group participate?
   (d) How deeply do you feel you understand the material it covers (0%–100%)?
   (e) If you have any other comments, we would like to hear them! Please write them here or send email to `jnfoster@cs.cornell.edu`.