Two hours

UNIVERSITY OF MANCHESTER
SCHOOL OF COMPUTER SCIENCE

Understanding Programming Languages

Wednesday 16th January 2008
Time: 09:45 – 11:45

Please answer any THREE Questions from the FIVE questions provided

This is a CLOSED book examination

The use of electronic calculators is NOT permitted.
All questions on this paper refer to the DO language whose syntax and structural operational semantics are given as follows.

\[
\begin{align*}
\text{a} & := n \mid x \mid a_1 + a_2 \mid a_1 \cdot a_2 \mid a_1 - a_2 \\
\text{b} & := \text{true} \mid \text{false} \mid a_1 = a_2 \mid a_1 \leq a_2 \mid \neg b \mid b_1 \land b_2 \\
\text{S} & := x := a \mid \text{skip} \mid S_1 ; S_2 \mid \text{if } b \text{ then } S_1 \text{ else } S_2 \mid \text{do } S \text{ while } b
\end{align*}
\]

Table 1: Syntax of Do

\[
\begin{align*}
[\text{ass}_{\text{sos}}] & \quad < x := a, s > \Rightarrow s[x \mapsto A[a], s] \\
[\text{skip}_{\text{sos}}] & \quad < \text{skip}, s > \Rightarrow s \\
[\text{comp}^1_{\text{sos}}] & \quad < S_1, s > \Rightarrow < S'_1, s' > \\
[\text{comp}^2_{\text{sos}}] & \quad < S_1, s > \Rightarrow < S'_1, s' > \\
[\text{if}^1_{\text{sos}}] & \quad < \text{if } b \text{ then } S_1 \text{ else } S_2, s > \Rightarrow < S_1, s > \quad \text{if } B[b], s = \text{tt} \\
[\text{if}^2_{\text{sos}}] & \quad < \text{if } b \text{ then } S_1 \text{ else } S_2, s > \Rightarrow < S_2, s > \quad \text{if } B[b], s = \text{ff} \\
[\text{do}_{\text{sos}}] & \quad < \text{do } S \text{ while } b, s > \Rightarrow < S; \text{ if } b \text{ then do } S \text{ while } b \text{ else skip}, s >
\end{align*}
\]

Table 2: Structural Operational Semantics for Do

\[
\begin{align*}
B[\text{true}], s & = \text{tt} \\
B[\text{false}], s & = \text{ff} \\
B[a_1 = a_2], s & = \begin{cases}
\text{tt} & \text{if } A[a_1], s = A[a_2], s \\
\text{ff} & \text{if } A[a_1], s \neq A[a_2], s
\end{cases} \\
B[a_1 \leq a_2], s & = \begin{cases}
\text{tt} & \text{if } A[a_1], s \leq A[a_2], s \\
\text{ff} & \text{if } A[a_1], s > A[a_2], s
\end{cases} \\
B[\neg b], s & = \begin{cases}
\text{tt} & \text{if } B[b], s = \text{ff} \\
\text{ff} & \text{if } B[b], s = \text{tt}
\end{cases} \\
B[b_1 \land b_2], s & = \begin{cases}
\text{tt} & \text{if } B[b_1], s \text{ and } B[b_2], s \\
\text{ff} & \text{if } \text{not } (B[b_1], s \text{ and } B[b_2], s)
\end{cases}
\end{align*}
\]

Table 3: The Semantics of Boolean Expressions

\[
\begin{align*}
A[n], s & = N[n] \\
A[x], s & = s x \\
A[a_1 + a_2], s & = A[a_1], s + A[a_2], s \\
A[a_1 \cdot a_2], s & = A[a_1], s \cdot A[a_2], s \\
A[a_1 - a_2], s & = A[a_1], s - A[a_2], s
\end{align*}
\]

Table 4: The Semantics of Arithmetic Expressions
1. a) Give a natural semantics for the statement part of the Do language. (5 marks)
b) Prove that your natural semantics is equivalent to the structural operational semantics of Do. (10 marks)
c) Can you extend your natural semantics to handle non-deterministic or statements of the form:

\[ S_1 \text{ or } S_2 \]

Provide a semantics or explain why this is not possible. (5 marks)

2. a) Give a structural operational semantics style store semantics for Do. (10 marks)
b) Extend this store semantics to include C-style pointer operations & and *. (10 marks)

3. a) Give an abstract machine and compiler for the statement part of the Do language. (5 marks)
b) Prove that for the statement part of the language your compiler and abstract machine is equivalent to the structural operational semantics of Do. (10 marks)
c) Extend your abstract machine and compiler to handle non-deterministic par statements of the form:

\[ S_1 \text{ par } S_2 \] (5 marks)

4. a) Give a denotational semantics for the statement part of the Do language. (5 marks)
b) Prove that your denotational semantics is equivalent to the structural operational semantics of Do. (10 marks)
c) Extend your denotational semantics to handle statements of the form:

\[ \text{abort} \] (5 marks)
5. a) Give an axiomatic semantics for the statement part of the \texttt{Do} language. (5 marks)

b) Prove that your axiomatic semantics is sound and complete with respect to the structural operational semantics of \texttt{Do}. (10 marks)

c) Certain programs are not intended to terminate (operating systems, micro-controllers).

Can we reason about such programs using an axiomatic semantics? What problems might we encounter? (5 marks)