UNIVERSITY OF EDINBURGH

FACULTY OF SCIENCE AND ENGINEERING

COMPUTER SCIENCE THREE

LANGUAGE SEMANTICS AND IMPLEMENTATION

Thursday 30 May 2002

09:30 to 11:00

Examiners:

Prof. K. Turner (External)

Prof. C. Stirling (Chairman)

INSTRUCTIONS TO CANDIDATES

Answer any TWO questions.

All questions carry equal weight.

1. Axioms and rules for LC evaluation include the following

$$\langle !l, s\rangle \Downarrow \langle n, s\rangle \ \text{ if } s(l) = n \qquad \langle \mathbf{skip}, s\rangle \Downarrow \langle \mathbf{skip}, s\rangle$$

$$\frac{\langle E, s\rangle \Downarrow \langle n, s'\rangle}{\langle l := E, s\rangle \Downarrow \langle \mathbf{skip}, s'[l \mapsto n]\rangle}$$

$$\frac{\langle C_1, s\rangle \Downarrow \langle \mathbf{skip}, s'\rangle \quad \langle C_2, s'\rangle \Downarrow \langle \mathbf{skip}, s''\rangle}{\langle C_1; C_2, s\rangle \Downarrow \langle \mathbf{skip}, s''\rangle}$$

(a) Give the LC evaluation rules for conditional **if** B **then** C_1 **else** C_2 and for while loops **while** B **do** C. *[5 marks]*

(b) Assuming the standard LC rules for integers and for booleans, give full derivations for the LC evaluation relation for the following configurations where state $s = \{l \mapsto 2, l' \mapsto 4\}$ *[8 marks]*

i. $\langle\, \mathbf{if}\ !l > 0\ \mathbf{then}\ l := !l' + 1; l' := !l + 2\ \mathbf{else\ skip}, s\rangle$

ii. $\langle\, \mathbf{while}\ !l > 0\ \mathbf{do}\ l' := !l' * !l; l := !l - 1, s\rangle$

(c) Some languages have mechanisms for multiple exits. Consider the following loop construct that allows for two exits

$$\mathbf{loop}\ C_1; \mathbf{if}\ B_1\ \mathbf{exit}; C_2; \mathbf{if}\ B_2\ \mathbf{exit}; C_3\ \mathbf{repeat}$$

The idea is that C_1 is executed and if B_1 is then true the loop is exited, and otherwise C_2 is executed and if B_2 is then true the loop is exited, and otherwise C_3 is executed and the loop starts over again. *[12 marks]*

i. Give evaluation rules for this new looping construct

ii. Assuming s is the state $\{l \mapsto 2, l' \mapsto 4\}$ give a full derivation for the evaluation relation for the following derivation.

$$\langle \mathbf{loop}\ l' := !l + !l'; \mathbf{if}\ !l > 2\ \mathbf{exit}; l := !l - 1; \mathbf{if}\ !l = 0\ \mathbf{exit}; \mathbf{skip\ repeat}, s\rangle$$

2. The language LLC extends LC with **let** name = P **in** P′. Static semantics for LLC use the types *int*, *bool*, *loc* and *cmd*. Axioms and rules of the static semantics include the following.

$$\Gamma \vdash x : \tau \quad \text{if } \Gamma(x) = \tau$$

$$\frac{\Gamma \vdash P : \mathit{loc} \quad \Gamma \vdash P' : \mathit{int}}{\Gamma \vdash P := P' : \mathit{cmd}}$$

$$\frac{\Gamma \vdash P : \mathit{cmd} \quad \Gamma \vdash P' : \mathit{cmd}}{\Gamma \vdash P; P' : \mathit{cmd}}$$

(a) Give static semantics rules for the dereferencing construct $!P$ and for the let construct **let** $x = P$ **in** P'. [*6 marks*]

(b) A program P without free names is said to have type τ if $\emptyset \vdash P : \tau$ is derivable from the axioms and rules of the static semantics. Assuming the standard static semantics rules for integer expressions, for each of the following either show that it has a type, by giving a full derivation, or show that it cannot have a type. [*10 marks*]

i. **let** $x = l$ **in let** $y = 8 + !x$ **in** $x := y; l' := !x$
ii. **let** $x = l$ **in let** $y = !l'$ **in** $x := y; y := !x$
iii. **let** $x = 12$ **in let** $y = 19 * x$ **in let** $z = x + y$ **in** z

(c) A useful property of the static semantics is subject reduction: if P has type τ and $\langle P, s\rangle \Downarrow \langle V, s'\rangle$, then V has type τ. The proof of this property is by rule induction on $\Downarrow$. [*9 marks*]

i. Explain what rule induction is.
ii. Consider the following simple axioms and rules for a transition relation $\downarrow$ that relates words in $\{0, 1\}^*$.

$$0 \downarrow 0 \qquad 1 \downarrow 1$$

$$\frac{w \downarrow w'}{0w \downarrow w'0} \qquad \frac{w \downarrow w'}{1w \downarrow w'1}$$

Prove by rule induction that for any word w, if $w \downarrow w'$ then w' is the reverse of w (for instance, 1110 is the reverse of 0111).

3. In LFP a term M_1 can be applied to another term M_2, written $M_1 M_2$.

(a) The evaluation semantics of $M_1 M_2$ is defined by the call-by-value and call-by-name rules which determine the conditions that must hold in order to infer $\langle M_1 M_2, s\rangle \Downarrow \langle V, s'\rangle$. Give these rules and explain the intuition behind them. *[6 marks]*

(b) Let $\Downarrow_v$, $\Downarrow_n$ be the LFP evaluation relations in which application is defined using call-by-value and call-by-name. Determine $\langle V_1, s_1\rangle$ and $\langle V_2, s_2\rangle$ such that *[10 marks]*

i. $\langle(\lambda x.(l := x + 1; l := x + 1))!l, \{l \mapsto 0\}\rangle \Downarrow_v \langle V_1, s_1\rangle$

ii. $\langle(\lambda x.(l := x + 1; l := x + 1))!l, \{l \mapsto 0\}\rangle \Downarrow_n \langle V_2, s_2\rangle$

Using the axioms and rules of the evaluation semantics, give derivations showing that your choices are correct.

(c) Let C, E be a command and an integer expression of the LC language, respectively. Since LFP extends LC, we can consider C and E as terms of LFP.

Are the following statements true? Prove carefully or provide a counterexample. *[9 marks]*

i. If $\langle(\lambda x.C)E, s\rangle \Downarrow_v \langle V, s'\rangle$ then $\langle(\lambda x.C)E, s\rangle \Downarrow_n \langle V, s'\rangle$

ii. If $\langle(\lambda x.E)C, s\rangle \Downarrow_v \langle V, s'\rangle$ then $\langle(\lambda x.E)C, s\rangle \Downarrow_n \langle V, s'\rangle$