March 25, 1991

CS 541 — Programming Languages 1

Exam

Please write your name here:

Before you begin, please take a moment to look over the entire test so that you can budget your time.

Answers to essay questions will be judged on their completeness and clarity. However, please do not feel obligated to fill up all the empty space on the page. Conversely, do not feel limited by the space on a page, since you may continue an answer on the back of a page if you wish.

problem	subject	points	score
1	write church-numerals in Scheme	15	
2	Scheme vs. lambda calculus	15	
3	derive the normal form	24	
4	write an operational semantics	36	
5	type derivation	10	
Total		100	

1. (15 points) Write an Scheme procedure such that `(church-numeral n)` returns a Scheme procedure (i.e., a closure) representing that Church numeral for that natural number `n`. For example, `(((church-numeral 0) (lambda (x) (+ x 1))) 7)` should return 7.

   (No, we won't tell you anything more about what a church numeral is; you're supposed to know that. But we won't take off much for small syntax errors.)

2. (15 points) Describe the differences, if any, between function abstraction (`lambda`) in Scheme and function abstraction in the untyped lambda calculus. Give an example of any differences. If there are no differences write "No differences." along with some reasons.

3. (24 points) Reduce each of the following lambda expressions to normal form if possible, or show that the expression has no normal form. You must show each of the reduction steps involved, if any are needed. (Each is worth 4 points)

   (a) $\lambda z \,.\, y \;\lambda y \,.\, (y \;\lambda x \,.\, x \; y)$

   (b) $\lambda f \,.\, (\lambda x \,.\, f \; x) \, (\lambda x \,.\, f \; x)$

   (c) $(\lambda x \,.\, \lambda y \,.\, z \; x \; y) \;\lambda x \,.\, x \; x$

   (d) $(\lambda w \,.\, (\lambda x \,.\, x \; x \; x)) \; f \;\lambda y \,.\, y \; y$

   (e) $\lambda b \,.\, (\lambda c \,.\, (c \; x \; y)) \, (\lambda n \,.\, \lambda e \,.\, \lambda k \,.\, k(f \; n))$

   (f) $(\lambda y \,.\, \lambda t \,.\, (\lambda e \,.\, \lambda c \,.\, y \; e \; (t \; e \; c)))(\lambda e \,.\, \lambda c \,.\, \lambda s \,.\, s)((\lambda x \,.\, x \; x) \, (\lambda x \,.\, x \; x))$

4. (36 points) Give a terminal transition system for the expression languge with erratic nondeterminism defined informally below. You should include a description of the set of configurations (Γ), the set of final configurations (T), the axiom schemes and inference rules that define the transition relation ($\Rightarrow$), and both the input ($\mathcal{I}$) and output ($\mathcal{O}$) functions.

   The language's grammar and informal definition are as follows.

   $$\begin{array}{rl} \langle expr \rangle ::= & \bot \\ | & \texttt{true} \mid \texttt{false} \\ | & \texttt{if}\ \langle expr \rangle\ \texttt{then}\ \langle expr \rangle\ \texttt{else}\ \langle expr \rangle\ \texttt{fi} \\ | & \texttt{(constant}\ \langle expr \rangle\texttt{)} \\ | & \texttt{(}\langle expr \rangle\ \Box\ \langle expr \rangle\texttt{)} \end{array}$$

   The meaning of $\bot$ is nontermination (an infinite loop). The meaning of `true`, `false`, and `if` are as usual. The meaning of `(constant` e_1`)` is the meaning of e_1. The meaning of an erratic choice ($e_1 \Box e_2$) is either the meaning of e_1 or the meaning of e_2, with the choice being nondeterministic.

5. (10 points) Derive the type, if any, of the following pre-term of the simply typed lambda calculus in the empty environment. If the pre-term does not have a type, write "no type".

   $\lambda g : \mathbf{o} \rightarrow \mathbf{o} \,.\, \lambda y : \mathbf{o} \,.\, g(g(y))$