Com Sci 221/321 — Programming Languages
Final Exam, Autumn 1999

Wednesday, 8 December, 10:30 AM - 12:30 PM

Ryerson 251

5 problems on pages 2–6 plus 1 graduate problem on page 7

- You may use the textbook, my class materials, any other published material, and notes of your own.
- There are 5 problems for Com Sci 221 and 321. There is 1 additional problem for Com Sci 321 only. Each numbered problem is weighted equally for grading purposes.
- You may use whatever sort of paper you find convenient, but please make your answers legible, and write your name on every sheet.

Com Sci 222 and 322

Problem 1: l- and r-values

Consider the assignment

$$A[i] = A[A[i]]$$

Describe carefully the sequence of seven operations on l-values and r-values involved in executing this assignment command. You don't need to give calculations in detail, but you must mention every time something is calculated from something else.

Give an example of initial values for A and i that cause $A[i] \neq A[A[i]]$ immediately after executing the assignment. Explain what happened *briefly* in terms of l- and r-values (don't go through all the steps again, just mention which l- and/or r-values caused the inequality).

Problem 2: reasoning with invariants

Suppose that you are given an array A of integers, indexed from 0 to n ($n \geq 2$), and you are guaranteed that the values in the array increase from $A[0]$ up to some maximum value at a position $A[t]$, then decrease from $A[t]$ to $A[n]$. That is, there exists a t with $0 < t < n$, such that

$$\begin{array}{l} \text{for all } i \text{ with } 1 \leq i < t, \quad A[i] < A[i+1], \text{ and} \\ \text{for all } j \text{ with } t \leq j \leq n, \ A[j] > A[j+1]. \end{array}$$

(Pay close attention to the difference between strict $<$, $>$ and nonstrict $\leq$, $\geq$.)

Write a simple while-loop program to find the value of t, by manipulating the values of three variables, *left*, *middle*, and *right*, using the invariant

$$\mathit{left} \leq \mathit{middle} \leq \mathit{right} \text{ and } A[\mathit{left}] < A[\mathit{middle}] > A[\mathit{right}]$$

It is essential to use the given invariant—there is a very different program using a different invariant. Try to look at the fewest elements of A that you can.

How is the value of t determined when your loop halts?

Problem 3: simulating control structures with higher-order functions in *ML*

Consider a **while** loop with a single **break** that exits the loop:

> **while** C_1 **do begin**
> SL_1
>   **if** C_2 **then break**
> SL_2
> **end**

Define a function *while_break* in pure functional *ML* to capture the behavior of the control structure above. The function needs 4 arguments, corresponding to C_1, C_2, SL_1, SL_2, and another, corresponding to the state.

Problem 4: conditional vs. pattern-matching in *ML*

Here is an attempt to define conditional in *ML*, using pattern-matching instead of a built-in conditional function.

$$
\begin{array}{llllll}
\textbf{fun} & \mathit{cond} & \textbf{true} & x & y & = x \\
| & \mathit{cond} & \textbf{false} & x & y & = y \\
; & & & & &
\end{array}
$$

Explain *very* briefly (one sentence) why the definition of *cond* above is not satisfactory.

What is the difference in results between the two following forms?

$$\mathit{cond}\; c\; (x\; z)\; (y\; z)$$

and

$$\mathit{cond}\; c\; x\; y\; z$$

What sorts of values for x, y, z will make them the same, vs. different?

Problem 5: occurs check in ***Prolog*** unification

Give an example of a small pure logic program in *Prolog* that demonstrates the lack of an occurs check in *Prolog*'s unification procedure. Describe *very* briefly (one sentence) the behavior of this program with vs. without the occurs check.

Com Sci 321 only

Problem G1: circularity in *ML* type inference

Suppose that we implement *ML* without any occurs check in the type inference mechanism, and that we accept a program as correct when we infer a "circular" type for it. For example,

$$\textbf{fun } paradox\ x\ = (x\ x);$$

is a correct program in circular-typed *ML*.

1. Invent a sensible way to denote the type of *paradox* in circular-typed *ML*.

2. What is the type of (*paradox paradox*)?

3. Let a and b be expressions in circular-typed *ML*, where a has a function type, and b has any type. (a (*paradox paradox* b)) is also an expression in circular-typed *ML*. That is, (*paradox paradox*) appears to be a universal type converter. In *Pascal* and *C*, the use of variant records constitutes a loophole in the type system, and destroys the security that objects are only used with their true types. Does the presence of (*paradox paradox*) similarly destroy the security of type checking in circular-typed *ML*? Explain very briefly (1–3 sentences).