Honors Programming Languages  
G22.3110 Fall 2001  

Final Exam

There are 6 questions to answer in 3 hours. Do the easy ones first and do not spend much more than half an hour on any one question.

1. (a) In class, we described concurrency as a partial order of events. Explain.
   (b) Given the Ada program below, draw the partial order of the events (e.g. by using a directed graph).

   
   procedure main is
   task one is
      entry uno;
      entry due;
      end one;

   task two is
      entry tre;
      entry quattro;
      end two;

   task body one is
      EventA;
      EventB:
      two.tre;
      EventC;
      accept uno do
         EventD;
         EventE;
         end uno;
      EventF;
      accept due;
      EventG;
      end one;

   task body two is
      EventH;
      accept tre do
         EventI;
         end tre;
      accept quattro;
      EventJ;
      one.due;
      end two;

begin
   EventK;
   two.quattro;
   EventL;
   one.uno;
   Event M;
end main;
2. (a) Describe how backtracking works in Prolog and give an example (using Prolog code).
   (b) In class, I gave the algorithm for a logic program interpreter. Write the algorithm for a Prolog interpreter. Be sure to account, as precisely as possible, for backtracking. You can ignore the cut, !, operator. **Hint:** Write a recursive interpreter (unlike the one I gave in class).

3. The Algol Copy Rule and β-reduction (including the definition of substitution) in the lambda calculus are quite similar.
   (a) State the Algol Copy Rule.
   (b) State the rule for β-reduction in the lambda calculus, along with the definition of substitution.
   (c) Your answer to part (a) should have included two conditions where variable renaming would be necessary. Describe how these conditions correspond to conditions in the lambda calculus substitution rule, and give an example in the lambda calculus where both kinds of renaming is necessary.

4. (a) In what way is a type derivation (as performed during type checking) like a proof of a theorem in mathematics?
   (b) Give the type derivation that shows that the expression
   ```
   let
   val f = fn x => fn y => if x = y then fn z => z else fn z => z+1
   in
   f true true 3 + f 3 4 5
   end
   ```
   is correctly typed. Be sure to give any type rules and axioms that are used in the derivation.

5. (a) In class, we described the denotational semantics of a language with assignment and pass-by-reference. Suppose the language were instead a pass-by-value language with no other facility for introducing aliasing. Would the notion of a store be necessary to give the denotational semantics of this language? Explain.
   (b) For the language in part (a), i.e. the one with assignment but no aliasing, give the semantics of + and :=. That is, define , the semantic function for expressions, on expressions of the form e1 + e2 and define on statements of the form x := e .
   (c) Suppose the language with pass-by-reference, assignment, and goto, whose semantics we defined in class, was extended to have a while loop. The syntax would be:
   ```
   s ∈ stmts, where s ::= ... | while e do s
   ```
   Extend the semantic function for statements to incorporate the while loop.
   (d) Suppose the language in part (c), above, also had the call/cc operator (as in Scheme). That is, the expression call/cc e captures the current continuation and passes it to the value of e (which should be a function). Extend the semantic function (again, from part (c)) to incorporate call/cc.

6. (a) Write, in Scheme, a function `sum` to compute the sum of the numbers a list, including those numbers contained in nested lists. For example,
(sumnums '(1 (2 3 4) 5) 6))
would return 21. Assume every element of a list is either a number or a list. The scheme predicate (number? x) returns true if the value of x is a number.

(b) Write a similar function in SETL, where instead of lists, the elements are contained in sets (and nested sets). The predicate is_integer(x) returns true if x is an integer. Try to make this function as concise as possible.

(c) In ML, lists are homogenous. Can one still write a single function sumints that could add the integers nested within an int list or an int list list, etc.? That is,

sumints [4,5,6]

would return 15 and

sumints [[1,2],[3,4]]

would return 10. If this is possible, write the definition of sumints. If not, explain why not.

(d) Define your own heterogeneous “list” type in ML. Although it won’t be able to use ML’s list syntax, you should be able to write things like:

fun the_length nil = 0
  |
  | the_length cons(x, xs) = 1 + mylength xs
  |
  | the_length _ = raise list_expected_exception

Your “list” type only has to be heterogenous in the sense that the levels of nesting within a list can vary. The atoms (non-lists) in the list should all be of the same type. Be sure, though, that your “list” definition is flexible enough so that one can use it to create lists who atoms are integers and to create lists who atoms are real numbers, etc.

(e) Once you have defined this type, write the function sumints that takes a “list” whose atoms are integers and returns the sum of those integers. What is the type of sumints?