CS 538

Final Exam

Friday, May 12, 2006
7:45 AM — 9:45 AM
1325 Computer Science

Instructions

Answer any four questions. (If you answer more, only the first four will count.) Point values are as indicated. Please try to make your answers neat and coherent. Remember, if we can't read it, it's wrong. Partial credit will be given, so try to put something down for each question (a blank answer always gets 0 points!).

1. (a) (8 points)
Assume we have a Prolog program with two contradictory rules of the form:

```
a(X) :- b(X,Y), c(Y), fail.
a(X) :- b(X,Y), c(Y).
```

What is the effect of these two rules? Do they cancel each other out? Does one take precedence? Do they cause the evaluator to crash or loop? Justify your answer.

(b) (17 points)
The following Prolog rules define a binary relation `y` (`x` is used as a subroutine). What is the relation that `y` defines?

```
x([A],[]).
x([A|B],[A|T]) :- x(B,T).

y([A],A).
y([A|B],C) :- x(B,T), y(T,C).
```

2. (a) (18 points)
What is the type of the following ML function? How did you infer the type you selected?

```
fun f(g1,g2,[],[]) = []
 |  f(g1,g2,h1::t1,h2::t2)=(g1(h1,g2(h2)))::f(g1,g2,t1,t2);
```

(b) (7 points)
What does the ML function of part (a) compute? What happens if `f`'s third and fourth parameters have different lengths?

3. (a) (13 points)

Complete the following Java class definition. Class `Set` implements a set of objects (of type `Object`). The constructor of no arguments creates an empty set. The constructor with an `Object` array argument creates a set containing the values in the array (with duplicates removed). Function `union` computes a set that is the union of its two sets parameters. Function `intersect` computes a set that is the intersection of its two sets parameters. Function `member` tests whether a given object is a member of a set.

```
class Set {
    Set() { ... }
    Set(Object[] values) { ... }
    static Set union(Set s1, Set s2) { ... }
    static Set intersect(Set s1, Set s2) { ... }
    static boolean member(Object v, Set s) { ... }
}
```

(b) (6 points)

Change your Java solution to part (a) into a Pizza definition that uses parametric polymorphism. That is, rather than containing objects of type `Object`, a set, now referenced as `Set<T>`, contains values of a single fixed type, `T`.

(c) (6 points)

Extend your solution to part (b) to include a member function `map(f,s)`. Parameter `f` is a function that takes a value of type `T` and returns a result of type `T`. Parameter `s` is a set of type `Set<T>`. Function `map` applies `f` to each member of `s`, returning a set of result values (with duplicates removed).

4. (25 points)

A well-known chess problem (often assigned in programming classes) is the "8 queens problem." This problem requires you to place 8 queens on a chessboard so that no queen attacks any other.

In this problem we'll address a simpler version of that problem in which 4 queens must be placed on a 4 by 4 grid so that only one queen appears on each row and on each column, and at most one queen appears on any diagonal. Assume we represent, in Prolog, a 4 by 4 chess board as a list containing four sub-lists. Each sublist contains 4 elements, which can be a `q` (representing a queen) or a `b` (representing a blank position). Thus the grid

q			
		q	
	q		
			q

would be represented as `[[q,b,b,b],[b,b,q,b],[b,q,b,b],[b,b,b,q]]`. (This position is *not* a solution since two `q`'s appear on the same diagonal.)

Write Prolog rules that define the relation `solution(L)`. `L` is a list of lists representing a reduced chessboard as defined above. Given that `L` is ground (already bound to a value), `solution` should succeed if `L` represents a solution to the 4 queens problem. That is, `solution(L)` should answer yes if `L` contains exactly 4 `q`'s, and only one `q` appears in each row and column, and at most one `q` appears on any diagonal.

5. (a) (6 points)

Write an ML datatype definition that defines a polymorphic tree, `'a tree`. A tree is defined to be either a **null** tree, or a **node** containing a single value of type `'a` and a *list* of subtrees of type `'a tree`. Your definition should start

```
datatype 'a tree = (* and you fill in the rest *)
```

(b) 6 points)

Write an ML function `count`, of type `'a tree -> int`, that counts the number of leaves in a tree. A leaf is a node that contains no subtrees.

(c) (6 points)

Write an ML function `map`, of type `'a tree * ('a -> 'b) -> 'b tree`, that walks a tree of type `'a` and applies a function of type `'a -> 'b` to each node. A tree of type `'b` is returned. This output tree has the same structure as the input tree, but with node values updated by application of the function. (Note that if the identity function `fn x => x` is mapped, a copy of the input tree is produced.)

(d) (7 points)

The **airity** of a tree is defined to be the maximum number of non-null children any node has. Thus a binary tree has airity 2 because each node has at most two children.

Write an ML function `airity`, of type `'a tree -> int`, that determines the airity of an input tree. (A null tree has airity 0.)

6. (a) (12 points)

Let `f` and `g` be Python functions and let `L` be a list containing values $[v_1, v_2, \ldots, v_n]$.

Write a Python function `munge(f,g,L)` that computes a list with the following `2*n` values:

$[f(v_1), g(v_n), f(v_2), g(v_{n-1}), \ldots, f(v_n), g(v_1)]$

(b) (13 points)

Let `f` and `g` be Python functions and let `L` be a list containing values $[v_1, v_2, \ldots, v_n]$, where `n` is even. Write a Python function `mangle(f,g,L)` that computes a list with the following `n` values: $[f(v_1), g(v_2), f(f(v_3)), g(g((v_4)), f(f(f(v_5))), g(g(g(v_6)))), \ldots]$