CS 538

Final Exam

Sunday, May 13, 2001
12:25 PM — 2:25 PM
107 Psychology

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(g,[],[]) = []
  | f(g,h1::t1,h2::t2)=((g h1),(g h2))::f(g,t1,t2);
```

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

3. (a) (12 points)
Let `f` and `g` be Python functions and let `L` be a list containing values $[v_1, v_2, ..., v_n]$.
Write a Python function `munge(f,g,L)` that computes a list with the following `2*n` values:
$[f(v_1), g(v_n), f(v_2), g(v_{n-1}), ..., f(v_n), g(v_1)]$

(b) (13 points)
Let `f` and `g` be Python functions and let `L` be a list containing values $[v_1, v_2, ..., v_n]$, where `n` is even. Write a Python function `mangle(f,g,L)` that computes a list with the following `n` values: $[f(v_1), g(v_2), f(f(v_3)), g(g((v_4)), f(f(f(v_5))), g(g(g(v_6)))), ...]$

4. (a) (15 points)
Complete the following Java class definition. Class `Queue` implements a queue of objects (of type `Object`). The constructor of no arguments creates an empty queue. The constructor with an `Object` array argument creates a queue containing the values in the array (with the first array element at the front of the queue). Function `isNull` signals whether the queue is empty. Function `front` returns the first (front) object in the queue. Function `remove` removes the first (front) object from the queue. Both front and remove throw an `EmptyQueue` exception if they are called when the `Queue` object is null. Function `enter` adds an object to the back of the queue.

```
class Queue {
   Queue() { ... }
   Queue(Object[] values) { ... }
   boolean isNull() { ... }
   Object front() throws EmptyQueue { ... }
   void remove() throws EmptyQueue { ... }
   void enter(Object v) { ... }
}
```

(b) (5 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 queue, now referenced as `Queue<T>`, contains values of a single fixed type, `T`.

(c) (5 points)
Extend your solution to part (b) to include a member function `map(f)`. Parameter `f` is a function that takes a value of type `T` and returns a result of type `T`. Function `map` applies `f` to each member of the queue, returning a new queue containing the result values.

5. (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 two different diagonals.)

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.

6. (a) (5 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) (7 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) (13 points)
Write an ML function `mode`, of type `'a tree -> 'a option`, that computes the **mode** of the values in a tree. The mode of a set of values is that value that occurs most often. For example, the mode of the list `[1,3,6,1,7,2]` is `1` since `1` appears twice and all other values appear only once. If the mode value isn't unique, you may return any one of the most frequently occurring values. A null tree contains no values, so you should return `none` as its mode.