CS 538

Final Exam

Thursday, May 17, 2007

2:45 PM— 4:45 PM

1325 Computer Science

Instructions

Answer any *four* questions. (If you answer more, only the first four will count.) Each question is worth 25 points. 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. (25 points)

   Explain how type-inference works in ML. Illustrate the process by solving for the type of the following function

```
fun ugh([]) = []
  |  ugh([]::t) = ugh(t)
  |  ugh([[a]]) = [[a]]
  |  ugh([a]::(s::t)) = ugh(([a]@s) :: t)
  |  ugh((a::(b::t))::r) = (a::(b::t)) :: ugh(r)
```

   What does this function do?

2. (25 points)

   Assume we have a list `L` of integers. Define a Prolog relation `listify(L,M)` that is true if we can divide `L` into one or more sublists, `M`, so that each sublist contains integers in non-decreasing (sorted) order. That is, if `v1` and `v2` are adjacent in `L` and `v1` ≤ `v2` then `v1` and `v2` are adjacent in the same sublist of `M`. However if `v1` > `v2` then `v2` ends one sublist and `v2` begins the next sublist in `M`. For example,

```
| ?- listify([3,5,1,8,9,2,1,0], [[3,5],[1,8,9],[2],[1],[0]]).
 yes

| ?- listify([1,2,3,4,5,6],X).
 X = [[1,2,3,4,5,6]]

| ?- listify([5,4,3,2,1],[5,4,3,2,1]).
 no
```

   Your solution needs to only handle the case where `L` is bound (known).

3. (a) (15 points)

Write Prolog facts and rules that define the relation `append3(L1,L2,L3,L4)`. `append3` represents a 3-way list append. That is, `append3(L1,L2,L3,L4)` is true if lists `L1`, `L2` and `L3` can be appended together to form list `L4`. For example, `append3([1,2],[3],[4,5],[1,2,3,4,5])` and `append3([1],[],[2],[1,2])` are true, while `append3([1],[2],[1],[1,2])` is false.

(b) (5 points)

Explain how Prolog would solve the query `append3([1],[2],[3],L)` using your definition of `append3`.

(c) (5 points)

Would your definition of `append3` work correctly for the query `append3(L1,L2,L3,[1,2,3])`? Why?

4. What do each of the following Python program fragments compute? In each case explain why.

(a) (5 points)

```
L=[1,2,3,4]
for i in [1,2,3]:
   L[-i:]=L[:i]
print L
```

(b) (5 points)

```
def f(a=1,b=2):
   return a+b
print f(f(f()),f(f()))
```

(c) (5 points)

```
print [(i,j) for i in range(1,10) for j in range(0,10,2) if i==j]
```

(d) (5 points)

```
dif=0; L=[1,2,3,4]
while L:
  dif -= L[-1]
  L=L[1:-1]
print dif
```

(e) (5 points)

```
def f(x):
   global a
   a=x*2+a
   return a+1
a=0
print  map ((lambda x: 1+f(x)),[1,2,3])
```

5. (a) (8 points)

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

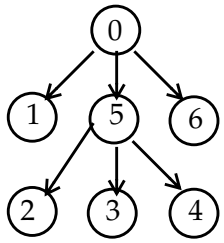
```
datatype 'a triTree = (* and you fill in the rest *)
```

(b) (8 points)

Write an ML function `max`, of type `int triTree -> int Option`, that returns the maximum value in an integer trinary tree. (If the tree is null, `None` is returned).

(c) (9 points)

A **level order** tree traversal first visits the root. Then sons are visited, left to right. Then grandsons are visited, left-to right, etc. For example, in the tree below, first node 0 is visited. Then nodes 1, 5 and 6 are visited. Finally, nodes 2, 3 and 4 are visited



Write an ML function `levelOrder(tree)` of type

```
'a triTree -> 'a list
```

that visits nodes of a `triTree` in level order and returns a list of the values at each node visited. For the `triTree` shown above, `levelOrder` would return `[0,1,5,6,2,3,4]`.

6. (25 points)

Define a Python function `matchList(list,pattern)` where `list` is a list of input strings composed of lower-case characters (`a` to `z`). `pattern` is a string of **pattern characters**. Pattern characters are single lower-case characters, as well as the special pattern characters, `?` and `+` and `*`.

`Pattern` characters are matched against the characters in an input string from left to right. In a pattern, a single letter matches only itself. A `?` symbol matches zero or one input string characters, a `+` in a pattern matches one or more consecutive input string characters, and a `*` in a pattern matches zero or more consecutive input string characters.

Your `matchList` function should match each string in `list` against `pattern`. If the string matches `pattern` (as defined above) it is included in a list of output strings. If a string in list doesn't match `pattern`, it is discarded.

For example, given a `list` defined as

```
["","abc","aaa","bcd","eeeee","bac","bca","a"]
```

and a `pattern` defined as

```
"?a*"
```

the following list would be returned:

```
["abc", "aaa", "bac", "a"]
```