Programming Languages
21 August 2019

Rules

Time at disposal: 4 h.

Read carefully the following questions. For every questions, provide an answer in English. Please remember to justify why the results hold by writing the logical flow that lead you to the answers.

To pass the exam a passing score should be reached in both the Haskell exercises (i.e., exercises 6,7, and 8) and on the remaining questions.

At the end of the exam, remember to submit the assignment both in the PeerGrade system and on Blackboard. The same pdf should be submitted to both systems. The submitted PDF should be named with your special DM552 anonymous ID number received by email (i.e., if the ID is 000DM522 the file should be named 000DM522.pdf). Remember to write the anonymous ID number on the submitted text too and avoid reporting your personal name and surname (the elaborate should be anonymous).

Exercises

1. Please provide an answer to the following questions.

   (a) Closures are often needed in languages that allow functions to be used as parameters. What is a closure? Why are they needed?

   (b) What are *deep binding* and *shallow binding*?

   (c) The following program is written using a pseudo-code. Assume that dynamic scoping is used. What does the following program write if shallow binding is used? Motivate the answer.

```
{ int x = 2;
  int f(int y){
    return x+y;
  }
  int g (int h(int b)){
    int x = 4;
    return h(3) + x;
  }
```

```
  { int x = 6;
    int z = g(f);
    write(z)
  }
}
```

(d) What will the previous code output if instead of shallow binding a deep binding is used?

(e) Can we use a static scoping in the previous code instead of dynamic scope? If so, what binding method is used with static scope: shallow binding or deep binding? What will be the output of the previous code in case static scoping is used?

2. Given $PE_{L_1}^{L_2}$ a partial evaluator of L_2 written in L_1, $I_{L_1}^{L_2}$ an interpreter of L_2 written in L_1, $C_{L_1}^{L_2,L_3}$ a compiler written in L_1 that compiles a language written in L_2 into L_3, $[\![P]\!]$ the function computed by the program P, define a program X written in L_c and a program Y such that the execution of $[\![PE_{L_a}^{L_b}]\!](Y, X)$ produces an interpreter for L_d. X and Y can be compilers, interpreters, or partial evaluators. Motivate your answer.

3. Please provide an answer to the following questions.

(a) What does the following code write, assuming a language with static scoping and call by name.

```
{ int x = 2;
  int y = 5;
  int z = 10;
  void foo(name int v, name int w){
    int x = 1000;
    w = v;
    v = v+w+z;
    z = 1000;
  }
  {  int x = 20;
     int y = 50;
     int z = 100;
     foo(x, y);
     write(x,y,z);
  }
  write(x,y,z) }
```

Motivate the answer.

(b) Can we use call by reference instead of call by name in the previous code? If so, describe what the previous pseudocode will print if instead of call by name the function `foo` uses call by reference. Motivate the answer.

(c) Can we use call by value instead of call by name in the previous code? If so, describe what the previous pseudocode will print if instead of call by name the function `foo` uses call by value. Motivate the answer.

4. Please provide an answer to the following questions.

(a) What is a (data) type? Mention at least two things types are used for?

(b) What is a scalar type and what is a composite type? Mention an example of scalar type and an example of composite type.

(c) What are the 2 rules for type equivalence? Please briefly describe them.

(d) What does it mean that a type system is polymorphic?

(e) How many kinds of polimorphisms exist? Briefly describe them?

5. Please provide an answer to the following questions.

(a) One way to distinguish objects that are still alive (garbage detection) is the *reference counting technique*. Briefly describe this technique.

(b) Mention one pros and one/two cons of reference counting compared to marking and copying?

(c) Consider the following fragment written in a pseudo-language with a reference-counting garbage collector.

```
class C { int n; C next;}
C foo(){
  C p = new C();    // object OGG1
  p.next = new C(); // object OGG2
  C q = new C();    // object OGG3
  q.next = p.next;
  return p.next;
}
C r = foo();
```

State what are the values of the reference counters for the three objects after execution of line `q.next = p.next` and then of line `C r = foo()`. Motivate your answer. Are there any objects that can be returned to the free list? If so, which ones?

6. Consider the following standard definition of `foldl` and `foldr` defined on lists.

```
foldl :: (a -> b -> a) -> a -> [b] -> a
foldl op u []     = u
foldl op u (x:xs) = foldl op (op u x) xs

foldr :: (a -> b -> b) -> b -> [a] -> b
foldr op u []     = u
foldr op u (x:xs) = op x (foldr op u xs)
```

Please provide an answer to the following questions.

(a) By using `foldl` or `foldr` (which one you think is better suited) define the function `maximum` that given a non empty list returns the maximal element of the list. Write the type signature of the function.

(b) By using `foldl` or `foldr` (which one you think is better suited) define the function `reverse` that given a list returns the list in reverse order. Write the type signature of the function.

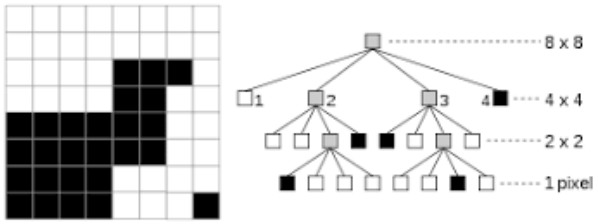

Figure 1: Graphical representation of a Quad Tree.

(c) By using `foldl` or `foldr` (which one you think is better suited) define the function `filter` that given a predicate taking elements of type a and a list ls containing elements of type a, returns the elements of list ls that satisfy the predicate. Write the type signature of the function.

(d) Consider the following expressions.

```
-- expression 1
foldl (&&) True (map (<2) [1..])

-- expression 2
foldr (&&) True (map (<2) [1..])
```

What are the following expressions doing? What value do they return?

7. A lot of techniques for the compression of images are based on a tree data structure called "Quad Tree". Assume that the image is square and the size of the square is a power of 2. If the image is homogeneous (same color) it is encoded, regardless of its dimension, as a leaf containing its color (see Figure 1 for a graphical representation). If the image is not homogeneous, then it is encoded as a node whose child encode i) the upper left square, ii) the upper right square, iii) the bottom right square, and iv) the bottom left square using the data type

```
data QT a = C a | Q (QT a ) (QT a ) (QT a ) (QT a )
```

Write a function in Haskell named `mylimit` that given a color c and a list of QuadTrees ls builds the list of QuadTrees filtering away the QuadTrees in ls that have an image with at least a pixel with a color $< c$.

Write the type signature of all the functions that you define.
Motivate your choices briefly describing your code.

8. Please provide an answer to the following questions.

(a) Consider the following Haskell program

```
getLine :: IO String
putStrLn :: String -> IO ()

main = do
  return "OK"
  line <- getLine
  return ()
  putstrLn line
```

Is this a valid Haskell program? If not, explain why. If it is, give the type signature of main and explain what `main` does when executed.

(b) In the previous program, unfold the do syntactic sugar notation.

(c) Consider the standard function mapM that can be defined on list as follows.

```
mapM :: Monad m => (a -> m b) -> [a] -> m [b]
mapM _ [] = return []
mapM f (x:xs) =  do
  y <- f x
  ys <- mapM f xs
  return (y:ys)
```

The list is an instance of Monad where

```
return x = [x]
xs >>= f = concat (map f xs)
```

where concat is the function that concatenates lists that can be defined as follows.

```
concat [] = []
concat (x:xs) = x ++ (concat xs)
```

Consider the expressions

```
-- expression 1
map (\x -> x) [[1],[2],[3]]

-- expression 2
mapM (\x -> x) [[1],[2],[3]]
```

What is the result of these two expressions? Explain the steps that Haskell takes to reduce both expressions. Moreover, define their type signature.