Exam Grading Rubric

October 24, 2016

1. Programming with higher-order functions. Total of 15 points.

   a. Use of map for iteration: 2 points
      • Correct navigation to boolean to be flipped: 1 point
      • Use of not to flip boolean: 1 points
      • Correct answer returned: 2 points

      Notes:
      • Any missing parts shouldn’t get credit for that part, such as no map
        or no function argument to map or no list to map over.
      • Explicit recursion = 0 points overall
      • Use of while/begin = 0 points overall

   b. Implementation of numTrue
      • Implementation uses filter function: 2 points
      • Implementation correctly filters out false variables: 3 points
      • Implementation uses fold or length function: 2 points
      • Implementation is fully correct: 2 point
      • Fold instead of filter (reimplementing filter) = lost 2 points for filter
      • Not functionally correct and no correct use of length/fold/filter = 0
        points overall
      • Use of if null to handle base case where fold can = lost 0.5 points for
        fold
2. Equational reasoning. Total of 16 points.

- Identify proof by induction on list (2 points).
- Base case is when `xs` is `[]` (1 point)
- Base case uses definition of `foldr` appropriately (1 point)
- Base case uses length-nil law appropriately (1 point)
- Induction case is when `xs` is `(y :: ys)` (1 point)
- Induction case uses definition of `foldr` appropriately (2 points)
- Inductive hypothesis is applied correctly (2 points)
- Induction case uses definition of `plusOne` appropriately (2 points)
- Induction case applies length-cons law appropriately (1 point)
- All the steps in the proof are appropriately connected (1 point)
- All the steps in the proof are correctly identified (2 points)


a. 8 points total, broken down as follows:
   - 4 points for `Var`, `Not`, `And`, and `Or` constructors.
   - 4 points for the correct arguments for each constructor.

b. 5 points total, broken down as follows:
   - 1 point for correct representation of all variables.
   - 1 point for correct representation of negation.
   - 1 point for correct representation of `or`.
   - 1 point for correct representation of `and`.
   - 1 point for correct structure of entire formula.

c. 12 points, broken down as follows:
   - 2 points for correct parameters with correct types. (Types need not be written).
   - 4 points for a case for each `bformula` constructor.
   - 1 point for `Var` case correctly querying the environment.
   - 1 point for `Not` case being correct.
• 2 points for Or case being implemented correctly.
• 2 points for And case being implemented correctly.

d. 5 points, broken down as follows:
• 1 point for correct answer for env1.
• 2 points for correct answer for env2. (1 point for saying it raises an exception)
• 1 point for mentioning short-circuiting or 1 point for saying the lookup function will raise an exception.
• 1 point for correctly concluding that short-circuiting implies the exception won’t be raised.

4. The semantics of \texttt{let}, \texttt{let*} and \texttt{letrec}. Total of 12 points.

(a) \textbf{Result: 1 point}

\textbf{Justification:}

• References let* rule: 1 point
• Explains new variables are added to the environment before evaluating the body of z: 1 point
• Explains new locations are added to the store and initialized before evaluating the body of z: 1 point

(b) \textbf{Result: 1 point}

\textbf{Justification:}

• References letrec rule: 1 point
• Explains new variables are added to the environment before the evaluation of z’s body: 1 point
• Explains new locations are not added to the store until after the evaluation of z’s body and that this causes a reference to an unspecified value: 2 points

(c) \textbf{Result: 1 point}

\textbf{Justification:}

• References let rule: 1 point
• Explains new variables are not added to environment until evaluation of let body: 1 point