Programming Languages: Final Examination

Shin-Cheng Mu

Jan. 9, 2014

You may assume that knowledge about arithmetic (e.g. \(\times\)) is associative, multiplication distributes over addition, etc) need no further proof and may be taken as axioms.

Algebraic proofs or derivations need not be carried out in gory details — you may skip some steps that you think are trivial. Do label the important steps, however, especially the step(s) where you use induction.

You can use all Haskell functions in the standard prelude, mentioned in the lectures.

In a group of problems, the earlier problems are often designed to help you and give you hints to solve the later ones.

1. (40 points) The function \(\text{les}\) (for longest even segment), given a list of numbers, computes the length of its longest segment whose elements are all even numbers:

\[
\text{les} = \text{maximum} \cdot \text{map length} \cdot \text{filter (all even)} \cdot \text{segs},
\]

\[
\text{segs} = \text{concat} \cdot \text{map inits} \cdot \text{tails},
\]

where \(\text{segs}\) computes all segments of a list (definitions of \(\text{inits}\) and \(\text{tails}\) were given in the lecture), and \(\text{all}\ \ p\) checks whether all elements of a list satisfy the predicate \(p\):

\[
\text{all} \quad :: (a \to \text{Bool}) \to [a] \to \text{Bool}
\]

\[
\text{all} \ p \; [] = \text{True}
\]

\[
\text{all} \ p \; (x:xs) = p \; x \; \&\& \; \text{all} \ p \; xs.
\]

The following group of questions guide you to discover an inductive way to compute \(\text{les}\) in linear time.

In addition to known properties of \(\text{map}\), \((+),\) etc, you may need the following properties:

\[
\text{filter} \ p \cdot \text{map} \ f = \text{map} \ f \cdot \text{filter} \ (p \cdot f), \tag{1}
\]

\[
\text{all} \ p \cdot (x :) = \text{if} \ p \; x \ \text{then} \ \text{all} \ p \ \text{else} \ \text{const} \ False, \tag{2}
\]

\[
\text{filter} \ (\text{const} \ False) \; xs = [\], \tag{3}
\]

\[
\text{length} \cdot (x :) = (1+) \cdot \text{length}, \tag{4}
\]

\[
\text{maximum} \cdot \text{map} \ (x+) = (x+) \cdot \text{maximum}, \tag{5}
\]

\[
\text{filter} \ p \; (xs \uplus ys) = \text{filter} \ p \; xs \uplus \text{filter} \ p \; ys, \tag{6}
\]

\[
\text{map} \ f \; (xs \uplus ys) = \text{map} \ f \; xs \uplus \text{map} \ f \; ys, \tag{7}
\]

\[
\text{maximum} \; (xs \uplus ys) = \text{maximum} \; xs \uparrow \text{maximum} \; ys. \tag{8}
\]
(a) (10 points) Come up with an inductive definition of \( \text{segs} \),

\[
\begin{align*}
\text{segs} \; [\;] &= \, ? \\
\text{segs} \; (x : xs) &= \, ?
\end{align*}
\]

where the right-hand side of the non-empty case is allowed to make calls to \( \text{inits} \), \( (++) \), and \( \text{segs} \) itself.

**Solution:** Apparently \( \text{segs} \; [\;] = [[[\;]]] \). For the non-empty case:

\[
\text{segs} \; (x : xs) \\
= \quad \{ \text{definition of } \text{segs} \} \\
= \quad \text{concat} \; (\map \; \text{inits} \; \text{tails} \; (x : xs)) \\
= \quad \{ \text{definition of } \text{tails} \} \\
= \quad \text{concat} \; (\map \; \text{inits} \; ((x : xs) : \text{tails} \; xs)) \\
= \quad \{ \text{definition of } \map \} \\
= \quad \text{concat} \; (\text{inits} \; (x : xs) : \map \; \text{inits} \; (\text{tails} \; xs)) \\
= \quad \{ \text{definition of } \text{concat} \} \\
= \quad \text{inits} \; (x : xs) + (+) \; \text{concat} \; (\map \; \text{inits} \; (\text{tails} \; xs)) \\
= \quad \{ \text{definition of } \text{segs} \} \\
= \quad \text{inits} \; (x : xs) + (+) \; \text{segs} \; xs.
\]

Therefore,

\[
\text{segs} \; [\;] = [[[\;]]] \\
\text{segs} \; (x : xs) = \text{inits} \; (x : xs) + (+) \; \text{segs} \; xs
\]

(b) (10 points) Define \( \text{lep} = \text{maximum} \cdot \map \; \text{length} \cdot \filter \; \text{all even} \cdot \text{inits} \). Prove that \( \text{lep} \) has the following inductive definition:

\[
\begin{align*}
\text{lep} \; [\;] &= 0 \\
\text{lep} \; (x : xs) &= \text{if even} \; x \; \text{then} \; 1 + \text{lep} \; xs \; \text{else} \; 0,
\end{align*}
\]

**Note** The definition is equivalent to

\[
\begin{align*}
\text{lep} \; [\;] &= 0 \\
\text{lep} \; (x : xs) \big| \text{even} \; x &= 1 + \text{lep} \; xs \\
\text{otherwise} &= 0.
\end{align*}
\]

Proving either will do.
Solution: It’s immediate that \( \text{lep} \, [] = 0 \). For the inductive case, we reason:

\[
\begin{align*}
\text{lep} \, (x \, : \, xs) &= \text{maximum} \, (\text{map length} \, (\text{filter} \, (\text{all even}) \, (\text{inits} \, (x \, : \, xs)))) \\
&= \{ \text{definitions of inits } \}
\text{maximum} \, (\text{map length} \, (\text{filter} \, (\text{all even}) \, ([] \, : \, \text{map} \, (x \, : \, (\text{inits} \, xs)))))
&= \{ \text{definitions of filter, map, and maximum } \}
0 \uparrow \text{maximum} \, (\text{map length} \, (\text{filter} \, (\text{all even}) \, (\text{map} \, (x \, : \, (\text{inits} \, xs)))))
&= \{ \text{by (1) } \}
0 \uparrow \text{maximum} \, (\text{map} \, ((1+) \cdot \text{length}) \, (\text{filter} \, (\text{all even}) \, (\text{inits} \, xs))))
&= \{ \text{definition of lep } \}
0 \uparrow 1 + \text{lep} \, xs
&= 1 + \text{lep} \, xs.
\end{align*}
\]

We perform a case analysis on \( \text{even} \, x \):

Case \( \text{even} \, x \)

\[
0 \uparrow \text{maximum} \, (\text{map} \, ((1+) \cdot \text{length}) \, (\text{filter} \, (\text{all even}) \, (\text{inits} \, xs))))
= \{ \text{(4) } \}
0 \uparrow \text{maximum} \, (\text{map length} \, (\text{filter} \, (\text{all even}) \, (\text{inits} \, xs))))
= \{ \text{(5) } \}
0 \uparrow 1 + \text{maximum} \, (\text{map length} \, (\text{filter} \, (\text{all even}) \, (\text{inits} \, xs))))
= \{ \text{definition of lep } \}
0 \uparrow 1 + \text{lep} \, xs
= 1 + \text{lep} \, xs.
\]

Case \( \neg(\text{even} \, x) \)

\[
0 \uparrow \text{maximum} \, (\text{map} \, ((1+) \cdot \text{length}) \, (\text{filter} \, (\text{all even}) \, (\text{inits} \, xs))))
= \{ \text{(3) } \}
0 \uparrow \text{maximum} \, (\text{map length} \, (\text{filter} \, (\text{all even}) \, (\text{inits} \, xs))))
= \{ \text{maximum } [] \}
= 0 \uparrow -\infty
= 0
\]

Thus we have

\[
\begin{align*}
\text{lep} \, [] &= 0 \\
\text{lep} \, (x \, : \, xs) \mid \text{even} \, x &= 1 + \text{lep} \, xs \\
\mid \text{otherwise} &= 0.
\end{align*}
\]
(c) (10 points) Express \( \text{les} \ (x : xs) \) in terms of \( \text{lep} \) and \( \text{les} \).

**Solution:**

\[
\text{maximum} \ (\text{map length} \ (\text{filter} \ (\text{all even}) \ (\text{segs} \ (x : xs)))) \\
= \{ \ \text{inductive definition of segs} \ \} \\
\text{maximum} \ (\text{map length} \ (\text{filter} \ (\text{all even}) \ (\text{inits} \ (x : xs) \uplus \text{segs} \ xs))) \\
= \{ \ (6), \ (7), \text{ and } (8) \ \} \\
\text{maximum} \ (\text{map length} \ (\text{filter} \ (\text{all even}) \ (\text{inits} \ (x : xs)))) \uparrow \\
\text{maximum} \ (\text{map length} \ (\text{filter} \ (\text{all even}) \ (\text{segs} \ xs))) \\
= \text{lep} \ (x : xs) \uparrow \text{les} \ xs.
\]

(d) (10 points) Define \( \text{leps} \ xs = (\text{lep} \ xs, \text{les} \ xs) \). Derive an inductive definition for \( \text{leps} \).

**Solution:**

\[
\text{leps} \ (x : xs) \\
= (\text{lep} \ (x : xs), \text{les} \ (x : xs)) \\
= \{ \ \text{definition of lep} \ \} \\
(\text{lep} \ (x : xs), \text{lep} \ (x : xs) \uparrow \text{les} \ xs) \\
= \text{let } p' = \text{lep} \ (x : xs) \text{ in } (p', p' \uparrow \text{les} \ xs) \\
= \{ \ \text{definition of lep} \ \} \\
\text{let } p' = \text{if } \text{even} \ x \text{ then } 1 + \text{lep} \ xs \text{ else } 0 \\
\text{in } (p', p' \uparrow \text{les} \ xs) \\
= \text{let } (p, s) = \text{leps} \ xs \\
\quad p' = \text{if } \text{even} \ x \text{ then } 1 + p \text{ else } 0 \\
\quad \text{in } (p', p' \uparrow s).
\]

\[
\text{leps} \ [ ] = (0, 0) \\
\text{leps} \ (x : xs) = \text{let } (p, s) = \text{leps} \ xs \\
\quad p' = \text{if } \text{even} \ x \text{ then } 1 + p \text{ else } 0 \\
\quad \text{in } (p', p' \uparrow s)
\]

2. (20 points) Simply-typed \( \lambda \) calculus and Curry-Howard isomorphism.
(a) (10 points) Prove, using natural deduction, that

\[(P \rightarrow Q \rightarrow R) \rightarrow P \rightarrow ((P \rightarrow Q \rightarrow R) \rightarrow Q) \rightarrow R.\]

(b) (10 points) Write down a \(\lambda\) term having the types above.

**Solution:**

(a) Abbreviate \(P \rightarrow Q \rightarrow R, P, (P \rightarrow Q \rightarrow R) \rightarrow Q\) to \(\Gamma:\)

\[
\begin{align*}
\Gamma \vdash P \rightarrow Q \rightarrow R & \quad \text{Hyp} \\
\Gamma \vdash P & \quad \text{Hyp} \\
\Gamma \vdash Q \rightarrow R & \quad E \\
\Gamma \vdash P & \quad \text{Hyp} \\
\Gamma \vdash (P \rightarrow Q \rightarrow R) \rightarrow Q & \quad \text{Hyp} \\
\Gamma \vdash P \rightarrow Q \rightarrow R & \quad \text{Hyp} \\
\Gamma \vdash Q & \quad E \\
\Gamma \vdash P \rightarrow Q \rightarrow R \rightarrow Q & \quad E \\
\Gamma \vdash P & \quad \text{Hyp} \\
\Gamma \vdash ((P \rightarrow Q \rightarrow R) \rightarrow Q) \rightarrow R & \quad \text{Hyp} \\
\Gamma \vdash (P \rightarrow Q \rightarrow R) \rightarrow ((P \rightarrow Q \rightarrow R) \rightarrow Q) \rightarrow R & \quad \text{Hyp} \\
\Gamma \vdash ((P \rightarrow Q \rightarrow R) \rightarrow Q) \rightarrow R & \quad I \\
\Gamma \vdash \, & \quad \text{I} \\
\end{align*}
\]

(b) \(\lambda x.\lambda y.\lambda z. x y (z x)\)

3. (30 points) The aim of this problem is to derive a quick way to compute \((\Sigma i, j : 0 \leq i < j < N : (a[i] - a[j])^2)\) for array \(a\) and constant \(N\).

(a) (15 points) Take \(n\) as a constant such that \(1 \leq n\). Express \((\Sigma i, j : 0 \leq i < j < n + 1 : (a[i] - a[j])^2)\) as a polynomial of

- \((\Sigma i, j : 0 \leq i < j < n : (a[i] - a[j])^2)\),
- \((\Sigma i : 0 \leq i < n : a[i]^2)\),
- \((\Sigma i : 0 \leq i < n : a[i])\),

and some other terms. You may need the following properties, for \(\oplus\) that is associative, commutative, with identity \(e\), and for \(\otimes\) such that \((x \oplus y) \otimes z = (x \otimes z) \oplus (y \otimes z)\):

\[
\begin{align*}
(\oplus i : R : F) \oplus (\oplus i : R : G) &= (\oplus i : R : F \oplus G), \\
(\oplus i : R : F \otimes G) &= (\oplus i : R : F) \otimes G, \quad \text{where } i \text{ is not in } G, \\
(\Sigma i : 0 \leq i < n : K) &= n \times K \text{ for constant } K.
\end{align*}
\]

**Solution:**

\[
(\Sigma i, j : 0 \leq i < j < n + 1 : (a[i] - a[j])^2)
\]

\[
= \{ \text{split off } j = n \} \\
(\Sigma i, j : 0 \leq i < j < n : (a[i] - a[j])^2) + (\Sigma i : 0 \leq i < n : (a[i] - a[n])^2)
\]
(b) (15 points) Derive a solution for:

|| con N : int{N ≥ 1}; a : array[0..N] of int;
| var r : int;
| S
| {r = (Σi, j : 0 ≤ i < j < N : (a[i] − a[j])2)}
| ||

You get full points only if you come up with a linear-time program, with invariant, bound and necessary proofs. You may refer to previous sub-problems when needed. **Hint:** when you split off a value, be sure that the range is non-empty.

**Solution:** Replace constant \( N \) by variable \( n \). Let

\[
P_0 : r = (\Sigma i, j : 0 \leq i < j < n : (a[i] - a[j])^2),
\]

\[
P_1 : 1 \leq n \leq N.
\]

and assume that we will use an up-loop ending with \( n := n + 1 \). Note that \( n \) is at least 1 in \( P_1 \) — had we had \( 0 \leq n \), the step “splitting off \( j = n \)” in the calculation below wouldn’t be valid, as the range \( 0 \leq i < j < n + 1 \) could be empty.

With the hint above, we introduce two more variables:

\[
Q_0 : s = (\Sigma i : 0 \leq i < n : a[i]^2),
\]

\[
Q_1 : t = (\Sigma i : 0 \leq i < n : a[i]).
\]
With the invariant $P_1$, the loop has to start with $n = 1$. Thus the code:

```plaintext
  || con N : int {N ≥ 1}; a : array [0..N) of int;
  var r, s, t : int;

  r, s, t, n := 0, a[0]^2, a[0], 1
  \{ P_0 \land P_1 \land Q_0 \land Q_1, bnd : N - n \}
  \; do n \neq N \rightarrow
     r := r + s - 2 \times a[n] \times t + n \times a[n]^2;
     s := s + a[n]^2;
     t := t + a[n];
     n := n + 1;
   od
  \{ r = (\Sigma i, j : 0 \leq i < j < N : (a[i] - a[j])^2) \}

  ||.
```

4. (10 points) Attached is the complete code for our first implementation of Conway’s Game of Life, where a world is represented by

```
  type World = (Int, Int) \rightarrow Bool,
```

that is, a function from the coordinates to a Boolean indicating whether there is life in the specific cell.

One of the extension was to give different colours to cells of different ages. For example, all the newly born cell are in red, all the cells surviving the first year are in orange, the two-year-old cells are in green, and all the cells older than three are in blue.

To deal with that we may have to use a different representation of the world that records more information:

```
  type World = (Int, Int) \rightarrow Maybe Int,
```

where `Maybe` is a built-in type defined by

```
  data Maybe a = Nothing | Just a.
```

The intention is that if there is no life in a cell, the coordinate maps to `Nothing`. Otherwise the coordinate maps to `Just n` where $n$ is the age of the living cell (between, for example, $0 - 3$).

With the change in data structure, you have to make changes in the rest of the code accordingly. Mark directly on the printed code where the changes should be. There may be quite a number of places.

You are free to decide how many colours you want (as long as that is more than one), what colours they are, etc.
You may find the following function helpful:

\[
\text{maybeElim} \quad :: \quad (a \rightarrow b) \rightarrow b \rightarrow \text{Maybe} \ a \rightarrow b
\]

\[
\text{maybeElim} \ f \ y \ \text{Nothing} = y
\]

\[
\text{maybeElim} \ f \ y \ (\text{Just} \ x) = f \ x.
\]

\[
\text{isJust} \quad :: \quad \text{Maybe} \ a \rightarrow \text{Bool}
\]

\[
\text{isJust} \ \text{Nothing} = \text{False}
\]

\[
\text{isJust} \ (\text{Just} \ x) = \text{True}
\]

\[
\text{fromJust} \quad :: \quad \text{Maybe} \ a \rightarrow a
\]

\[
\text{fromJust} \ (\text{Just} \ x) = x
\]