FUNCTIONAL PROGRAMMING 2017

MOISE GABRIEL

QUESTION 4

(a)

```
> inits :: [a] -> [[a]]
> inits [] = [[]]
> inits (x:xs) = [] : map (x:) (inits xs)
```

(b)

```
> scanl f e = map (foldl f e) . inits
```

map :: (a -> b) -> [a] -> [b]
foldl f e :: [a] -> b
=> map (foldl f e) :: [[a]] -> [b]
inits :: [a] -> [[a]]
=>

=> map (foldl f e) . inits :: [a] -> [b] => scanl :: (b -> a -> b) -> b -> [a] -> [b]

(c) The previous definition of scanl is inefficient because it requires inits first, which runs in $O(N^2)$, where N = length xs, as the number of steps for an N-sized list is given by $T(N) = 1 + N - 1 + T(N-1) = N + T(N-1)$, with $T(0) = 1 \Rightarrow T(N) \approx \frac{N(N+1)}{2} \Rightarrow T(N) = O(N^2)$. After this we obtain a list consisting (N+1) lists, and because we apply (foldl f e), which runs in $O(N)$ on each of them, we again need $O(N^2)$.

What we can do is keep track of the changes that happen in e, the accumulator, and attach them to the list we will return, like a foldl. The new definition looks like this

```
> scanl' :: (b -> a -> b) -> b -> [a] -> [b]
> scanl' f e [] = [e]
> scanl' f e (x:xs) = e : scanl' f (f e x) xs
```

This version of scanl runs in $O(N)$ because we only apply f N times and we keep track of the intermediate results at each step.

QUESTION: How do I make a derivation? Is what I did close to that? I've seen this type of task several times, but I think I have not fully understood the concept behind it. Can you please show me a way to solve it?

(d)

```
> data Natural = Zero | Succ Natural

> foldNat :: (a -> a) -> a -> Natural -> a
> foldNat f e Zero = e                              (*)
> foldNat f e (Succ m) = f (foldNat f e m)          (**)
```

and we also know that:

① f is strict

② $f\ a = b$

③ $f \cdot g = h \cdot f$

and we want to show that

$f.\ \text{foldNat}\ g\ a = \text{foldNat}\ h\ b$

As $\text{foldNat}\ h\ b :: \text{Natural} \to \alpha$, we will work on three cases:

I

$(f.\ \text{foldNat}\ g\ a)\ \bot =$

$= \{\text{definition of } (.)\}$

$f(\text{foldNat}\ g\ a\ \bot) =$

$= \{\text{strictness of foldNat (from pattern-matching)}\}$

$f\ \bot =$

$= \{①\}$

$\bot$

$\text{foldNat}\ h\ b\ \bot =$

$= \{\text{strictness of foldNat}\}$

$\bot$

II

$(f.\ \text{foldNat}\ g\ a)\ \text{Zero} =$

$= \{\text{definition of } (.)\}$

$f(\text{foldNat}\ g\ a\ \text{Zero}) =$

$= \{(*)\}$

$f\ a =$

$= \{②\}$

b

$\text{foldNat}\ h\ b\ \text{Zero} =$

$= \{(*)\}$

b

III We will now suppose that the equality we want to prove holds for m from the "Natural" datatype, and we will show that it also holds for (Succ m).

$$(f \cdot \text{foldNat } g\ a)\ (\text{Succ } m) =$$
$$= \{\text{definition of } (.)\}$$
$$f\ (\text{foldNat } g\ a\ (\text{Succ } m)) =$$
$$= \{(**)\}$$
$$f\ (g\ (\text{foldNat } g\ a\ m)) =$$
$$= \{\text{definition of } (.)\}$$
$$(f \cdot g)\ (\text{foldNat } g\ a\ m) =$$
$$= \{3\}$$
$$(h \cdot f)\ (\text{foldNat } g\ a\ m) =$$
$$= \{\text{definition of } (.)\}$$
$$h\ (f\ (\text{foldNat } g\ a\ m)) =$$
$$= \{\text{definition of } (.)\}$$
$$h\ ((f \cdot \text{foldNat } g\ a)\ m) =$$
$$= \{\text{inductive hypothesis}\}$$
$$h\ (\text{foldNat } h\ b\ m) =$$
$$= \{(**)\}$$
$$\text{foldNat } h\ b\ (\text{Succ } m)$$

From I, II and III, we proved that

$$f \cdot \text{foldNat } g\ a = \text{foldNat } h\ b$$