CPSC 429/529: Functional Programming
Carsten Schürmann
Date: February 28, 2001

Midterm

February 28, 2001.
9:30am.
Closed Book.

1. Lazy Streams

An infinite sequence $x_0, x_1, x_2, \ldots$ can be represented by a lazy stream. For each function your write below give its type, including auxiliary functions where necessary. You may always assume that the integer argument to your functions are greater or equal to 0 without checking.

1.1. [10 points]

Give a representation of the sequence $1, -\frac{1}{3}, +\frac{1}{5}, -\frac{1}{7}, \frac{1}{9} \ldots$.

1.2. [10 points]

Write a function `diff` which, given $x_0, x_1, x_2, \ldots$ computes the sequence of differences, consiting of $x_1 - x_0, x_2 - x_1, \ldots$.

1.3. [10 points]

Write a function `sum` which, given a sequence $x_0, x_1, x_2, \ldots$ returns the sequence consisting of the partial sums $x_0, x_0 + x_1, x_0 + x_1 + x_2, \ldots$.

1.4. [10 points]

The number π can be written as the infinite sum

$$4(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} \ldots).$$

Write a function `piapprox` which expects one argument n and computes an approximation of π using the first n elements of this series.

2. Continuations and Higher-Order Functions

Consider the language $\mathcal{S}$ of all siamese strings such as "bobo" or "mama" which consist of an arbitrary string, repeated twice:

$$\mathcal{S} = \{s \mid s = xx \text{ for some string } x\}$$

Note that the empty string ϵ is a member of $\mathcal{S}$. It turns out that the language $\mathcal{S}$ is *not* the language of any regular expression. This problem asks you to implement an acceptor for this language using continuations.

2.1. [15 points]

Using continuations, write a function

```
twin' :: string -> (string, string -> bool) -> bool
```

satisfying the following specification:

1. `twin'` $s\ k \hookrightarrow$ `true`
   if there exists s_1 and s_2 such that $s = s_1 s_2$ and $k(s_1, s_2) \hookrightarrow$ `true`.

2. `twin'` $s\ k \hookrightarrow$ `false`
   otherwise.

Note, that `twin'` is quite general and will be used in the next question to write an acceptor for Siamese strings.

2.2. [10 points]

Using the function `twin'`, write a function

```
twin :: string  -> bool
```

with the specification that `twin` $s \hookrightarrow$ `true` if $s \in \mathcal{S}$, and `twin` $s \hookrightarrow$ `false` otherwise.

2.3. [15 points]

Prove the correctness of your function `twin`, assuming the correctness of the helper function `twin'`.

3. Monads

Let $M\ a$ be a monad with the standard operations

$$\begin{array}{lcl} \text{unit} & :: & (a \to M\ a) \\ \star & :: & M\ a \to (a \to M\ b) \to M\ b \end{array}$$

which satisifies the following laws:

$$\begin{array}{lrcl} \text{Left unit:} & \text{unit}\ a \star \lambda b.\, n & = & n[a/b] \\ \text{Right unit:} & m \star \lambda a.\, \text{unit}\ a & = & m \\ \text{Associativity:} & m \star (\lambda a.\, (n \star \lambda b.\, o)) & = & (m \star \lambda a.\, n) \star \lambda b.\, o \end{array}$$

Let the following laws be given:

$$\begin{array}{lcl} \text{map} & :: & (a \to b) \to (M\ a \to M\ b) \\ \text{map}\ f\ m & = & m \star (\lambda a.\, \text{unit}\ (f\ a)) \\ \text{join} & :: & M\ (M\ a) \to M\ a \\ \text{join}\ z & = & z \star (\lambda m.\, m) \end{array}$$

3.1. [10 points]

Prove that

$$m \star k = \text{join}\ (\text{map}\ k\ m).$$

Justify each step of your proof properly.

3.2. [10 points]

Prove that

$$\text{map}\ (f \circ g) = (\text{map}\ f) \circ (\text{map}\ g).$$

Recall, that $(f \circ g)\ x = f(g\ x)$. Justify each step of your proof properly.