Funktionale Programmierung

`http://proglang.informatik.uni-freiburg.de/teaching/functional-programming/2013/`

Übungsblatt 10 (Arrows, Streamfunktionen)

Mi, 2014-01-22

Hinweise

- Lösungen sollen in das persönliche Subversion (svn) Repository hochgeladen werden. Die Adresse des Repositories wird per Email mitgeteilt.
- **Alle** Aufgaben müssen bearbeitet und pünktlich abgegeben werden. Falls das sinnvolle Bearbeiten einer Aufgaben nicht möglich ist, kann eine stattdessen eine Begründung abgegeben werden.
- Wenn die Abgabe korrigiert ist, wird das Feedback in das Repository hochgeladen. Die Feedback-Dateinamen haben die Form `Feedback-<user>-ex<XX>.txt`.
- Allgemeinen Fragen zum Übungsblatt können im Forum (`http://proglang.informatik.uni-freiburg.de/forum/viewforum.php?f=38`) geklärt werden.

Abgabe: Mi, 2014-01-29

Hinweis: Auf der Vorlesungsseite finden Sie das Module `Arrows.hs`, das die Definitionen für Arrows aus der Vorlesung zusammenfasst.

Aufgabe 1 (filterA)
Definieren Sie

```
filterA :: ArrowChoice arr => arr a Bool -> arr [a] [a]
```

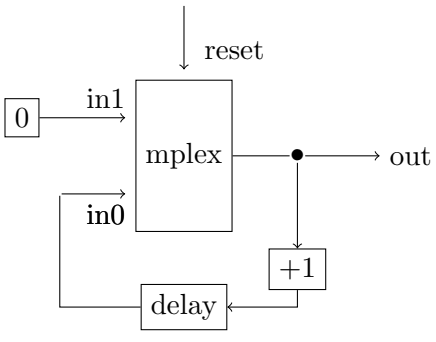
Für die ArrowChoice-Instanz von Funktionen soll sich filterA so verhalten wie **filter** und für Kleisli-Arrows so wie **filterM** (aus Control.**Monad**).

Um herauszufinden wie filterA sich für die SF-Instanz verhält, experimentieren Sie mit Ausdrücken wie

```
filterA (arr even >>> delay True)
```

auf Eingabeströmen von Listen verschiedener Längen. Beschreiben Sie das Verhalten.

Aufgabe 2 (Zähler)
Betrachten Sie das folgende „Schaltkreisdiagramm" eines Zählers:



Der Zähler hat einen booleschen Eingang reset und einen Integer Ausgang out. Solange das reset-Signal auf **False** steht, zählt der Ausgangswert das vorherige Ergebnis nach oben. Wenn das reset-Signal auf **True** gesetzt wird, setzt der Zähler den Ausgang auf 0 zurück.

Implementieren Sie den Zähler als Streamfunktion SF **Bool Int**, indem Sie die „Bauteile" aus dem Diagram implementieren und verbinden. Benutzen Sie dabei nur die in der Vorlesung definierten Streamfunktionen und die Methoden der Arrow-Klassen. (D.h. verwenden Sie *nicht* die konkrete Implementierung der Streamfunktionen, also den Konstruktor SF und die darunterliegenden [a] -> [b] Funktionen.)

Hinweis: Das Bauteil „mplex" stellt eine Art Multiplexer da. Wenn das reset-Signal auf 1 steht wird der Eingang in1 an den Ausgang übertragen, ansonsten der Eingang in0.

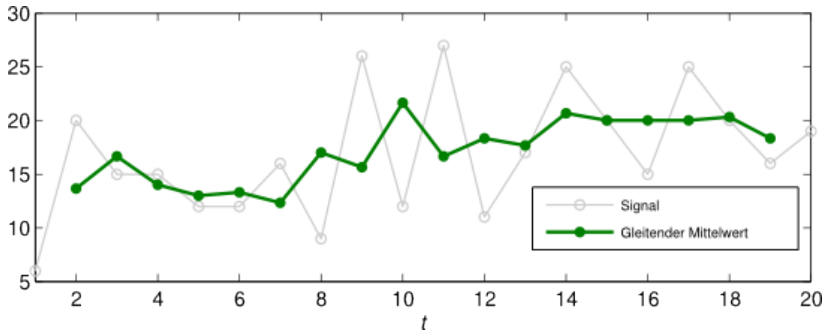
Aufgabe 3 (Gleitender Mittelwert)
Der *einfach gleitende Mittelwert* [1](SMA, simple moving average) n-ter Ordnung für eine diskrete Zeitreihe $x(t)$ ist definiert als:

$$\mathrm{sma}_n(t) = \frac{1}{n}\sum_{i=0}^{n-1} x(t-i)$$

Hier ist ein Beispiel des laufenden Mittelwerts für $n = 3$ und die Reihe

`[6, 20, 15, 15, 12, 12, 16, 9, 26, 12, 27, 11, 17, 25, 20, 15, 25, 20, 16, 19]`

(für t = [1..20], Quelle: Wikipedia)



Diskrete Zeitreihen $x(t)$ lassen Sich als Streamfunktionen vom Typ SF **Int Double** approximieren. Implementieren Sie also den gleitenden Mittelwert für Streamfunktionen:

```
sma :: Int -> SF Int Double -> SF Int Double
sma n xt = ...
```

Gehen Sie dabei wie folgt vor: Implementieren Sie zuerst *eine* der beiden folgenden Funktionen:

- windowSF :: a −> **Int** −> SF a [a] : Das Ergebnis ist eine SF, die die letzten n Werte ausgibt, statt nur den aktuellen. Die Funktion windowSF lässt sich mit den in der Vorlesung definierten SF-Operationen definieren, ohne das Sie die konkrete Implementierung von SF (**data** SF a b = SF ([a] −> [b])) verwenden müssen.
- foldSF :: (a −> b −> a) −> a −> SF b a: Das Ergebnis ist eine SF, die den laufenden Links-Fold bis zum aktuellen Element ausgibt. Hier müssen sie auf der konkreten Implementierung von SF arbeiten.

und benutzen Sie diese dann zur Definition des Summenoperators. Mit dem Summenoperator können Sie die Definition von sma dann einfach mit den Arrow-Methoden hinschreiben.

Wie unterscheidet sich das Verhalten ihrer Implementierung von der mathematischen Definition oben bzw. auf was müssen Sie beim Implementieren achten?

Hinweis: Die Funktion **fromIntegral** kann zur Umrechnung von **Int** auf **Double** verwendet werden.

Aufgabe 4 (State-Transformer)
In der Vorlesung wurde kurz die Familie der State-Transformer erwähnt:

```
newtype SB s a b = SB { runSB :: (s -> a) -> (s -> b) }
```

1. Definieren Sie die Arrow-Instanz für SB s.
2. Definieren Sie die Funktion runSFfromSB :: SB **Int** a b −> [a] −> [b], die einen State-Transformer als Streamfunktion interpretiert.

[1] `http://de.wikipedia.org/wiki/Gleitender_Mittelwert`