ITU-FRP2010, Copenhagen, 21–25 June 2010

Exercises 2: Yampa Basics

Henrik Nilsson

The objective of these exercises is to get acquainted with Yampa. The first step is to install Yampa (from Hackage, using Cabal) unless you have already done so.

Warning! Yampa was always "work in progress". While it has been used by many over the years and proved useful, the API is somewhat untidy. Moreover, there are scalability issues. Thus, the whole system is best viewed as a proof of concept.

To use Yampa, simply import `FRP.Yampa` into the module where it is needed. Or, from within, GHCi, simply switch to this using the `:module` command. Use `:browse` to see what Yampa provides.

```
Prelude> :module FRP.Yampa
Prelude FRP.Yampa> :browse
```

Alternatively, refer to the source code. The module `FRP.Yampa.Utilities` provides additional, derived, combinators. To use the arrow syntax in a module, start it with:

```
{-# LANGUAGE Arrows #-}
```

(Put language pragmas before anything else in the file, at the very first lines.)

1. Define and try some simple signal functions. The easiest way to run them to use `embed` and maybe also `deltaEncode` to prepare the input. You may want to define your own version of `reactimate` along the lines of what you used in the first set of exercises using these to functions.

2. Implement a Yampa version of the following CFRP counter example from the first set of exercise, but now make sure switching does not reset the counter:

   ```
   let
       c = hold 0 (count (repeatedly 0.5))
   in
       c `until` after 5 -=> c * 2
   ```

3. Define the behavior e^t by exploiting the identity:

   $$e^x = 1 + \int_0^x e^y \, dy$$

   This should now work without problems.

4. Repeat the experiment, but now compute $f(t)$ defined by:

   $$f(x) = 1 + \int_0^x f(y) f(y) \, dy$$

   This should now work without problems.

5. Implement a bouncing ball model using `impulseIntegral`.