Yampa is a Haskell library for programming reactive systems. It is based on the concept of stream functions which takes an input stream and produce an output stream. Streams are infinite sequences of values and stream functions are non terminating functions. The lazyness of Haskell allows to implement streams and stream functions regardless of their infinite nature. In a strict language like OCaml this is not possible instead of using lazyness extensions. Here we propose an implementation of Yampa primitives in OCaml based on coiterators.

1 coiterators

A concrete stream carrying values of type a is a pair $(\mathit{step} : s \to a \times s, \mathit{init} : s)$ where s is the type of states used to generate the actual stream and init is the initial state.

$$\mathit{run}\ (c, s) = (\lambda(a, s).a :: (\mathit{run}\ (c, s)))(c\ s)$$

We have a family of functor indexed by type of states

$$F_S\ X = (S \to X \times S) \times S$$

2 Yampa primitives

Operators define functions over concrete stream

The `arr` primitive apply a function to all elements of a streams

$$\begin{array}{lcl} \mathit{head}\ (\mathit{arr}\ f\ s) & = & f\ (\mathit{head}\ s) \\ \mathit{tail}\ (\mathit{arr}\ f\ s) & = & \mathit{arr}\ f\ (\mathit{tail}\ \ s) \end{array}$$

This behavior can be obtained Given a concrete stream c, the $\mathit{arr}\ f\ c$ generates a new concrete stream. Arrow is a simple mapping of a function to all elements of a stream. It is the mapping function for the functor F.

At each step, it uses c to generate a value and a state and returns the vale (f a) and the new state.

$$\mathit{mapleft}\ f\ (x, y) = (f\ x, y)$$

$$\begin{array}{lcl} \mathit{arr} & :: & (A \to B) \to \mathit{Co}\ A\ C \to \mathit{Co}\ B\ C \\ & = & \lambda f\ (\mathit{co}\ h\ s).\mathit{co}\ ((\mathit{mapleft}\ f) \circ h)\ s \end{array}$$

- all frp operators are polymorphic in the state of the coiterators passed as input (which is a good thing, the computation should not depend on the representation) f l1 = f l2 si l1 et l2 représentent les memes suites de valeurs