1 Defunctionalizing Re

1.1 Informal Semantics

Defunctionalization of actions in the reactive resumption is similar to the transformation for resumptions, except that signaling over input/output ports must be taken into account. The reactive resumption monad includes a non-proper morphism, *signalRe*, defined in Haskell as:

$$signalRe :: Req \longrightarrow Re\ Rsp$$
$$signalRe\ q = P\ (q,\ \eta_K\ \circ \eta_{Re})$$

In CT, this standard definition is modified to take an additional argument specifying the port over which the request or response is transmitted. CT admits the declaration of input and output ports as part of the *State* layer of the monad, using first class types *InPort* and *OutPort* respectively, so that we may define *signal* as a CT built-in thus:

$$type\ Port = InPort + OutPort$$
$$signal :: Port \longrightarrow Req \longrightarrow Re\ Rsp$$

where for any $port \in Port$, *signal port* is semantically equivalent to an instance of *signalRe* in which the underlying communication channel is taken to be *port*.

1.2 Defunctionalization Procedure

The preceding construction allows a definition of defunctionalization in the reactive resumption monad as:

$$\lceil signal\ p\ q \rceil_{Re}\ =\ counter \star_M\ \ \lambda i\ .\ \eta_M((i,\ putreq_p\ q) \mapsto (i+1,\ getrsp_p\ q)) \tag{1}$$

The defunctionalization transformation is identical to that for non-reactive resumptions for all terms typed in Re.

The functions *putreq* and *getrsp* are intermediate representations of port behavior, with meaning determined by whether p is an input or output port. In the case that $p :: InPort$, the request q is implicit so that *putreq q* has no effect; *getrspq* has the effect of reading from p a response of the form anticipated by q. Conversely, in the case that $p :: OutPort$, the response to q is implied and *getrsp q* has no effect,, while *putreq q* writes signal q to to the output port p. These two functions then admit a straightforward translation into assignments in *FSMLang* in which $putreq_p$ represents the appearance of p on the left-hand side of an assignment, while $getrsp_p$ represents the appearance of p on the right-hand side.

The functions $putreq$ and $getrsp$ are represented as actions on the underlying state as:

$$\lceil putreq_p\ q \rceil_K = (x_1, ..., p, ..., x_c, v) \mapsto (x_1, ..., q, ..., x_c, v) \tag{2}$$

when p is an output port and

$$\lceil getrsp_p\ q \rceil_K = (x_1, ..., x_c, v) \mapsto (x_1, ..., x_c, read\ p) \tag{3}$$

when p is an input port. Only one of the above transitions occurs as a result of an application of $signal$; which is determined by the type ($InPort$ or $OutPort$) of p.