

Design and Selection of Programming Languages

7 October 2005

Exercise 4.1

We define a type of transition functions that define state transitions triggered by a *inputs* and also producing *outputs*:

type *Transition state input output* = (*state*, *input*) → (*state*, *output*)

(a) Define a Haskell function

process :: *Transition state input output* → *state* → [*input*] → [*output*]

that calculates the list of outputs produced by a transition function given a starting state and a list of inputs.

Using *process* from (a) and prelude functions, the definition

runprocess :: *Transition state String String* → *state* → IO ()

runprocess tr s = **do**

hSetBuffering stdout LineBuffering -- requires: "import System.IO" at beginning of module
interact (unlines ∘ process tr s ∘ lines)

allow *runprocess* to turn a transition with *String* inputs and outputs into a runnable program.

Try: *runprocess id 0*

(b) Define a transition function

countEcho :: *Transition Integer String String*

that keeps a counter as its state and otherwise just reproduces the input prefixed with line numbers as output.

Try: *runprocess countEcho 0*

(c) Define a transition function

trAdd :: *Transition Integer String String*

that uses the prelude functions *read* and *show* to add the *Integer* reading of the input to the accumulating state, and outputs that state as a string.

Try: *runprocess trAdd 0*

(d) Define a transition function

polish :: *Transition [Integer] String String*

that implements a reverse Polish notation calculator by pushing number inputs on the stack, always outputting the top of the stack (if present), and interpreting +, -, *, / as taking their arguments from the stack and pushing the result back onto the stack.

Try: *runprocess polish []*