Software System Design and Implementation

Effects and IO Monad Practice

Curtis Millar
CSE, UNSW (and Data61)
1 July 2020

The IO Type

A procedure that performs some side effects, returning a result of type `a` is written as `IO a`.

World interpretation

`IO a` is an abstract type. But we can think of it as a function:

```
RealWorld -> (RealWorld, a)
```

(that's how it's implemented in GHC)

```
(>>=) :: IO a -> (a -> IO b) -> IO b
pure  :: a -> IO a

getChar :: IO Char
readLine :: IO String
putStrLn :: String -> IO ()
```

Two-player Tic-Tac-Toe

Example

Simple two-player Tic-Tac-Toe game

Done in editor

State Monads

```
newtype State s a = State (s -> (s, a))
```

State Monad

```
get :: State s s
put :: s -> State s ()
modify :: (s -> s) -> State s ()
```

Here we use a monadic interface to simplify the passing of our state around, so that we don't need to manually plumb data around.

Tic-Tac-Toe A.I

Example

Adding A.Is for Tic-Tac-Toe

Done in editor

QuickChecking Monads

QuickCheck lets us test `IO` (and `ST`) using this special property monad interface:

```
monadicIO :: PropertyM IO () -> Property
pre       :: Bool -> PropertyM IO ()
assert    :: Bool -> PropertyM IO ()
run       :: IO a -> PropertyM IO a
```

Testing a Tic-Tac-Toe A.I

Example

Testing A.Is for Tic-Tac-Toe

Done in editor

Homework

1. Next week is flexibility week
2. Last week's quiz is due on Friday. Make sure you submit your answers.
3. The fourth programming exercise is due by the start if my next lecture (in 14 days).
4. This week's quiz is also up, it's due Friday week (in 16 days).

Consultations

- Poll on Piazza to register interest. Will not run if there are no votes.
- Tomorrow, 9am to 11am on Blackboard Collaborate.
- Link on course website & Piazza.
- Make sure to join the queue on Hopper. Be ready to share your screen with REPL (`ghci` or `stack repl`) and editor set up.