CSE216 Programming Abstractions

Monads

YoungMin Kwon

Monads

- Monads model computations
- Computation
  - Like a function that maps from input to output
  - But it may do something more
    - Side effects (e.g. printing)
  - Monads provide an abstraction of effects

Monads

- Monad operations
  - return
    - Put a value in some wrapper
    - Takes a value of type *a* and returns a monadic value of type *m a*
  - bind (>>=)
    - Takes a function *f* of type $a \rightarrow m\ b$ and returns a monadic value of type *m b* after applying *f* to *a*

State Monad

- Managing a state without assignments

```
module State = struct
    (*monad operations*)
    let ret v = fun s -> (s, v)   (*ini state -> fin state, val*)
    let (>>=) m f =
        fun s ->
            let (s', v) = m s in  (*s/s': ini/fin state of m*)
            (f v) s'              (*s'  : ini state of (f v)*)
    (*other operations*)
    let get = fun s -> (s, s)     (*val is curr state s*)
    let put s' = fun s -> (s', ())(*replace curr state s with s'*)
end
```

  - Monad is a function from an initial state to a final state with a value

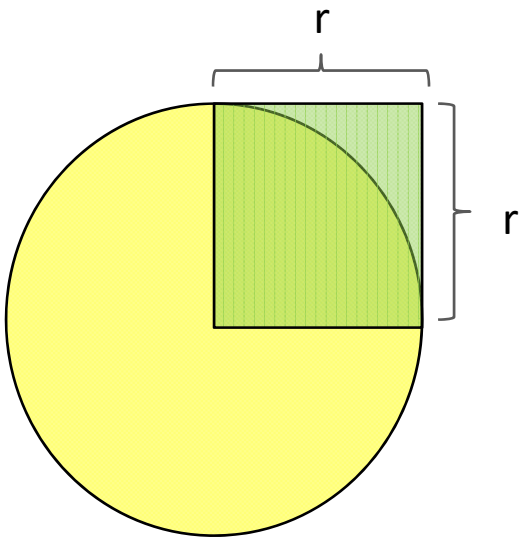
State Monad

```
module State = struct
    (*monad operations*)
    let ret v = fun s -> (s, v)
    let (>>=) m f =
        fun s ->
            let (s', v) = m s in
            (f v) s'
    (*other operations*)
    let get = fun s -> (s, s)
    let put s' = fun s -> (s', ())
end
```

- ret
  - A function whose initial state and final state are the same
- >>=
  - A function which uses the final state s' of its first argument as the initial state of its second argument
- get
  - A function which leaves the state unmodified and returns it as a result (get the current state)
- put
  - A function which ignores the initial state, replacing it with the value supplied as an argument (put the new state)

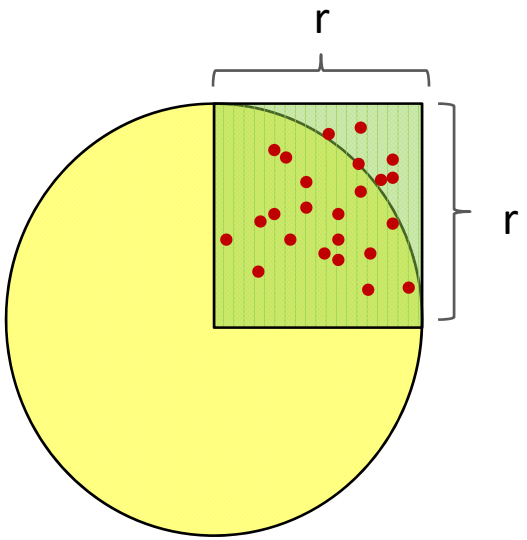
Estimating π

- Estimating π
  - Area of the square
    $r \cdot r = r^2$
  - Area of the disk under square
    $\pi \cdot r^2 / 4$
  - The area of the quarter disk over the area of the square
    $(\pi \cdot r^2 / 4) / r^2 = \pi / 4$



Estimating π

- To estimate π
  - Mark *n* random points within the square
    - (*x*, *y*) where *x* = *rand r*, *y* = *rand r*
  - Counter the number of points (x, y) within the quarter disk
    - $x^2 + y^2 < r^2$
  - The fraction of the points within the quarter disk over *n* is approximately π / 4



Implement rand

- Using the State monad, implement rand

```
let rand n =
    (* random number generator
        implement a random number generator
        using the State monad such that

        x[0] = given
        x[i] = x[i-1] * 16807 mod 0x7fffffff
        return value x[i] mod n
    *)

let _ = (*7, 9, 3*)
    let test _ =
        rand 10 >>= fun x -> ret (printf "%d\n" x) in
    (test () >>= test >>= test) 1
```

Estimate π using rand

```
let pi n =
    (* estimate pi
        - generate random numbers x and y, in the range of
          [0..999] with the seed value of 2, n times
        - count the number of cases when x^2 + y^2 < 1000^2
        - the fraction of the count over n is about pi/4
    *)


let _ = pi 100000 (*3.14292*)
```