CSE216 Programming Abstractions

Streams

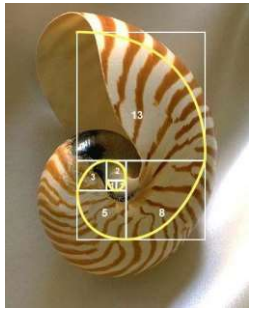
YoungMin Kwon

Streams

```
(** Stream ***************************************
*)
type 'a stream = Nil | Cons of 'a * (unit -> 'a stream)

let cons = fun h thunk ->
    Cons (h, thunk)

(*TODO: implement car
*)
let car = function


(*TODO: implement cdr
*)
let cdr = function


(*TODO: implement map
*)
let rec map f strm =
```

```
(*TODO: implement index (n-th element of strm)
*)
let rec index n strm =


(*print the first n elements of strm*)
let rec print n strm =
    strm    |>  map (fun x -> Printf.printf "%6.3f, " x)
            |>  index n;
    Printf.printf "\n"

(*TODO: implement const
    - const c returns a stream of c
*)
let rec const c =


let zeros       = const 0.
let ones        = const 1.

let _ = zeros |> print 5
let _ = ones  |> print 5
(*  0.000,  0.000,  0.000,  0.000,  0.000,  0.000,
    1.000,  1.000,  1.000,  1.000,  1.000,  1.000,
*)
```

Fibonacci Stream

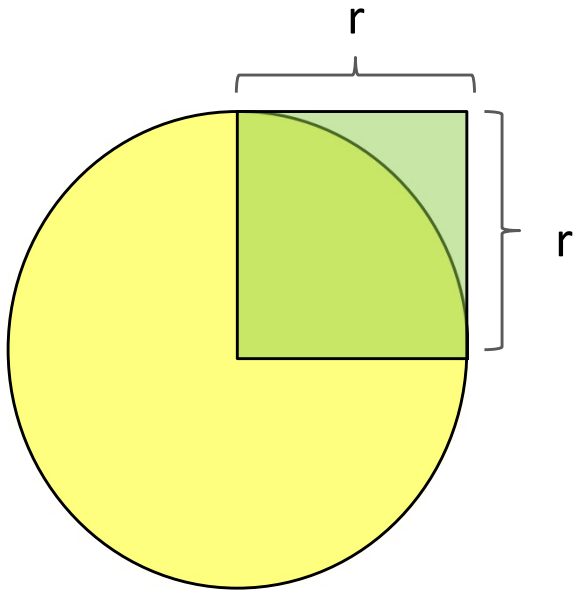


```
(** Fibonacci stream  ***********************************************
*)
(*TODO: implement sum
    - sum a_strm b_strm returns a stream of the sum of a_strm and b_strm
    - a_strm and b_strm are streams of float
    - e.g. sum [1; 3; 1; 2; ...] [0; 1; 2; 3; ...] = [1; 4; 3; 5; ...]
*)
let rec sum a_strm b_strm =


let _ = sum ones ones |> print 5
(* 2.000,  2.000,  2.000,  2.000,  2.000,  2.000,*)
```

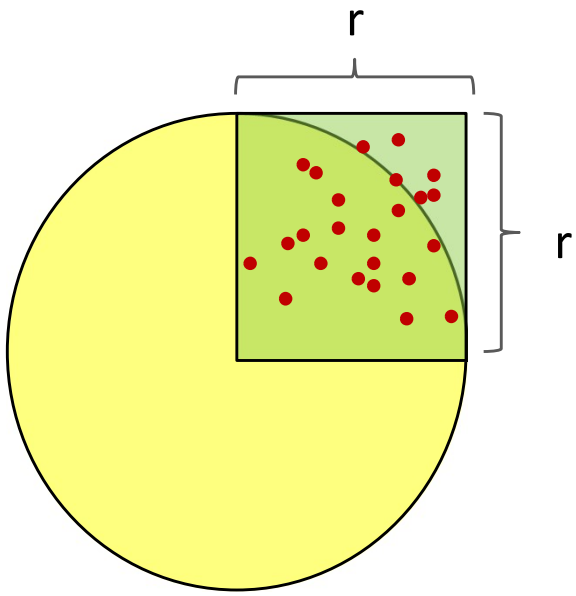
```
(*TODO: implement fibs using sum
    - fibs returns a stream of Fibonacci numbers
    - fib(t) = fib(t-1) + fib(t-2)

      fib(0, 1, 2, ...): 0, 1, 1, 2, 3, 5, ...
    + fib(1, 2, 3, ...): 1, 1, 2, 3, 5, ...
    = fib(2, 3, 4, ...): 1, 2, 3, 5, 8, ...
*)
let rec fibs () =


let _ = fibs () |> print 8
(* 0.000,  1.000,  1.000,  2.000,  3.000,  5.000,  8.000, 13.000, 21.000,*)
```

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
    $(\pi \cdot r^2 / 4) / r^2 = \pi / 4$

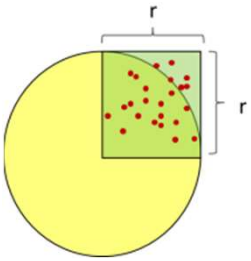


Estimating π



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

Estimating π




```
(** pi stream ***********************************************
*)
(* TODO: implement rand_stream
   - return a stream of random numbers in between 0. and 1.
*)
let rec rand_stream seed =
   let rand_max = 0x7fffffff in
   let rand_update x = (x * 16807) mod rand_max in
```

```
(*TODO: implement inside stream
    - take two random numbers for x and y and
      test if they are inside a unit circule
*)
let rec inside_stream r_strm = (*count inside, count total, rand stream*)
    let inside x y =
        x *. x +. y *. y < 1. in
```

```
(*TODO: implement monte carlo stream
    - returns a stream of estimates for the success probability of experiment
*)
let rec monte_carlo nr_passed nr_trials experiment =



(*TODO: implement pi stream
    - a stream of estimations for pi
    - pipeline rand_sream 1, inside_stream, monte_carlo 0 0
      and map with (fun x -> 4. *. x)
*)
let rec pi_stream =


let _ =
    pi_stream
    |> index 1000000
    |> Printf.printf "pi ~ %f\n"

(*pi ~ 3.142096*)
```