Recitation 3 – *Scan*

Parallel and Sequential Data Structures and Algorithms, 15-210 (Spring 2015)

January 27th, 2015

1 Announcements

- How was *SignalLab*?
- Lab 2 – *SkylineLab* has been released and is due next Monday, February 2, 2015. This lab is conceptually much more difficult than the previous one, so start early!
- Questions from lecture or homework?

2 Semantics

```
scan: ('a * 'a -> 'a) -> 'a -> 'a seq -> 'a seq * 'a
```

`scan` takes three arguments:

- A function from `'a * 'a` to `'a`, much like `reduce`
- An initial value of type `'a`
- A sequence of type `'a seq` to operate on.

`scan` returns a list of intermediate results along with the final result. `scan f b s`, where `b` is the left-identity of `f`, is logically equivalent to

```
(tabulate (fn i => reduce f b (take (s, i))) (length s), reduce f b s)
```

Like `reduce`, `scan` evaluates in parallel to achieve a $O(n)$ work and $O(\log n)$ span (assuming that the function passed in has constant work).

Question:
What is the result of `scan (op+) 0 ⟨1, 2, 3, 4⟩`?

3 Associativity

Our discussion about `scan` assumes that the function passed in is *associative*. Recall the mathematical definition that a function f is said to be associative if and only if $\forall a \forall b \forall c . f(f(a,b),c) = f(a,f(b,c))$.

Question:
What is the result of `scan (op-) 0 ⟨1, 2, 3, 4⟩`?

Question:
Partition the following functions into Associative and Non-Associative:

1. `Int.max`
2. `Int.min`
3. `Int.mod`
4. `String.concat`
5. `fun copy (a, b) = case b of NONE => a | SOME _ => b`

3.1 Copy-Scan

A notable feature of scan is that it can be used to propagate information down a sequence, like `iter`, but in parallel! The only condition is that the function used must be associative.

Question:
Given

```
fun copy (a, b) =
    case b
     of NONE => a
      | SOME _ => b
```

what is the result of:

```
scan copy
     (SOME 0)
     ⟨NONE, NONE, SOME 1, NONE, SOME 2, SOME 3, NONE, NONE⟩
```

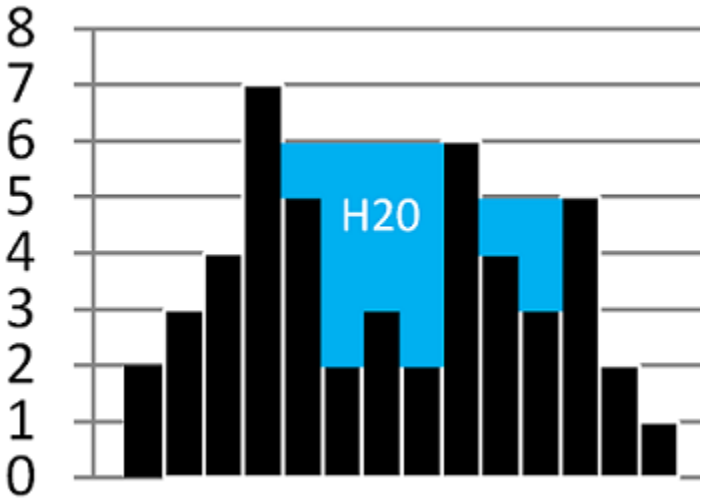
4 Applications

At first glance, `scan` seems to offer not much that isn't already available through `reduce`. With clever choices of associative functions, though, `scan` can be used to compute some surprising things efficiently in parallel.

4.1 Histogram

Consider the following problem:

Given a histogram, if we were to pour water over it, how much water (in terms of area) would it hold? For simplicity we will represent a histogram as a sequence of non-negative integers. For example the histogram shown below is represented by the sequence `s` $= \langle 2, 3, 4, 7, 5, 2, 3, 2, 6, 4, 3, 5, 2, 1 \rangle$, and holds 15 units of water.



Question:
Write a function to solve the above problem.

Consider calculating for each index the largest value to the left of it and the largest value to the right of it. We can then use the two values to calculate the amount of water held above that index.

```
fun histogramFill (hist : int seq) =
```