Blelloch 2

How to generate intermediate values?

(1, 0, 2, 7, 0, 5)

((0, 1, 10), 15) (1, 9, 5)

(1, 3, 10)

able to get partial sums by adding contracted items with remaining ones

works for any assoc operator

update : α seq $\times$ (int $\times \alpha$) $\rightarrow \alpha$ seq.

what semantics of exception in parallel setting? (can combine in parallel way)

fun update (A, (i, v)) =
tabulate (λj. if ($j = i$) then v else A[j]) |A|

$W = O(|A|)$
$S = O(1)$

(copy is embarassingly parallel)

inject | : α seq $\times$ ((int $\times \alpha$) seq) $\to \alpha$ seq

note : intro could be the same
(doesn't fully define this operation)

contention : last one wins

ex: $\langle a, b, c, d \rangle$

$\langle (0,x), (2,y), (0,z) \rangle$ ← last wins

$\langle z, b, y, d \rangle$

inject(A, U) W: $O(|A| + |U|)$
S: $O(\log |U|)$ breaks ties in log time

inject must be primitive to get costs

w/o costs:
inject(A, U) = iterate A update U

in terms of asymtotic costs, these are:

length
a[i] (?)
tabulate
inject

filter: f s
: $(\alpha \to bool) \to \alpha\ seq \to \alpha\ seq$

easy to do sequentially

$s = \langle a, b, c, d, e, g \rangle$
map f s = $\langle t, f, f, t, t, f \rangle$ map of f on s

$\langle a, d, e \rangle$

goal: $W : O(|s|)$
$S : O(\lg n)$

(could do: reduce append)

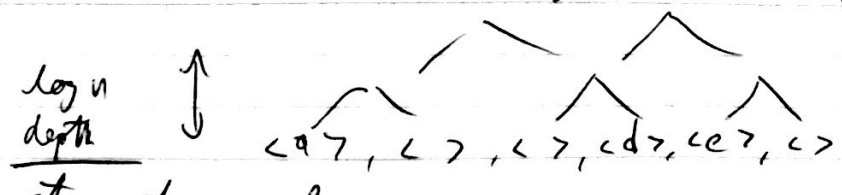
if T, return singleton
F return empty

$(\langle a \rangle, \langle\rangle, \langle\rangle, \langle d \rangle, \langle e \rangle, \langle\rangle)$ $W: O(|s|)$
$S: O(1)$

reduce <> append

append is O(1) span, so get lg n span

$S = O(\log n)$
$W = O(n \log n)$ $\Big\}\ n = |s|$ $\longrightarrow$ Not meet goal! ($W = O(n)$)



How to get linear work?

inject (A, V)

seq of flags <a, b, c, d, e, g>
<t, f, f, t, t, f>
0 1 2 ← locations in result

result <a, d, e>

prefix sums <1, 0, 0, 1, 1, 0>
→ <[0], 1, 1, [1], [2], 3>

fun filter f A =

let val fl = map (fn x ⇒ if f x then 1 else 0) A
val (off, total) = scan 0 op+ fl
(λ(x,y) → x+y)
val V = tabulate (fn i ⇒ (off[i], A[i])) |A|

in inject (tabulate (fn _ ⇒ a[0]) total) V

(note: still injecting filtered-out elements so that last one update wins)

(note: last elem at location 3 should be ruled out)

total span: log
work: linear

flatten $A : (\alpha\ seq)\ seq \to \alpha\ seq$

ex. $\langle\langle a,b\rangle\ \langle\rangle\ \langle c,d,e\rangle\ \langle f\rangle\ \langle g,h\rangle\rangle$

$\downarrow$

$\langle a,b,c,d,e,f,g,h\rangle$

seq : sum of length of subsequence

fun flatten A =
  reduce <> append A

- log n levels
- linear work on each level

R - result

$W = O(|R| \log |A|)$
$S = O(\log |A|)$

$|R| = \sum_i |A[i]|$

update expensive (linear work, copy)
to parallel imple filter, does many updates
(bulk update w/ inject)
parallelism is an antidote to side effect