Lindsey Bieda and Joe Frambach
Reduction and Parallel Problems
11.18.2011

22. Give a parallel algorithm for the following problem that runs in time O(log n) on an EREW PRAM. The input is a binary tree with n nodes. Assume that each processor has a pointer to a unique node in the tree. The problem is determine the balance factor of each node in the tree. The balance factor of a node is the height of its left subtree minus the height of its right subtree.

This solution relies on the algorithm for expression evaluation, given in the course notes. Initially, we give every edge a function, $f(x) = x$. Then we contract the tree by defining a *cut* operation and applying it to odd-numbered left leaves and odd-numbered right leaves. The leaves are numbered using pointer doubling as discussed in class.
The *cut* operation:

```
   A
    \ x+a
     \
      B
 x+d / \ x+c
    /   \
   D     C
        / \
```

becomes:

```
    A
     \ x+a+d+1-c
      \
       C
      / \
```

and

```
    A
     \ x+a
      \
       B
  x+d / \ x+c
     /   \
    D     C
   / \
```

becomes:

```
    A
     \ x+a+d-1-c
      \
       D
      / \
```

24. Design a parallel algorithm that takes a binary expression tree, where the leaves are Boolean values 0 or 1, and the internal nodes are the three standard logical operations: NOT, OR, and AND. The output should be the value of the expression represented by the tree. Your algorithm should run in O(log n) time on a CREW PRAM with n processors, where n is the number of nodes in the tree. You may assume that each processor initially has a pointer to a unique node in the tree.

This solution is analogous to arithmetic expression tree evaluation, since AND operations are similar to multiplication, and OR operations are similar to addition, and NOT operations are similar to negation.
We apply the standard *cut* procedure as before.

```
    NOT
      \ a(x)
       \
        OR
  d(x) / \ c(x)
      /   \
    AND    1
    / \
```

becomes

```
   NOT
     \ a(d(x) OR c(1))
      \
      AND
      / \
```

The expression on the edge can be immediately simplified.
$d(x)$ OR 1 becomes 1
$d(x)$ OR 0 becomes $d(x)$
$d(x)$ AND 1 becomes $d(x)$
$d(x)$ AND 0 becomes 0