Abhishek Pathak *apathak2*
Rohan Aggarwal *rohanagg*

Project Checkpoint

15-618
November 21, 2019

Summary

The original paper, "A General Technique for Non-blocking Trees" which we will use to implement non-blocking trees, contained a reference to the first paper of chromatic trees, "Chromatic binary search trees - A structure for concurrent rebalancing" by Nurmi and Soisalon-Soininen. This describes the various algorithms used in the implementation of chromatic trees, so for our checkpoint we focused on understanding this paper and implementing a sequential version of chromatic trees based on it.

In the sequential version, we implemented searching, inserstion, deletion and rebalancing. In a chromatic tree, the rebalancing operations are carried out separately and at a later time from the insertions and deletions, thus allowing for easier fine-grained locking and lock-free implementation. Regarding fine-grained locking, we understood from the paper that this implmentation will let us use a finite number of locks for the rebalancing operation, and that insertion and deletion only require one lock. In a nutshell, fine-grained locking would have been tough if trying to implement standard red-black trees, but using the relaxed chromatic tree variant makes it simpler.

On the other hand, this means that the rebalancing operation itself is more complicated when compared to the vanilla red-black trees - the paper describes five different rebalancing operations (with mirrored counterparts), which need to be carried out wherever the rebalancing thread/function finds a relevant violation.

As can be seen in the figures, the cases were numerous and complex. For example, in case 3, it was not obvious that the weights of v and t had to be interchanged, and in cases 2 and 5 it was not obvious that the weights for u and v had to be interchanged. This led to some trouble in implementation until the operation description was dully clear.

We also checked the correctness of our implementation by inserting many random keys into the tree, checking if all of them are searchable, testing the tree for satisfaction of chromatic tree conditions after a rebalance traversal over the tree, then again checking for seachability of nodes, then deleting many keys and rebalancing and doing the searchability and balance checks at each stage.

Also, the paper vaguely described an in-order traversal of the tree which would apply suitable re-balancing operations at each found violating node. It mentioned that one would have to be careful not to re-visit already balanced nodes/subtrees, since sub-trees can change position in rebalancing operations. This would be very difficult to implement. However, based on the lemmas and theorems in the paper, we realized that a pre-order traversal would have the same effect, and would be simpler.

The next step now is to implement fine-grained locking, which after this first step will not be very time-taking. Then, we will spend more time on implementing lock-free chromatic trees, which will be challenging but will build on our basic foundation of sequential chromatic trees.

It was a bit difficult to implement this paper since there were no formal algorithm descriptions or pseudocode, only high-level descriptions. With a total of about 600 lines consisting

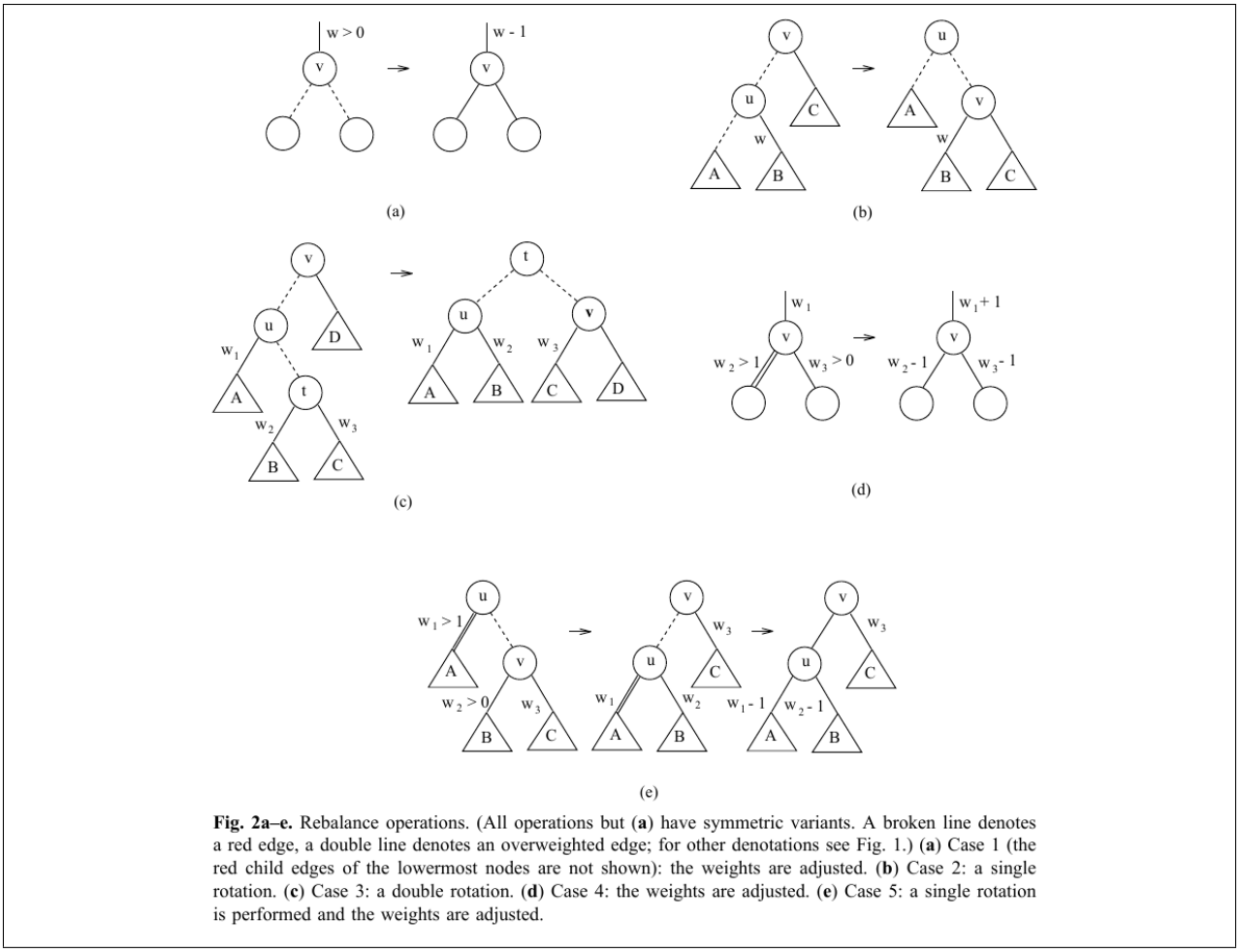

Fig. 2a–e. Rebalance operations. (All operations but (**a**) have symmetric variants. A broken line denotes a red edge, a double line denotes an overweighted edge; for other denotations see Fig. 1.) (**a**) Case 1 (the red child edges of the lowermost nodes are not shown): the weights are adjusted. (**b**) Case 2: a single rotation. (**c**) Case 3: a double rotation. (**d**) Case 4: the weights are adjusted. (**e**) Case 5: a single rotation is performed and the weights are adjusted.

Figure 1: Rebalance operations

of tree operations and correctness checking code with complex pointer operations in which debugging was quite tough, we wound up our checkpoint progress by successfully implementing a sequential chromatic tree. We invested effort in understanding the proofs and intuition supplied in this paper, so we may implement fine-grained locking and then non-blocking logic easily in the future.

In the coming weeks we plan to first finish fine-grained locking, and then do non-blocking chromatic trees.

November 23 - Finish fine-grained locking

November 27 - Write routines for non-blocking tree

November 30 - Write atomic operations used in routines

December 04 - Finish implementation with multiple threads

December 07 - Check correctness and measure performance