# Projects in Multi-Core Synchronization
# Tree-Based Combining for Concurrent Data Structures

**The project is offered by Prof. Hagit Attiya.** To register, email `hagit@cs.technion.ac.il`.

*Combining* is an approach for reducing the synchronization overhead when implementing concurrent data structure. **This project aims to implement a new combining mechanism, called *TreeCombine*, and evaluate its performance on several data structures.**

A concurrent data structure is like a regular data structure, but it allows several threads to apply operations to it, concurrently. An *implementation* of a concurrent data structure gives procedures (code) for executing these operations in terms of *primitive* operations, like reads, writes, Compare&Swap, Fetch&Add, Test&Set, and possibly, lock acquisition and release.

The implementation must be *linearizable*, which is a condition similar to (strict) serilizability, namely, the operations must *appear* to execute in some sequential order. Specifically, there is an serial execution of the same operations with the same results, in which operations that *do not overlap* preserve their order. (The formal definition is more complicated, but is not needed for the project.)

There are many ways to design such implementations: Some rely on *coarse locking*, i.e., protecting the whole data structure with a lock, while others use *fine-grain locking*, i.e., protecting smaller parts of the data structure with locks, and acquiring / releasing them carefully.

This project considers a different approach to implementing concurrent data structures, called *combining*: The data structure is protected as in coarse locking, however, the thread that acquires the lock on the whole data structure, also performs operations of other waiting threads, and then notifies them. This technique is beneficial when operations are simple, and the cost of applying them is small in comparison with the cost of obtaining the lock.

The key to good combining is (efficiently) knowing which threads have pending operations, and (efficiently) notifying them when their operations are done. In this project you will check a new mechanism that is inspired by a new (mutex) lock implementation, TreeMutex.

The project consists of two tasks:

1. Implement TreeCombine in C, including easily configurable options to pick operations to combine.

2. Provide a framework to compare with alternative combining approaches, and run benchmarks.

The final results should be obtained by running on a multi-core machine in the computer lab which we will provide access to. For coding and debugging, you can use your own machine.

**Required background:** *operating systems*; *parallel and distributed programming* is a plus.