Projects in Multi-Core Synchronization
Tree-Based Mutex Lock

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

This project aims to implement a new (mutex) lock, called TreeMutex, and evaluate its performance within a new sandbox for mutex algorithms. TreeMutex, a recent tree-based lock, is depicted in the figure, and outlined next.

The algorithm uses a binary tree so that leaf $L_p$, for $p \in \{0, \ldots, n - 1\}$, is statically assigned to thread $p$. To obtain the lock, thread $p$ traverses up the path from $L_p$ to the root, writing its identifier to each node along this path. Then, $p$ performs a single memory barrier to ensure that these writes become visible, and then attempts to capture the lock using a CAS operation. If the CAS succeeds, then $p$ has obtained the lock.

The algorithm uses a promotion mechanism. This mechanism allows a thread performing its exit section to facilitate the entry of other threads whose identifiers it reads along the path from the root to its leaf. Identifiers of promoted threads that did not yet enter the critical section are stored in a FIFO queue named $promQ$. Threads apply operations to $promQ$ before they release the lock, hence its implementation is not required to support concurrent access.

When exiting the critical section, a thread descends down the path from the root to its leaf, reading the identifiers written at every internal node along this path and its child nodes. Thread $p$ promotes any process whose identifier it reads, by adding it to the promotion queue if it is not already there. Finally, $p$ checks the promotion queue: if empty, $p$ releases the lock; otherwise, $p$ dequeues the first process from the promotion queue, and hands over the lock to it.

The project includes three tasks:

1. Implement TreeMutex in C.
2. Install the mutex evaluation sandbox, and compare TreeMutex within it.
3. Find performance optimizations for TreeMutex.

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.