



SUPERTECH SEMINAR

Date: Tuesday, June 8 2004
Refreshments: 3:00 p.m.
Talk: 3:15 p.m.
Place: 32-D507

“On-the-Fly Maintenance of Series-Parallel Relationships in Fork-Join Multithreaded Programs”

by Jeremy Fineman

Abstract: A key capability of data-race detectors is to determine whether one thread executes logically in parallel with another or whether the threads must operate in series. This talk provides two algorithms, one serial and one parallel, to maintain series-parallel (*SP*) relationships “on the fly” for fork-join multithreaded programs. The serial *SP-order* algorithm runs in $O(1)$ amortized time per operation. In contrast, the previously best algorithm requires a time per operation that is proportional to Tarjan’s functional inverse of Ackermann’s function. *SP-order* employs an order-maintenance data structure that allows us to implement a more efficient “English-Hebrew” labeling scheme than was used in earlier race detectors, which immediately yields an improved determinacy-race detector. In particular, any fork-join program running in T_1 time on a single processor can be checked on the fly for determinacy races in $O(T_1)$ time. Corresponding improved bounds can also be obtained for more sophisticated data-race detectors, for example, those that use locks.

By combining *SP-order* with Feng and Leiserson’s serial *SP-bags* algorithm, we obtain a parallel *SP-maintenance* algorithm, called *SP-hybrid*. Suppose that a fork-join program has n threads, T_1 work, and a critical-path length of T_∞ . When executed on P processors, we prove that *SP-hybrid* runs in $O((T_1/P + PT_\infty) \lg n)$ expected time. To understand this bound, consider that the original program obtains linear speed-up over a 1-processor execution when $P = O(T_1/T_\infty)$. In contrast, *SP-hybrid* obtains linear speed-up when $P = O(\sqrt{T_1/T_\infty})$, but the work is increased by a factor of $O(\lg n)$.

This research represents joint work with Michael A. Bender, Seth Gilbert, and Charles E. Leiserson.

Host: Charles E. Leiserson