Paper summary

Multithreaded programs are difficult to debug and error-prone hence programmers resist using threads despite their potential benefits. It is easier to make synchronization mistakes which can lead to data race conditions. Data race errors are timing-dependent and very difficult to find. Race detectors based on the "happens-before-relationship" fails to capture all the race condition scenario. A potential tool called "Eraser" is suggested to prevent the data race condition, which automates the problem of detecting possible data races. It takes an unmodified binary program as input and outputs a new binary that implements the lockset algorithm. It looks for unprotected access to shared variables. According to the lockset algorithm, the shared variable is protected by locks. The locks assigned to which variable is inferred from the execution history. It examines the collection of locks the program already has and the collection of locks it already has. If the already has locks is less, shared variables are not well protected. The potential pitfall with Eraser is that it can cause false alarm and have three categories: Memory reuse, private locks, benign race. The paper suggests its potential application in deadlock detection. Also, Eraser has a slowdown by a factor of 10 to 30. The testing is performed on AltaVista, Petal, and undergraduate multithreaded programming projects. Eraser also introduces program annotations that overcome false alarms. Hence the implementation of the Lockset method used in Eraser is promising.

Strengths

1. Easy to follow and painless implementation for the lockset algorithm.
2. Language independent since takes unmodified binary as input.
3. Better than existing data race implementation such as "happens-before."
4. Using it for deadlock detection is an interesting proposal.

Weaknesses

1. Increase in binary size.
2. Slowdown factor: 10 to 30.
3. False-positive are still possible.
4. Multiple locks were not implemented.

Comments for author

Is it necessary to modify the binary? Can there be alternatives where the code base is not changed? What can be the potential benefits and drawbacks of this approach if used?