# Non-race concurrency bug detection through order-sensitive critical sections

## Non-Race & Order-sensitive critical section

 A pair of critical section that can lead to non-deterministic shared memory state depending on the order in which they execute.

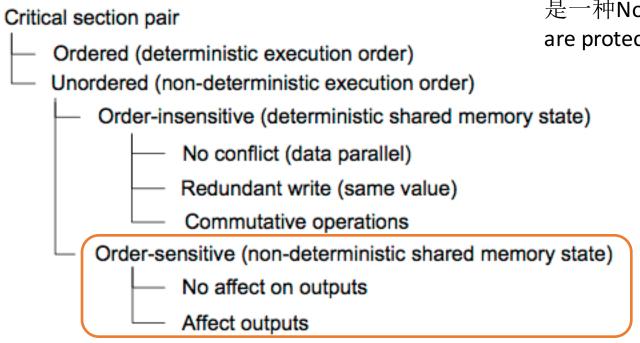
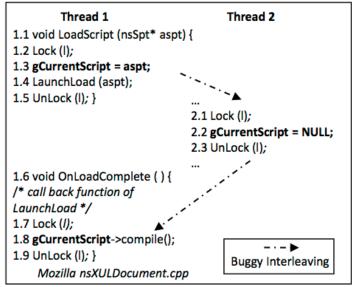


Figure 2: Types of critical section pairs.

是一种Non-race bug,因为memory access at same location are protected by mutex, 所以并没有存在data-race



(c) Order-sensitive critical section pair. Non-race atomicity violation bug (Mozilla-1) [11].

#### How to detect OSCS

• Filter our ordered and order-insensitive critical sections.

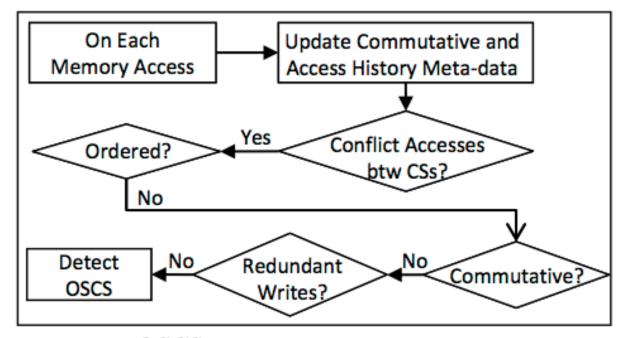


Figure 6: OSCS operations on every memory access.

Thread 1 Thread 2 1.1 void LoadScript (nsSpt\* aspt) { 1.2 Lock (I); 1.3 gCurrentScript = aspt; 1.4 LaunchLoad (aspt); 1.5 UnLock (I); } 2.1 Lock (I); 2.2 gCurrentScript = NULL; 2.3 UnLock (I); 1.6 void OnLoadComplete () { /\* call back function of LaunchLoad \*/ 1.7 Lock (*I*); 1.8 gCurrentScript->compile(); 1.9 UnLock (I); } **Buggy Interleaving** Mozilla nsXULDocument.cpp

(c) Order-sensitive critical section pair. Non-race atomicity violation bug (Mozilla-1) [11].

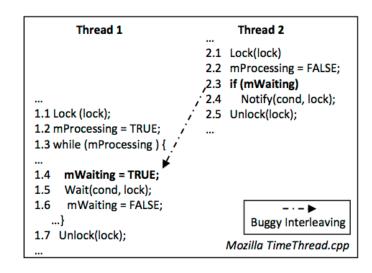


Figure 4: An ordering violation bug example (KB3) [24].

Commutative operation is determined by read-write sequence

## Evaluation

- Concurrency bugs
  - Atomicity violation
  - Ordering violation
  - Multi-variable bugs
- False positive
- False negative

# Synchronization

- Ordering synchronization operations
  - Barrier
  - Wait-signal pair
- Mutex synchronization
  - Mutex
  - semaphore