Stefan Savage, Michael Burrows, Greg Nelson, Patrick Sobalvarro, and Thomas Anderson
Eraser: A Dynamic Data Race Detector for Multithreaded Programs
Transactions on Computer Systems, Vol. 15, No. 4, November 1997, Pages 391–411.

1. Consider the following first implementation of the lockset algorithm:

```
Let locks held(t) be the set of locks held by thread t.

For each v, initialize C(v) to the set of all locks.
On each access to v by thread t,
       set C(v) := C(v) ∩ locks held(t);
       if C(v)= { }, then issue a warning.
```

For the following example program execution, show the values of locks_held and C(A) for variable A over time. Will a data race detected? If so, when?

CV = everything candidate set

Lock(i)
Lock(m);
A++;
Unlock(m);

locks_held: i, m CV = {i, m}

Lock(m);
A--;
Unlock(m);
Unlock(i);

locks_held i, m CV = {i, m}

Lock(i);
Lock(j);
A++;
Unlock(i);
Unlock(j);

locks_held i, j CV = {i}

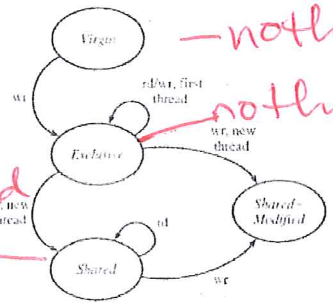
Lock(i);
B++;
Unlock(i);

Lock(m);
A--;
Unlock(m);

locks_held m CV = ~~[illegible]~~ null
→ RACE

(Optional: Can you come up with a better code example?)

2. Consider the following new states for a variable to handle initialization and read-sharing. How is C(v) updated when a variable v is in each state? What checks are made for a variable in each state?



—nothing

nothing

C(V) updated (no notes)

C(V) updates races reported

For the following code sequence, what is the state of each variable A and B over time? Over time, what are the values of locks_held and C(A) and C(B)? Would a race be reported? Can any of these locks be removed and still be valid?

Thread 1 A,B virgin	**Thread 2** A,B virgin
// code not involving A or B	// code not involving A or B;
A = 1; B = A+5; exclusive	
A++; exclusive	B now shared modified
Lock(n); B++; Unlock(n); C(B) = {n}	Lock(n); B--; Unlock(n); locks-held → n C(B) = {n}

	Lock(m);
	If (A == 2) {
	// do something(); C(A) = {m}
	}
	Unlock(m); A is shared

A is shared
Lock(m); C(A) = {m}
If (A == 2) {
// do something();
Unlock(m);	

(Optional: Can you come up with a better code example?)

- okay
- Lock m can be removed (if this all that happens) when only in shared state

3. Consider the following version to handle read-write locks:

```
Let locks held(t) be the set of locks held in any mode by thread t.
Let write locks held(t) be the set of locks held in write mode by thread t.
For each v, initialize C(v) to the set of all locks.
On each read of v by thread t,
    set C(v) := C(v) ∩ locks held(t);
    if C(v) := { }, then issue a warning.
On each write of v by thread t,
    set C(v) := C(v) ∩ write locks held(t);
    if C(v) = { }, then issue a warning.
```

For the following code sequence, what is the state of each variable A and B over time? Over time, what are the values of locks_held, write_locks_held, and C(A) and C(B)? Would any races be reported (if so, where)? Can any of these locks be removed and still be valid? Assume if Lock is given the parameter WRITER it is a write lock, otherwise it is only a read lock.

(Optional: Can you come up with a better code example?)

Thread 1	**Thread 2**	**Thread 3**
A = 1; B = A+5; C=B+1;		
		Lock(e); Lock(g); If (C == ...) do something(); Unlock(g); Unlock(e);
Lock(k); Lock(j); If (A == ...) do_something(); Unlock(j); Unlock(k);		
	Lock(e); Lock(f); If (C == ...) do something(); Unlock(f); Unlock(e);	
Lock(d); If (B == ...) do something(); Unlock(d);		
		Lock(m); Lock(n); If (A == ...) do_something(); Unlock(n); Unlock(m);
Lock(e, WRITER); Lock(c, WRITER); C++; Unlock(c); Unlock(e);	Lock(j); Lock(m); If (A == ...) do_something(); Unlock(m); Unlock(j);	Lock(g, WRITER) Lock(d); B++; Unlock(d); Unlock(g);
	Lock(d); If (B == ...) do something(); Unlock(d);	