TIMO SCHNEIDER <TIMOS@INF.ETHZ.CH>

DPHPC Recitation Session 4
Linearizability

Open Questions from previous sessions:

- Why do we need a BusRdX* message if we already have a BusRdX?
  - For correctness, we do not need it – MESI works fine if we just use BusRdX.
  - BusRdX implies: “Fetch this line from memory!”.
  - But if we have a line in S state and the processor writes to it we don't need to fetch this line again, so we can save some memory bandwidth by not fetching it!
  - Hence, we have a seperate message for that.

Linearizability

- Why do we need it?
  - Suppose you have a shared variable and you observe the following:

    A writes 1, B writes 2, B reads 1
  - In sequential consistency terms:

    A: w(1)

    B: w(2); r():1
  - Is this sequentially consistent?
    - Yes!
    - But probably not what we want

      → need a new formalism!

Linearizability Terms

- **Explain the term**
  - History
  - Thread projection
  - Sequential history
    - Each method call is immediately followed by its response
  - Concurrent history
    - Opposite of sequential history: Method calls can overlap!
  - Well-formed history
    - Per thread projection is sequential

Linearizability Terms

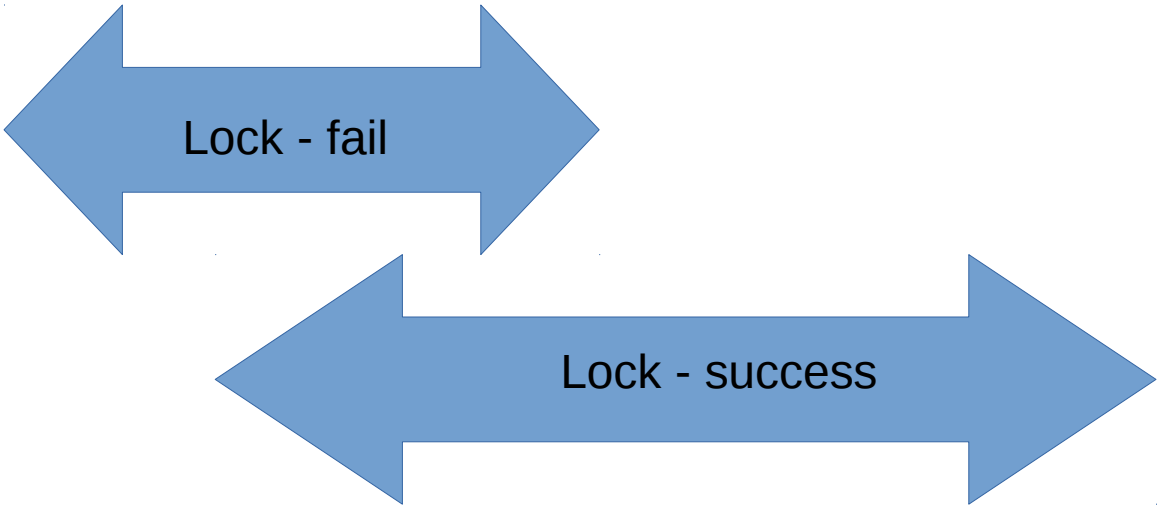
- **Explain the term**
  - Equivalent histories
    - Per thread projections are the same
  - Legal history
    - For every object x, H|x conforms with the specification of x
  - Precedence
    - M1 precedes M2 iff M1 response precedes M2 invocation
  - Overlap
    - Opposite of Precedence

Linearizability

- A history is linearizable iff
  - It can be turned into a legal sequential history H' by
    - Dropping pending invocations
    - Reordering events while observing the rule
      - If a response preceded an invocation in H it must precede it in H'

Linearizability – Example 1

- **Graphical Example**



Linearizability – Example 1

- **Same example in written form:**

  A: l.lock()

  B: l.lock()

  A: l:fail

  B: l:success

- **We can reorder this as**

  A: l.lock()

  A: l:fail

  B: l.lock()

  B: l:success

- **No response preceded an invocation in H, so we don't need to worry about "illegal" reordering**
- **But it does not conform to the specification of how a lock should behave!**

Linearizability – Example 1

- **Same example in written form:**

  A: l.lock()

  B: l.lock()

  A: l:fail

  B: l:success

- **We can reorder this as**

  B: l.lock()

  B: l:success

  A: l.lock()

  A: l:fail

- **Check: No response was moved before an invocation which it originally preceded! (Since all responses came after all invocations)**
- **And it conforms to the specification! → History is linearizable**

Linearizability – Example 2

- FIFO queue with operations enq(x)/void and deq()/x

A: r.enq(x)
A: r:void
B: r.enq(y)
A: r.deq()
B: r:void
A: r:y

What are the possible reordered histories?

Is any of them legal?

Linearizability – Quiz

If H|p and H|q for threads p and q is linearizable, H is linearizable?

If H|p and H|q for threads p and q is sequentially consistent, H is linearizable?

If H|x and H|y for objects x and y is linearizable, H is linearizable?

(Assume the history only contains the given two threads/objects)