Sri Hari Malla - CS19BTECH11039
Prof : Dr. Sathya Peri
Parallel and Concurrent Programming
Indian Institute of Technology Hyderabad

THEORY ASSIGNMENT - III

Sri Hari Malla - CS19BTECH11039

February 24, 2022

1 Given figures, justify if they are quiescently and sequentially consistent along with linearizability

1.1 fig 3.13:

Clearly from the figure,the Order of execution is

- r.write(1)
- r.read(1)
- r.write(2)
- r.read(2)

These calls are perfectly linearizable as they are compositional and effective. As linearizability implies sequential consistency, it is sequentially consistent and quiescently consistent.

1.2 fig 3.14:

clearly, from the figure the order of execution is

- r.write(1)
- r.read(1)
- r.write(2)
- r.read(1)

These events are also compositional and instantaneously effective, making them linearizable and thus quiescently and sequentially consistent.

2 FIFO Queue linearizable?

Given FIFO queue implementation stores items in array items for simplicity sake, let's consider it is of unbounded size. It has two atomic integer fields which use `compareAndSet` instruction. Head is the index of the next slot in which to remove the item and similarly tail is the index of the next place in which to place the next new item.

Given this, let's consider if two threads are invoked and the first thread calls enqueue(x). It continues to go until it has to set the first item but it doesn't finish. Then the second thread calls the enqueue(y) and it completes then it calls dequeue. Now it tries to remove the first item which is impossible as head will be pointing to null and throws an exception.

So, with counter example we can tell that this is non linearizable.

3 Safe Boolean MRSW:

We learnt that if a register should be regular iff a read call overlaps with write call, then it should return a value either old or new.

- If read call doesn't overlap with the write call, then it is obvious that we get a value which is the last updated.
- Else if a read call overlaps with write call, let's say if one thread reads at a field and another thread overlaps the write call with it, if writer writes before the reader reads, the reader returns new value.
- In the other case as it is a regular safe boolean mrsw register it returns a new value or old value based on the writer.

So if we replace the safe Boolean SRSW register array with an array of regular M-valued SRSW registers, then the construction yields a regular M-valued MRSW register. The statement is true.

4 Regular Boolean MRSW:

We learnt that if a register should be regular iff a read call overlaps with write call, then it should return a value either old or new.

As we have seen in the above question, the safe boolean mrsw gives only either new or old values and that is for sure.

But in this case it is a m valued register, if read call overlaps with another write call, then it could return any value not definitely the old or the new value which are present in that field.

If we replace the safe Boolean MRSW register with a safe M-valued MRSW register, then the construction doesn't yield a regular M-valued MRSW register. So the given statement is false.

5 Peterson's two thread mutual exclusion algo and shared atomic flag registers:

Let's consider a case where a write operation to a shared atomic flag by one of the many threads overlaps with the other threads whilst reading in the while loop.

If thread t1 is writing flag f in lock method, and that time another thread t2 reading the same block might not read the old value stored in it which never gets out of the loop. Considering, this it would've set itself to be victim which would've allowed the thread t2 to acquire the lock.

Thread t1 would then get caught in the while loop and then only one thread would get in to the lock as expected. The write to flag in unlock method concludes that a read operation could change to new value compared to the old value which was read. There is a case that this wouldn't happen. If the thread performing a read operation reads a value which is in a while loop and then breaks, making the second read non permissible. If we look at victim value and if this value changes then on the first read of the fresh value then the while loop breaks not permitting the second read operation.

So, the peterson's two thread mutual exclusion algorithm works if the shared atomic flag registers are replaced by regular registers.

LaTeX generated document

THE END