EE 382C/361C: Multicore Computing **Fall 2016**

Lecture 11: September 29

Lecturer: Vijay Garg *Scribe: Shruti Subramanian*

11.1 Introduction

Exam Overview, Sequential Consistency, Linearizability, Composability,

11.2 Exam 1 Overview

Exam in class on Thursday 10/6. This is the last day of material for Exam 1. The exam is closed book.

11.2.1 What to Know

The puzzles handout which covers basic parallel algorithms.

Know the meaning of the basic constructs for OpenMP.

Syntax is not important and psudeocode is okay EXCEPT for Condition Variables/Semaphores/Reentrant Locks

HW3 has been posted. The first 3 questions are relevant to the first exam.

Review: Lectures + Handouts + Chapters 1 - 4 + Assignments

11.2.2 How to Study

1. Take sample exam posted online in 75 minutes.
2. Work with friends and discuss practice exam.
3. Check with the solutions posted online.

11.3 Overview

Rememeber that $(H,<_H)$ is the concurrent history.

Sequential history $(S,<_S)$: A history is $(S,<_S)$ is sequential if $<_S$ is total order (total order is from last time)

11.4 Equivalence

Two histories are equivalent if they are composed of the same operations. The processes in both concurrent

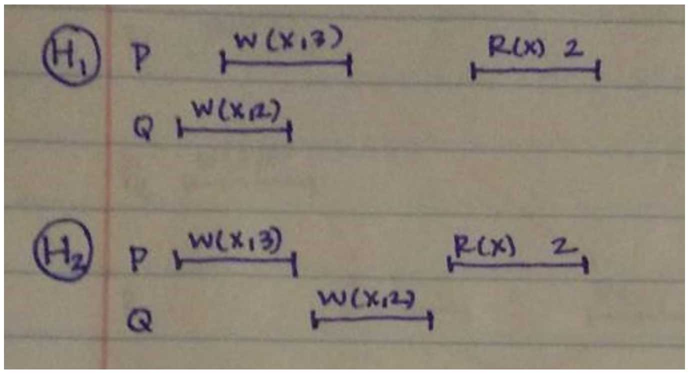

Figure 11.1: Two equivalent histories

histories are equivalent, but not identical since (H,$<_H$) is not the same. In this example the three operations are permuted.

11.5 Lamports Sequential Consistency

A concurrent history (H,$<_H$) is sequentially consistent if there exists a legal sequential history S such that: 1. S is equivalent to H 2. S preserves the process order in (H,$<_H$)

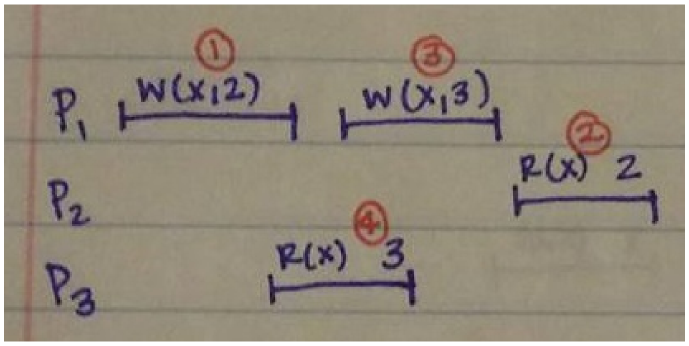

Figure 11.2: A sample sequential consistency

Order across processes do not matter. Both histories from figure 11.1 are sequentially consistent.

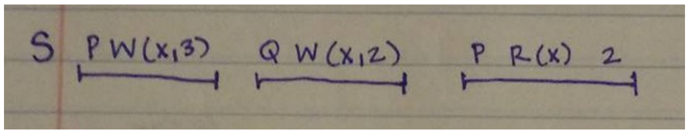

Figure 11.3: Sequential consistency of figure 11.1

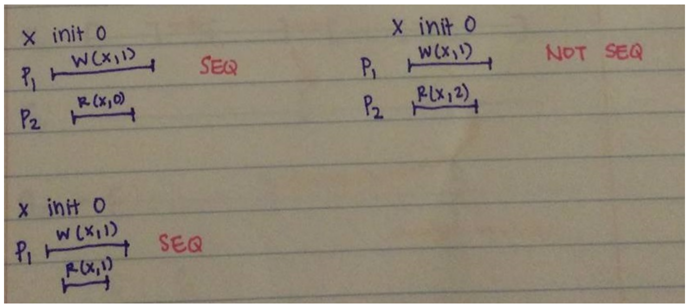

Figure 11.4: Sequential consistency example 1

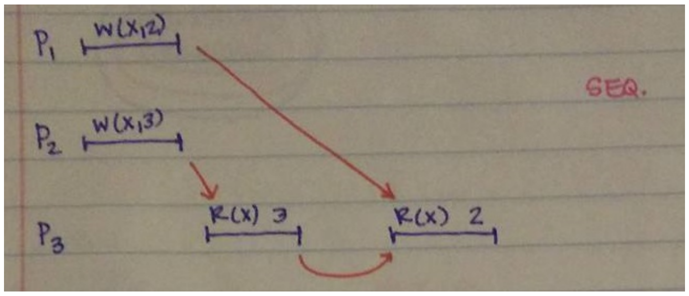

Figure 11.5: Sequential consistency example 2

11.5.1 Not Sequential Consistency

To show that it is not sequentially consistent, you show there is not any way to be sequentially consistent by either:

1. The brute force method where you try out every combination for a case analysis

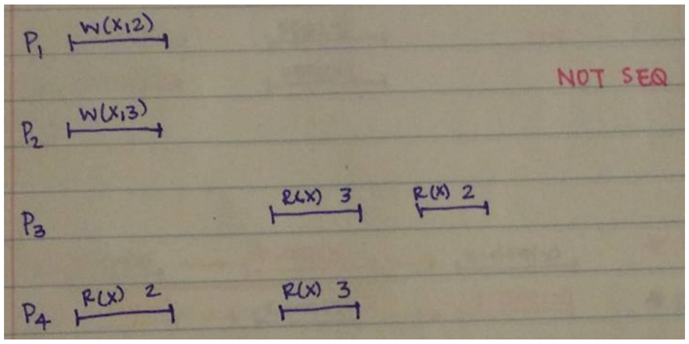

Figure 11.6: Not sequentially consistent

2. Check if there is a cycle such that the order cannot be preserved.

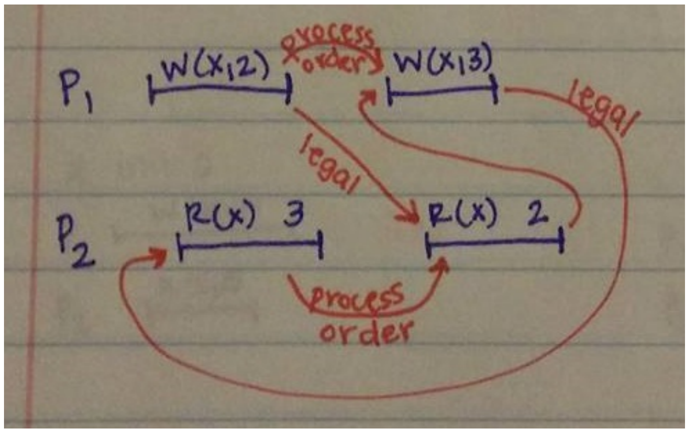

Figure 11.7: Not sequentially consistent cycle

11.5.2 Unfinished history

Extend the unfinished by finishing up the history for a complete history that it is sequentially consistent

11.6 Henlihy and Wings Consistency

This correctness is stronger than sequential consistency and thus better for software purposes. A history $(H,<_H)$ is linearizable/atomically consistent if there exists a legal sequential history S such that: 1. S is equivalent to H 2. S preserves $<_H$(the real time order)

11.6.1 Linearizability and Sequential Consistency

Theorem: H is LIN $\rightarrow$ H is SEQ

Proof: $e <_{\text{process order}} f \rightarrow e <_H f$

The process-order is realtime order on the same process.

11.6.2 Linearization Point

That gives legality without operations. Linearization points should not match and must have some order.

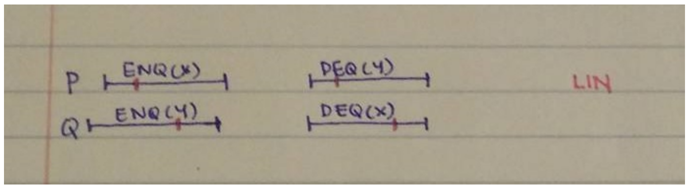

Figure 11.8: A linearizable history with linearization points

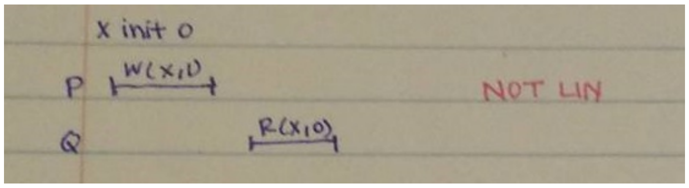

Figure 11.9: A non-linearizable history

S1 = enq(x), eng(y) deq(x) deq(y) S1 = enq(y), eng(x) deq(y) deq(x)

Theorem: H is linearizable iff there exists linearization points for all operations such that the resulting sequence is legal and preserves the real time order.

11.7 Linearizability and Sequential Consistency

Suppose we have a legal program for queues and/or stacks and we join them.

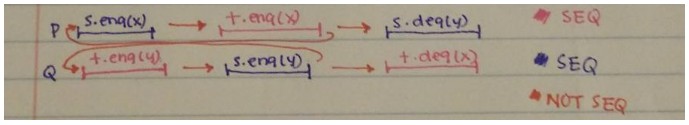

Figure 11.10: Illegal combination of two legal programs

Defn: A consistency condition CONSISTENCY satisfies composability/locatlity if for all x: H | x satisfies consistency $\longleftrightarrow$ H satisfies consistency

11.7.1 Linearizability and Composability

Theorem: LIN satisfies COMPOSABILITY

Proof:

Claim: The graph stays acyclic. Two types of arrows e $<_H$ f resp(e) occurred before inv(f) f $<_n$ g resp(f)

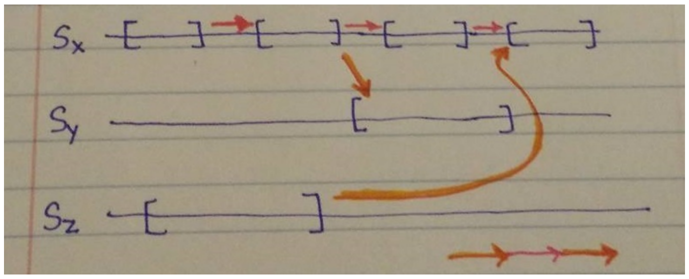

Figure 11.11: Linearizability cycle

occurred before inv(g) g $<_H$ h resp(g) occurred before inv(h) By transitivity $\rightarrow$ resp(e) occurred before inv(h) if any cycle $\rightarrow$ then to begin with thus no cycle Thus, with a cycle we are not sequentially consistent.

11.8 Other types of consistency

Causal consistency, FIFO consistency, Entry Consistency

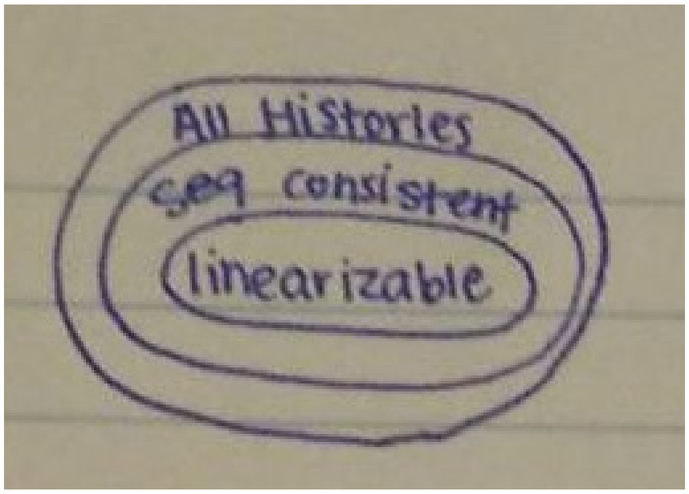

Figure 11.12: Consistency circles

11.8.1 Consistency Tradeoffs

Stronger consistency gaurantees ease of programs, however this causes a loss of performance.

References

[1] VIJAY K GARG, Multicore Computing, The University of Texas at Austin. September 2016.