Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

HS 2012

S. Welten / Prof. R. Wattenhofer

Distributed Systems Part II

Solution to Exercise Sheet 8

1 Consistency Models

a) Sequential Consistency → Causal Consistency
In sequential consistency, all writes must be seen in the same order by all processes. In causal consistency, causally related writes must be seen in the same order. As the causally related writes form a subset of all writes, this requirement for causal consistency is certainly fulfilled if the requirement for sequential consistency is fulfilled.

b) Causal Consistency ← Linearizability
Linearizability implies sequential consistency, and thus – using the result from subtask a) – also causal consistency. To see that linearizability implies sequential consistency, we can look at the partial orders (real-time partial order $<_r$ vs. client partial order $<_c$) that have to be fulfilled by the two consistency models. As $<_r$ implies $<_c$ linearizability implies sequential consistency.

c) Linearizability → Read-your-Writes Consistency
If an execution is linearizable, the total order on the data type agrees with the one on each client. Thus, the own writes (or a newer value) will always be read.

d) Read-your-writes Consistency ← Causal Consistency
Causal consistency implies read-your-writes consistency because a read operation of a process has a causal dependency on the write operation that changed the value the last time. If that causality order is respected the process always reads the lastest value it has written in the variable.

2 Library

a)
- **Linearizability:** The "execution" is not linearizable because the first read operation does not read the value of the write operation that was executed directly before it. There is no linearization that leads to the same results of the read operations.
- **Sequential consistency:** The "execution" is not sequentially consistent because the client partial order requires $w(0) < w(1) < r(0))$ and $r(0)$ requires that no write operation occurs between $w(0)$ and $r(0)$ (because it reads the value writte by $w(0)$).
- **Monotonic Read Consistency:** The "execution" is monotonic read consistent because the second read operation reads a value that was written later than the value read by the first read operation.
- **Read-your-Writes-Consistency:** The "execution" is not read-your-writes consistent because $r(0)$ that is executed directly after $w(1)$ does not read the value written by $w(1)$.

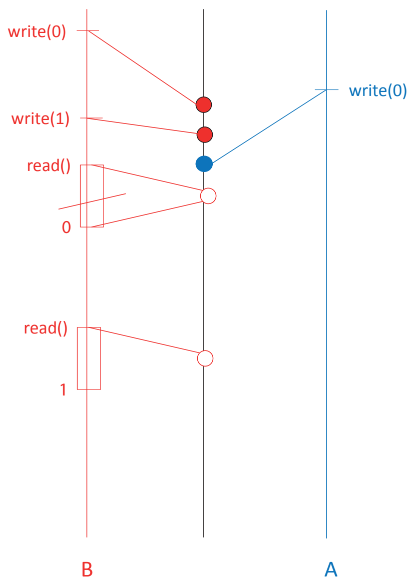

Figure 1: A linearizable library "execution" when other people (A) might have been in the library at the same time as Barbara.

- **Causal consistency:** The "execution" is causally consistent (see subtask c).

b)

- **Linearizability:** The "execution" could be linearizable if for example someone had borrowed the book after she gave it back and before she rechecked the index (see Figure 1).
- **Sequential consistency:** The "execution" could be sequentially consistent because the system could be linearizable (and linearizable implies sequential consistency)
- **Monotonic Read Consistency:** The "execution" could be monotonic read consistent because the system could be linearizable (and linearizable implies monotonic read consistency)
- **Read-your-Writes-Consistency:** The "execution" could be read-your-writes consistent because the system could be linearizable (and linearizable implies read your writes consistency)
- **Causal consistency:** The "execution" could be causally consistent because the system could be linearizable (and linearizable implies causal consistency)

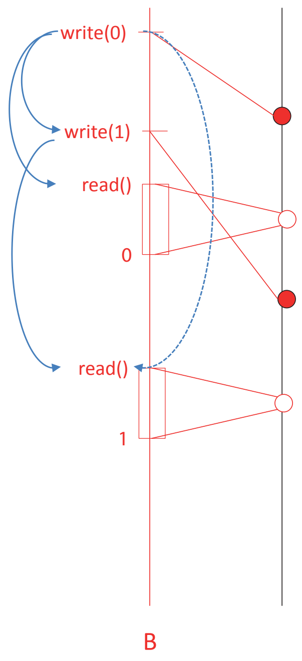

Figure 2: The library "execution" with causal dependencies.

c) Figure 2 shows the library "execution" (red) with the causal dependencies (blue), according to the definition of causal dependency in slide 18 in part 3 of chapter 7. Note that the blue dotted line on the right indicates a dependency that is induced by the transitive closure of the definition.

- $write(0)$ Barbara borrows the book from the library.
- $write(1)$ Barbara takes the book back.
- $read = 0$ Barbara checks the index.
- $read = 1$ Barbara checks the index again.