Dr. Annette Bieniusa
M.Sc. Peter Zeller
M.Sc. Mathias Weber

TU Kaiserslautern
Fachbereich Informatik
AG Softwaretechnik

Exercise 5: Programming Distributed Systems (SS 2018)

This sheet is discussed in the exercise on Thursday, June 14.

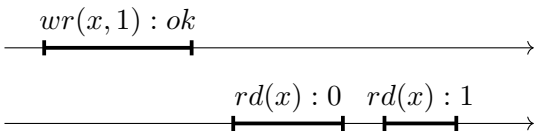
1 Consistency Models

For the following executions, decide which of the consistency models Linearizability (L), SequentialConsistency (SC), CausalConsistency (CC), and BasicEventualConsistency (EC) would allow the execution. Multiple or no options might apply. Also state how confident you are in each answer by giving a value between 0% and 100%.

Each execution uses read (rd) and write (wr) operations on registers x and y. The initial value of registers is 0.

For all subtasks here: If an execution is linearizable, it is also sequentially consistent. If it is sequentially consistent, it is also causally consistent. And if it is causally consistent, it is also eventually consistent.

a)

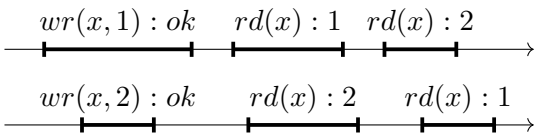


L	SC	CC	EC	Confidence (%)
☐	☐	☐	☐	

It is sequentially consistent. The order $rd(x) : 0, wr(x,1) : o, rd(x) : 1$ explains the results and respects the order on each process.

It is not linearizable, because $wr(x,1)$ happens before reading 0, so it does not satisfy the realtime constraint.

b)

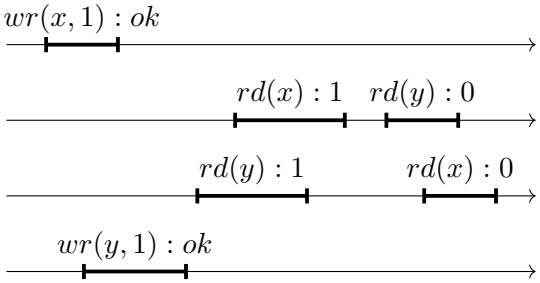


L	SC	CC	EC	Confidence (%)
☐	☐	☐	☐	

This is just eventually consistent.

It is not causally consistent, because ...

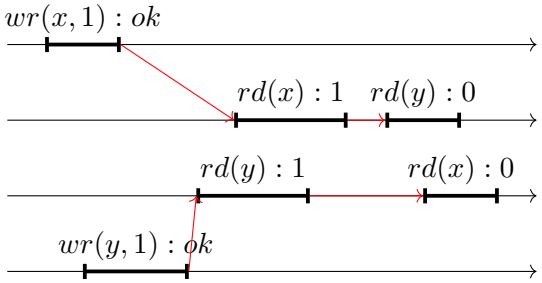
c)



L	SC	CC	EC	Confidence (%)
☐	☐	☐	☐	

This satisfies causal consistency: the write to x is only visible to process 1 and 2 and the write to y is only visible to process 3 and 4, which is allowed by causal consistency.

We can draw the happens-before relation (omitting transitive edges) to explain the results:

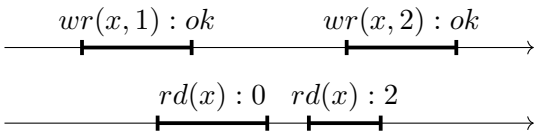


The execution is not sequentially consistent: For it to be sequentially consistent there must be a total ordering of operations that explains the results. However, we have the following restrictions:

- $wr(x,1)$ must be before $rd(x) : 1$ because of the returned value
- $wr(y,1)$ must be before $rd(y) : 1$ because of the returned value
- $rd(y) : 0$ must be before $wr(y,1)$ because of the returned value
- $rd(x) : 0$ must be before $wr(x,1)$ because of the returned value
- $rd(x) : 1$ must be before $rd(y) : 0$ because of session order (ReadMyWrites)
- $rd(y) : 1$ must be before $rd(x) : 0$ because of session order (ReadMyWrites)

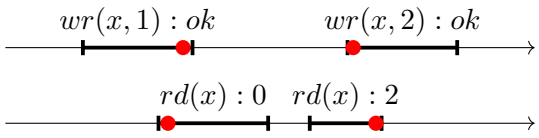
Satisfying all of these constraints is obviously not possible (if you don't believe me, you can ask a theorem prover `https://rise4fun.com/Z3/J7Cf`).

d)



L	SC	CC	EC	Confidence (%)
☐	☐	☐	☐	

This is linearizable. To explain it, we can mark the linearization points in the execution (here as red dots), so that the result is explained.

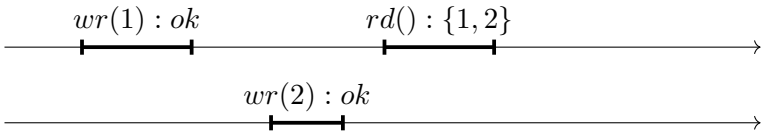


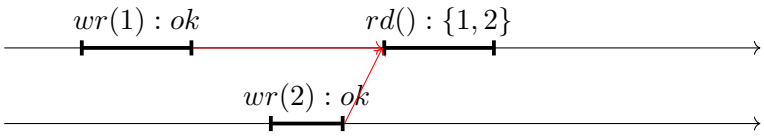
2 Concrete and abstract executions

Consider the following concrete executions with operations on a Multi-Value Register object and provide an abstract execution that explains, why the execution is allowed under causal consistency.

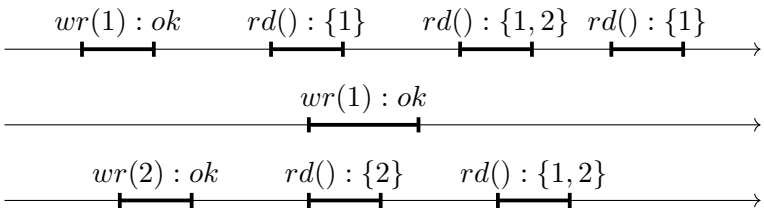
Hint: To give an abstract execution, you need to specify the relation vis, which you can visualize in the pictures below by drawing arrows between operations.

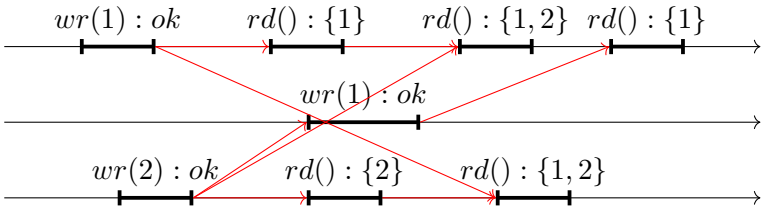
a)





b)





3 CRDTs

Consider the Add-wins Set (Observed-remove Set, non-optimized version) presented in the lecture on CRDTs on slide 30. Prove that the downstream effect of an add-operation commutes with the effect of a concurrent remove-operation.

First add, then remove:

$$E := E \cup \{(e,n)\} \setminus T$$
$$E := E \setminus R$$
$$T := T \cup R$$

$\Rightarrow E_1' = (E \cup \{(e,n)\} \setminus T) \setminus R$

$\Rightarrow R_1' = T \cup R$

First remove, then add:

$$\begin{aligned} E &:= E \setminus R \\ T &:= T \cup R \\ E &:= E \cup \{(e, n)\} \setminus T \end{aligned}$$

$\Rightarrow E_2' = (E \setminus R) \cup \{(e, n)\} \setminus T$

$\Rightarrow R_2' = T \cup R$

Obviously, $R_1' = R_2'$. The interesting case is to show $E_1' = E_2'$:

$$\begin{aligned} E_2' &= (E \setminus R) \cup \{(e, n)\} \setminus T \\ &= (E \cup \{(e, n)\}) \setminus R \setminus T && \text{Because } (e, n) \text{ contains unique } n, \text{ which cannot appear in } R \\ &= (E \cup \{(e, n)\}) \setminus T \setminus R \\ &= E_1' \end{aligned}$$