

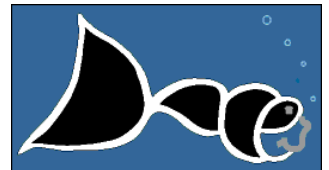
- Atomic register specification -

R. Guerraoui
Distributed Programming Laboratory
lpdwww.epfl.ch

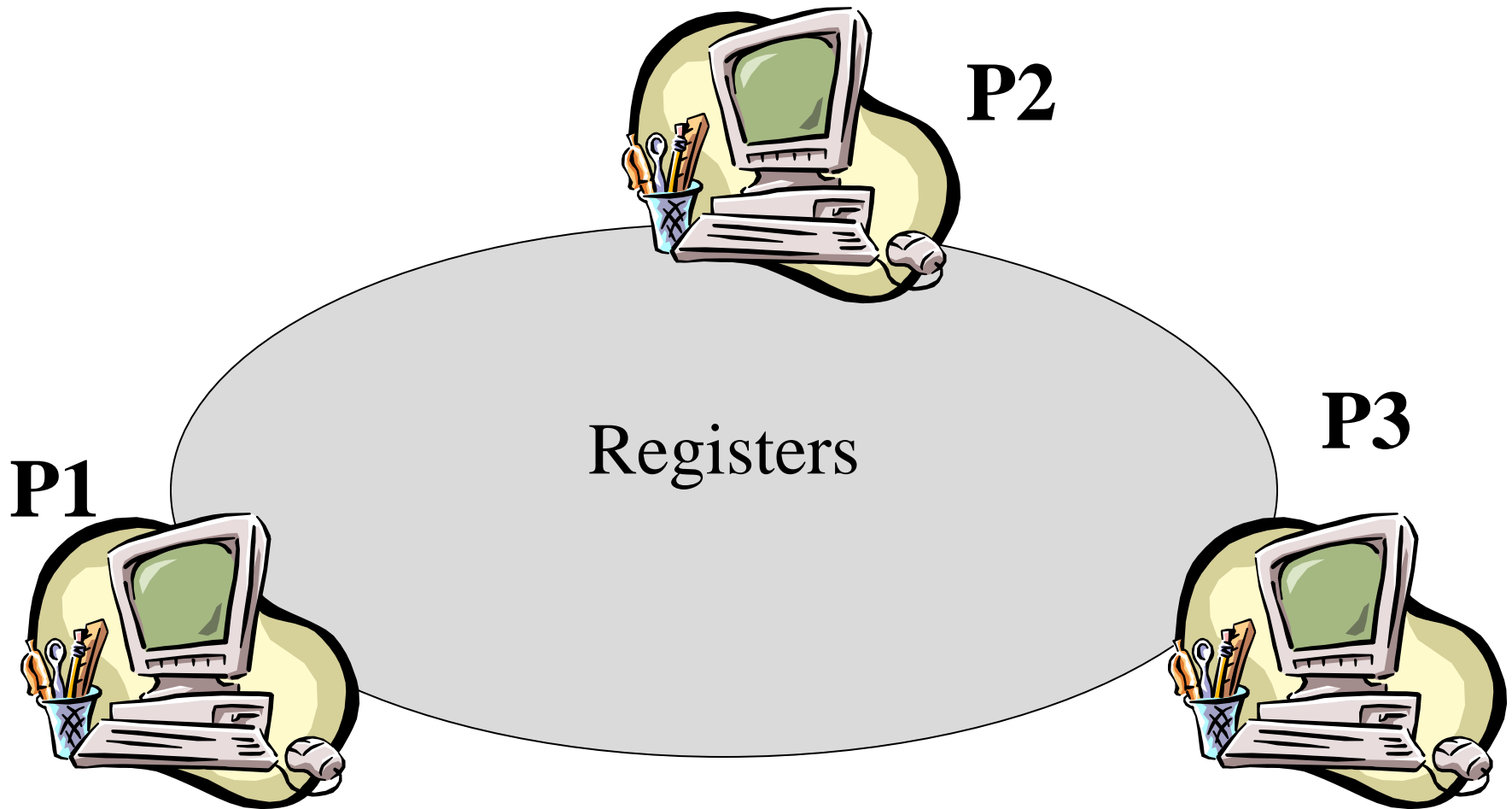


© R. Guerraoui

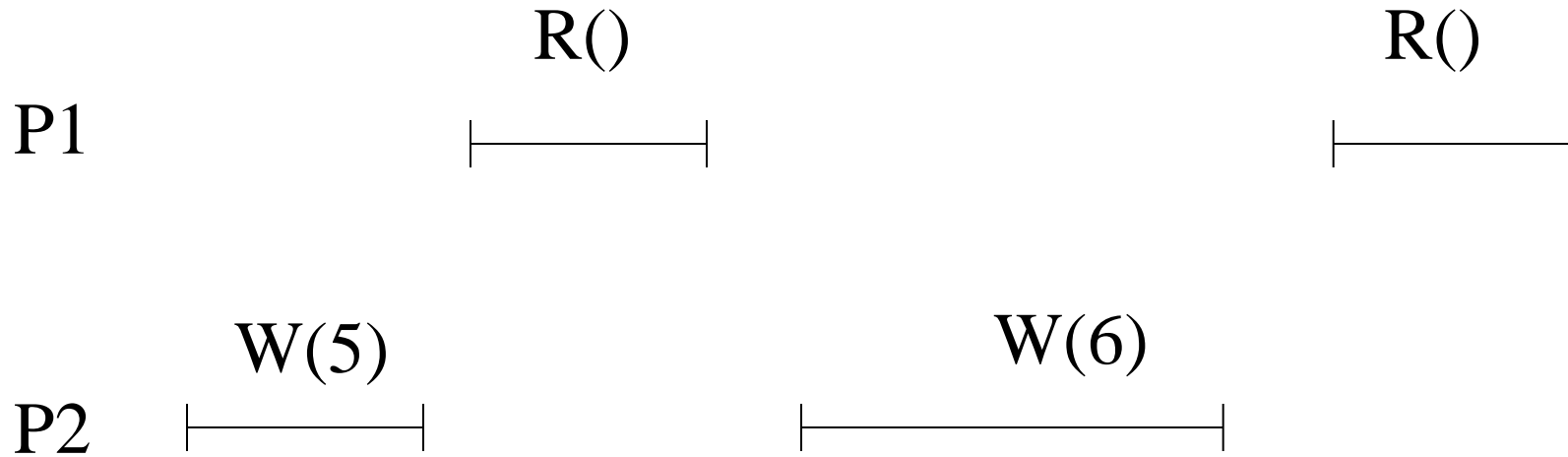
1



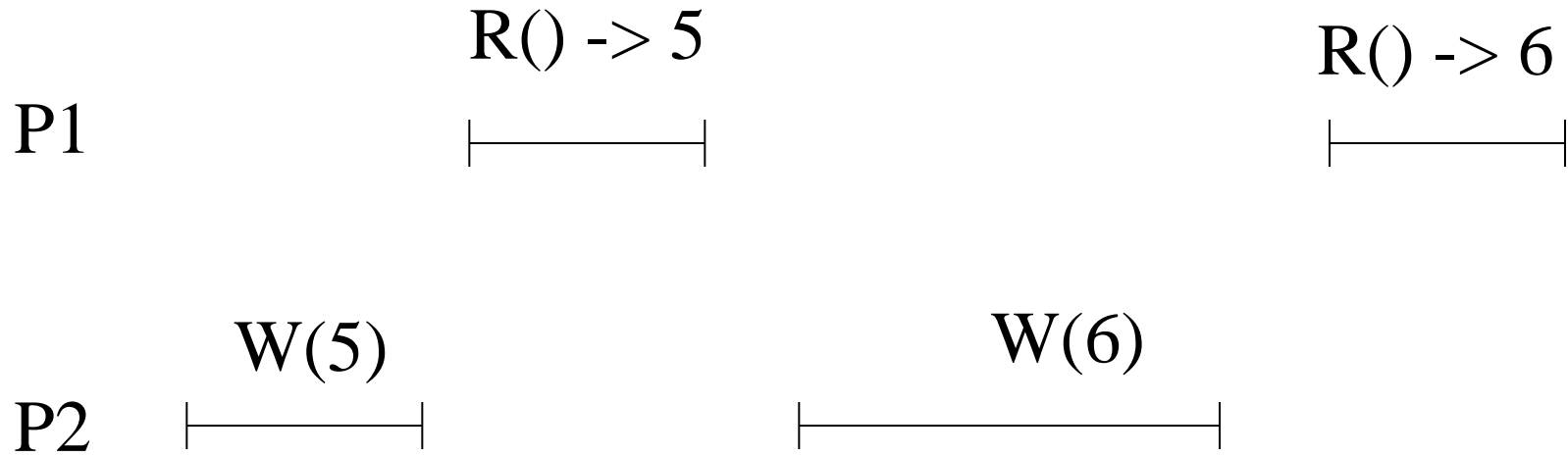
The application model



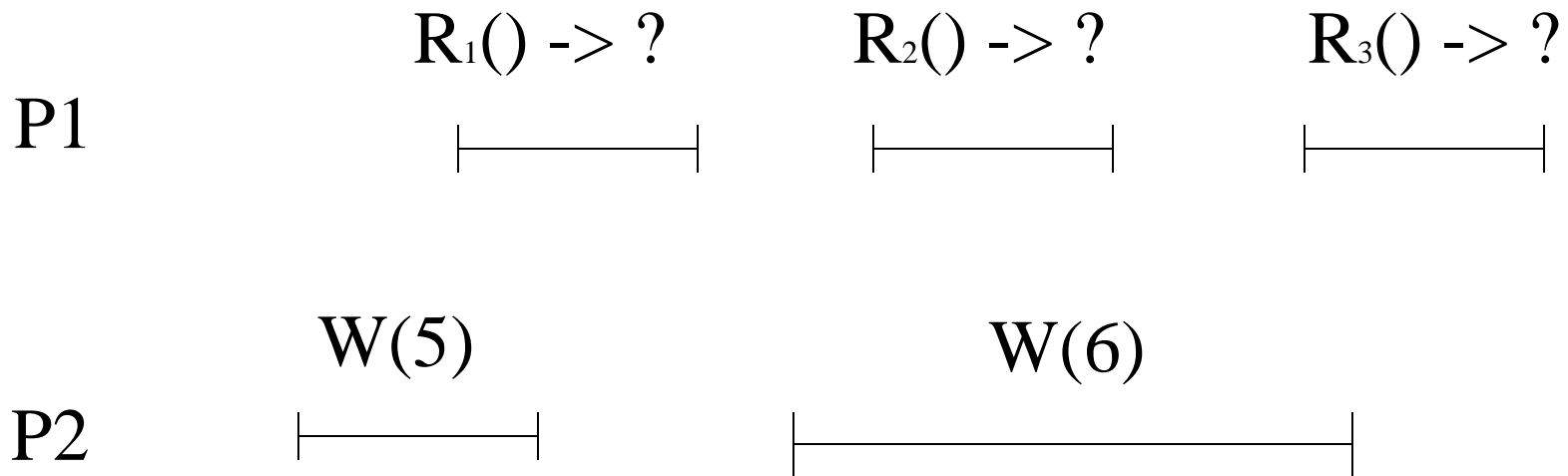
Sequential execution



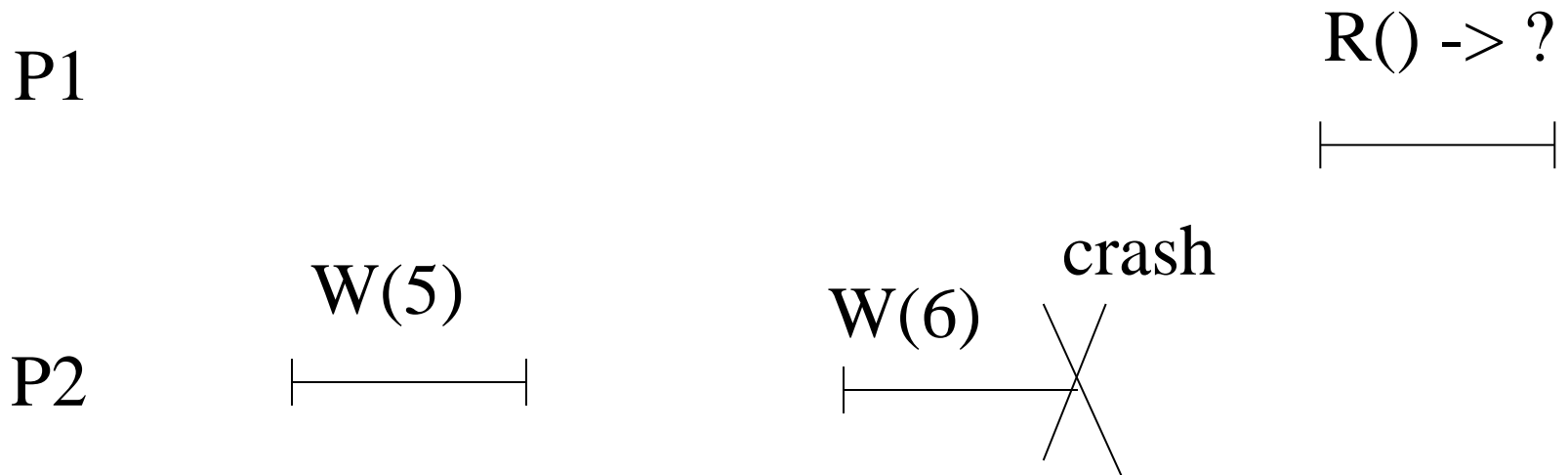
Sequential execution



Concurrent execution



Execution with failures



Safety

- ***An atomic register*** provides strong guarantees even when there is concurrency and failures
- The execution is equivalent to a sequential and failure-free execution (***linearization***)

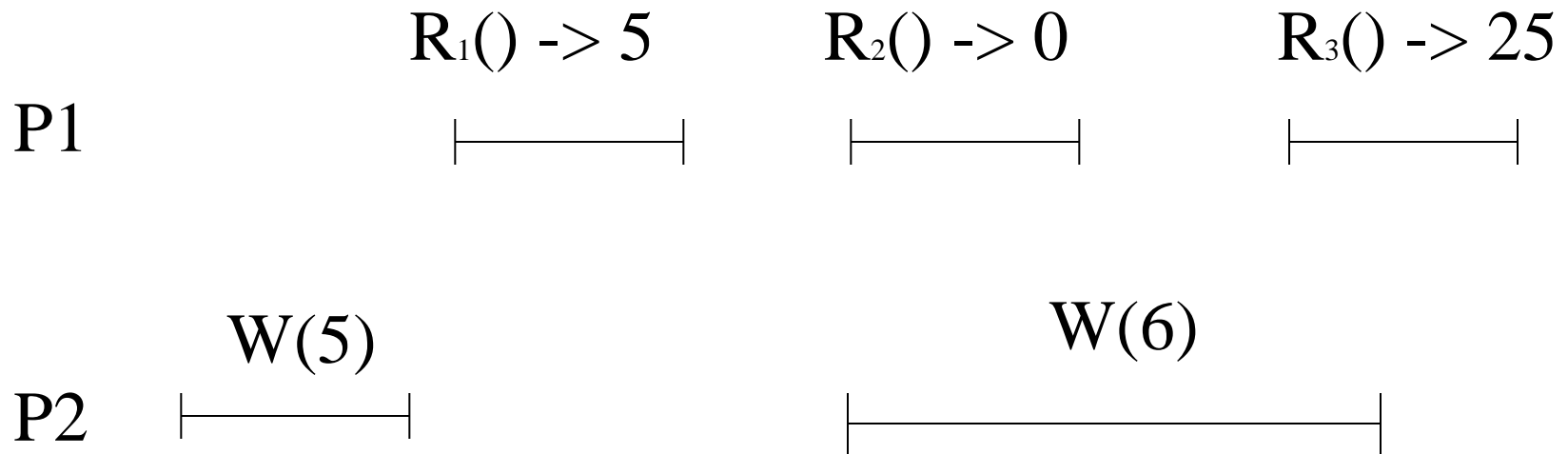
Atomic register

- Every failed (write) operation appears to be either complete or not to have been invoked at all

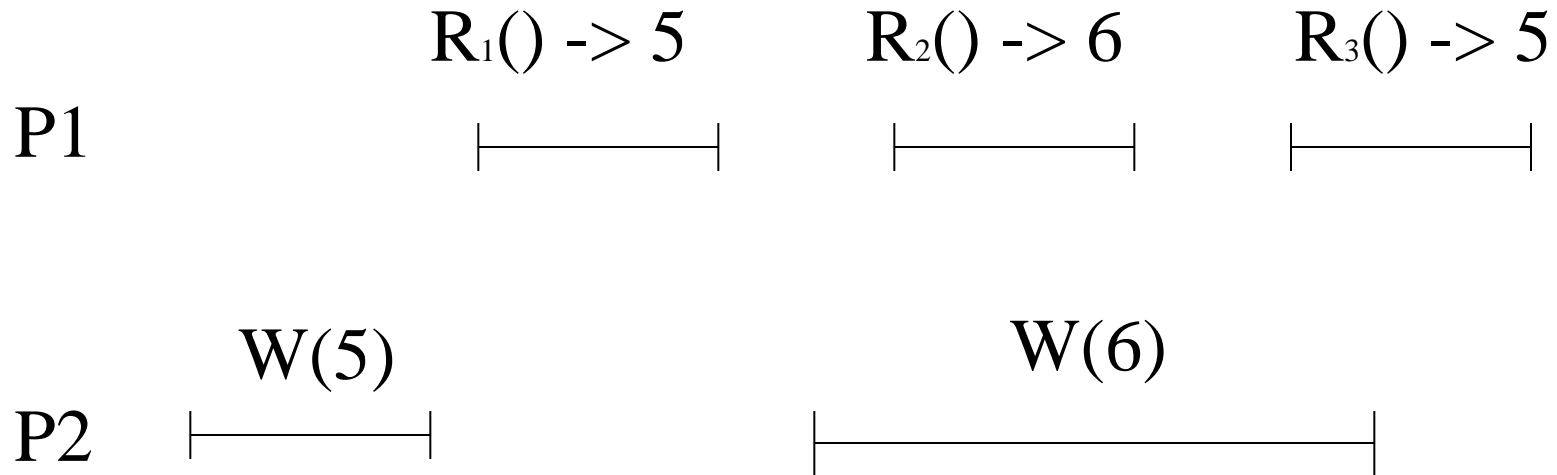
And

- Every complete operation appears to be executed at some instant between its invocation and reply time events

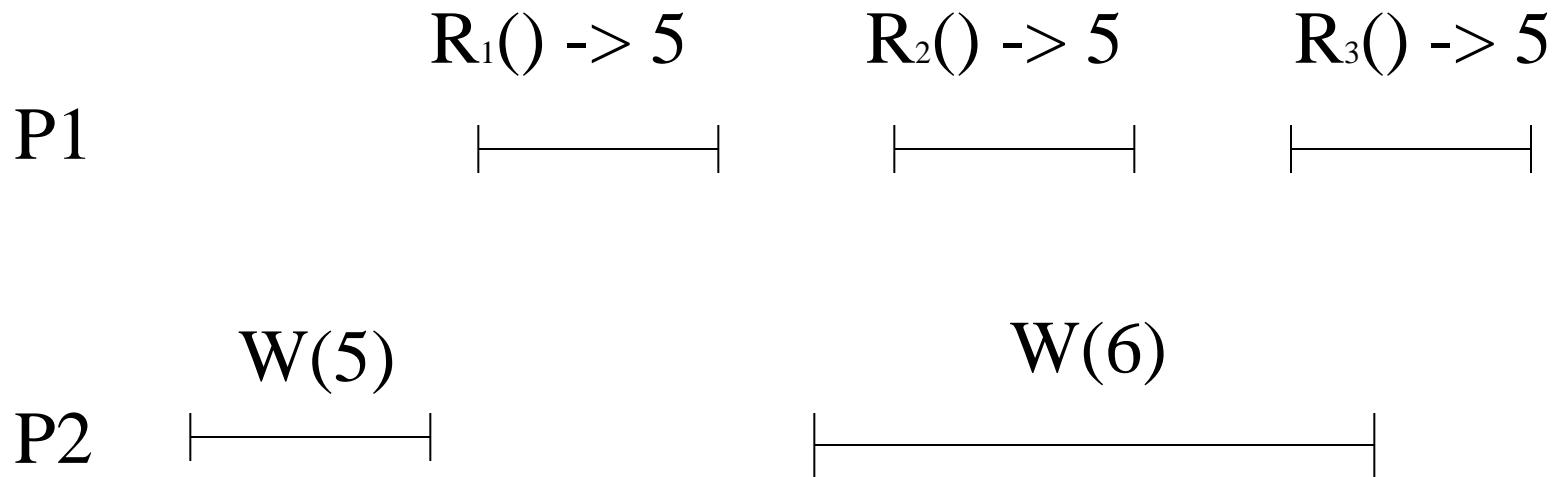
Execution 1



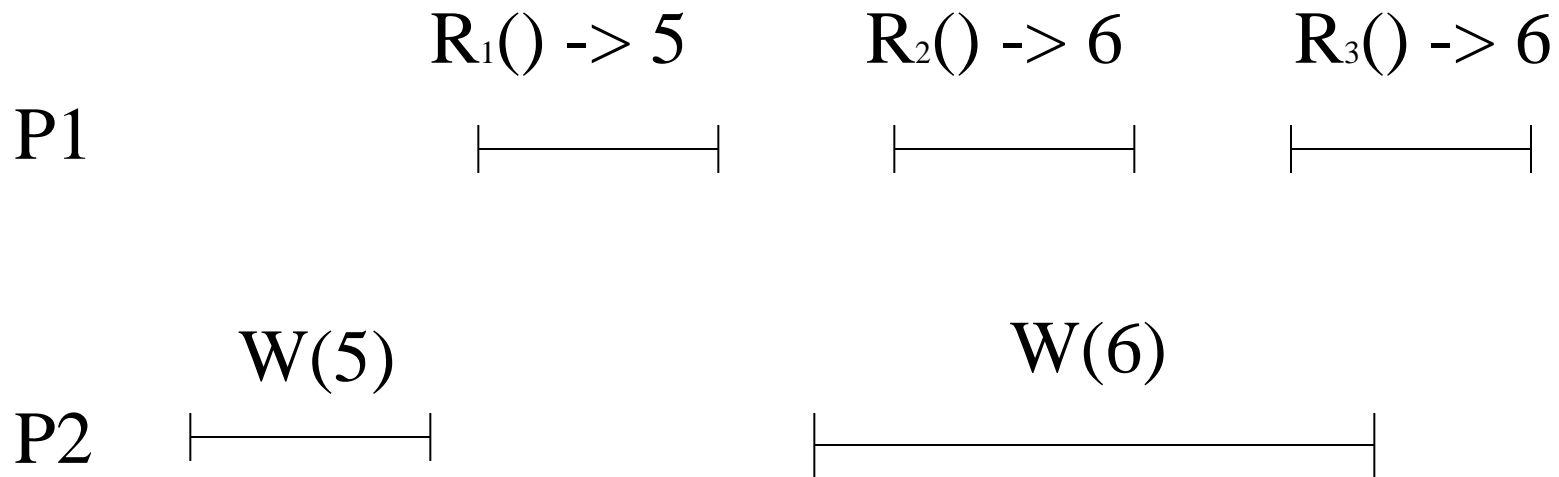
Execution 2



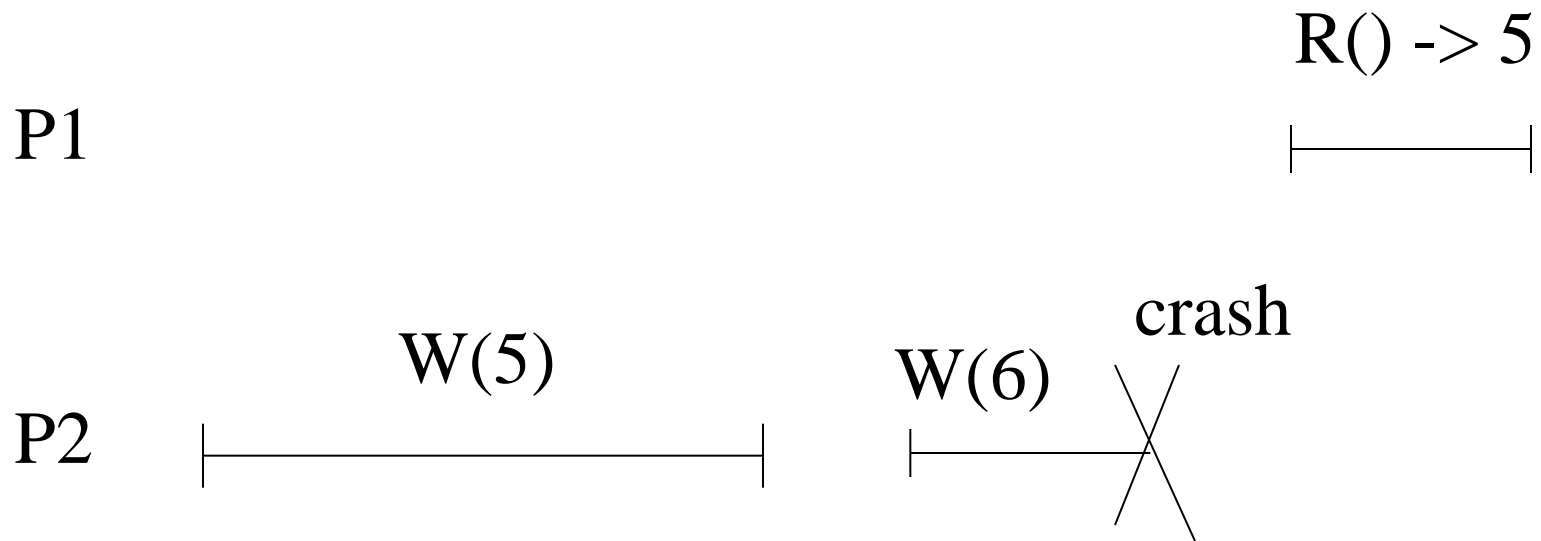
Execution 3



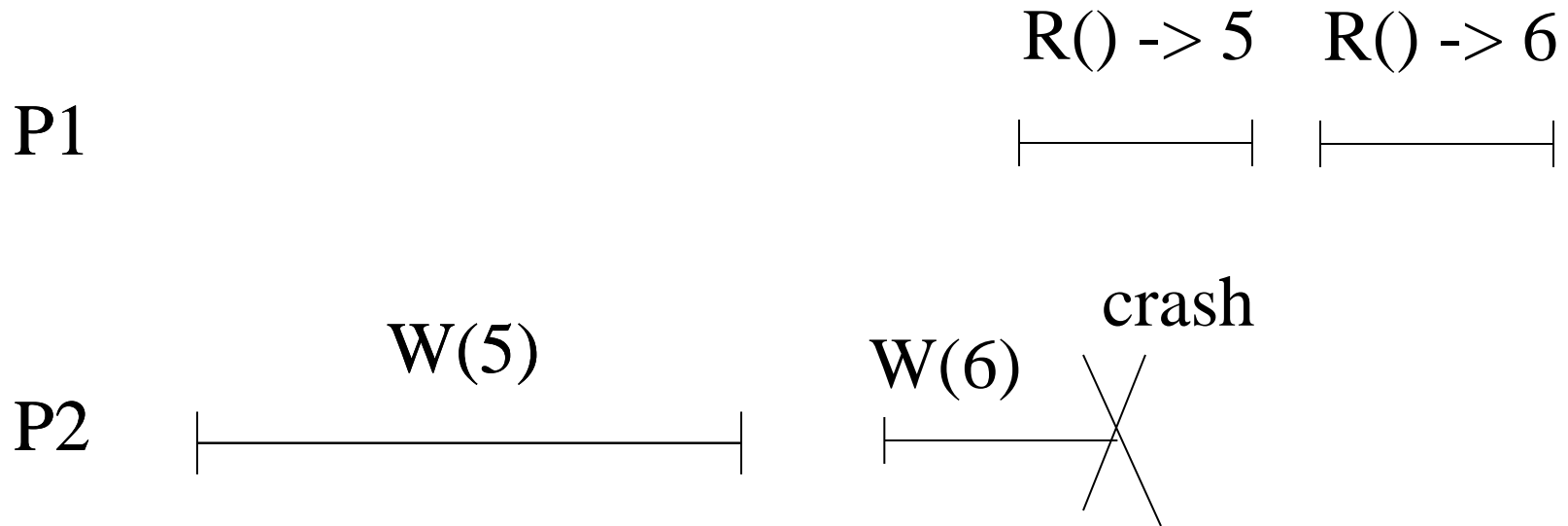
Execution 4



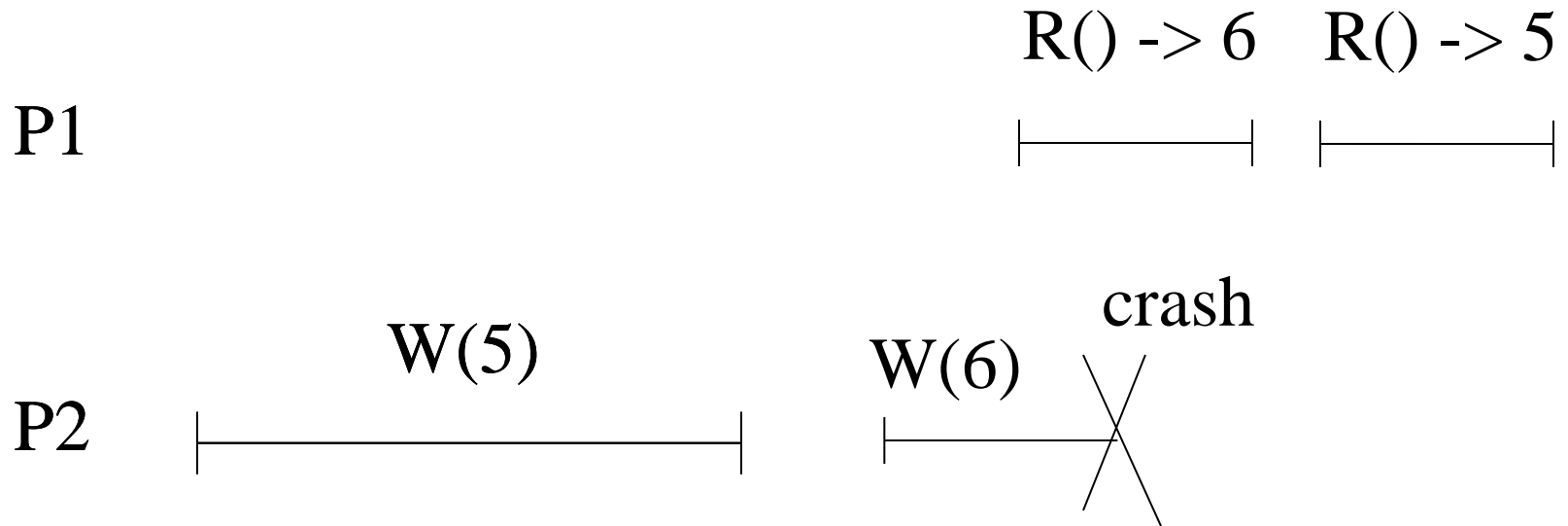
Execution 5



Execution 6



Execution 7



Correctness

- Execution 1: non-regular (safe)
- Executions 2 and 7: non-atomic (regular)
- Executions 3; 4, 5 and 6: atomic

Regular vs Atomic

- With one writer and no failed **Write()**, for a regular register to be atomic, two successive **Read()** must not overlap a **Write()**
- The regular register might in this case allow the first **Read()** to obtain the new value and the second **Read()** to obtain the old value