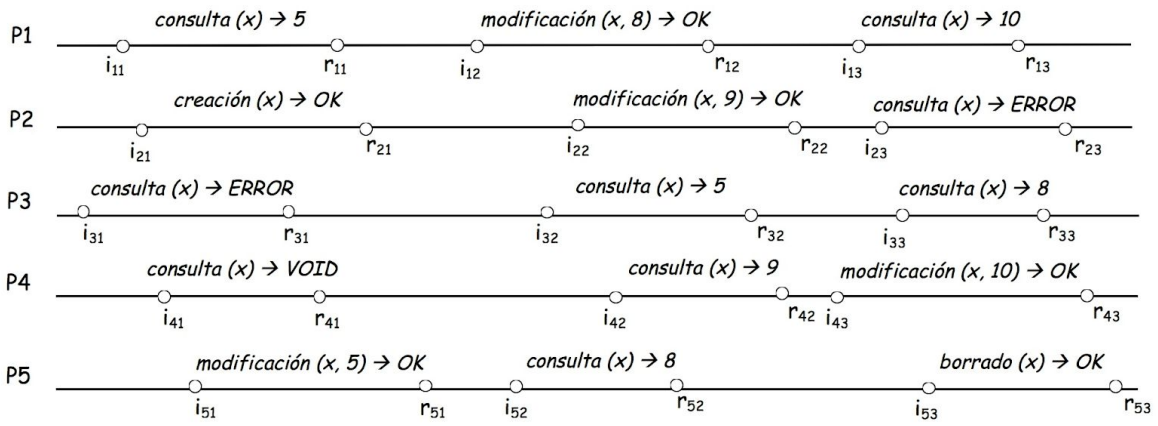
What is there to be done

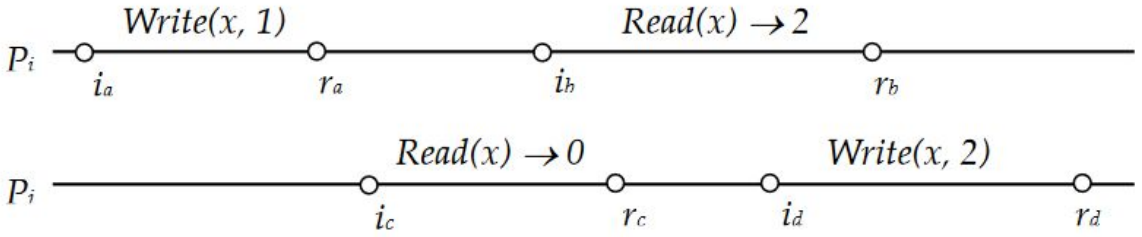
1. The teacher will propose a model of events to describe the interaction between the clients of the service, considered this as a black box. In an example, you will define the concepts of **interlaced sequence** of events and **valid execution** (also called *legal*). For each of the two cases listed above, we will discuss the consistency we should expect from the service. You can consult the Notes of Topic 3 (Sections 3.5.1 and 3.5.2).
2. In the following Figure, according to the semantics defined above, discuss in the group whether it meets the consistency criteria appropriate for respectively Case 1 and Case 2.



3. Prepare a brief report that includes: (a) three examples of invalid executions, (b) a few valid executions representative of the consistency criteria, and (c) for each of those valid executions, what consistency criterion is met. If the appropriate consistency criteria were not met, for one of the examples, comment on the behavior the server should have had to provide the required consistency.

1.- **Interlaced Sequence of events & valid (legal) execution**

If the operations of all the processes are considered globally, we obtain a sequence, where the particular sequences of each process are interlaced. This is known as an interlaced sequence of events. Since the operations overlap, the order of the invocation and response must be specified in the interlaced sequence.



In this example the interlaced sequence is [i_a, r_a, i_c, i_b, r_c, i_d, r_b, r_d].

From an interlaced sequence many different executions can be obtained. An execution is the application of operations in a certain order. Note that not all the executions are legal. A **legal execution** is one that follows the specification of the operations. Following with the previous example, a legal execution is [o_c, o_a, o_d, o_b] that can also be read as [$\text{Read}_j(x) \rightarrow 0$, $\text{Write}_i(x, 1)$, $\text{Write}_j(x, 2)$, $\text{Read}_i(x) \rightarrow 2$].

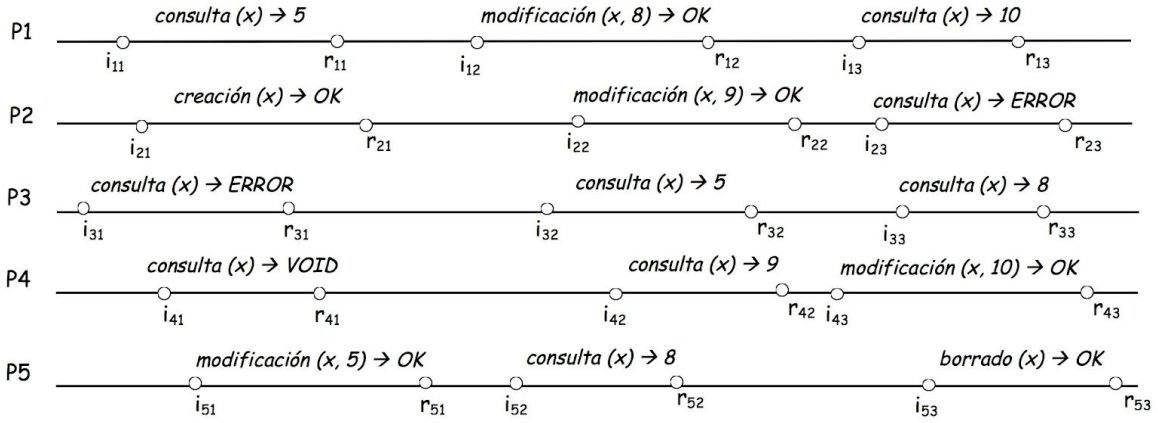
2.- **<u>On the following Figure, according to the semantics defined above, discuss in the group whether it meets the consistency criteria appropriate for respectively Case 1 and Case 2.</u>**

We will consider two possible Cases in the access to the objects of a service:

1.- Non-shared objects that are accessed from a single device. This is the case of a file system in the cloud that would not allow file sharing and a user could only use it from a single device.

2.- Objects shared or that can be accessed from different devices. This is the case of systems like Dropbox, or the one we finally intend for our cloud file service.

This is the figure:



And these are the semantics:

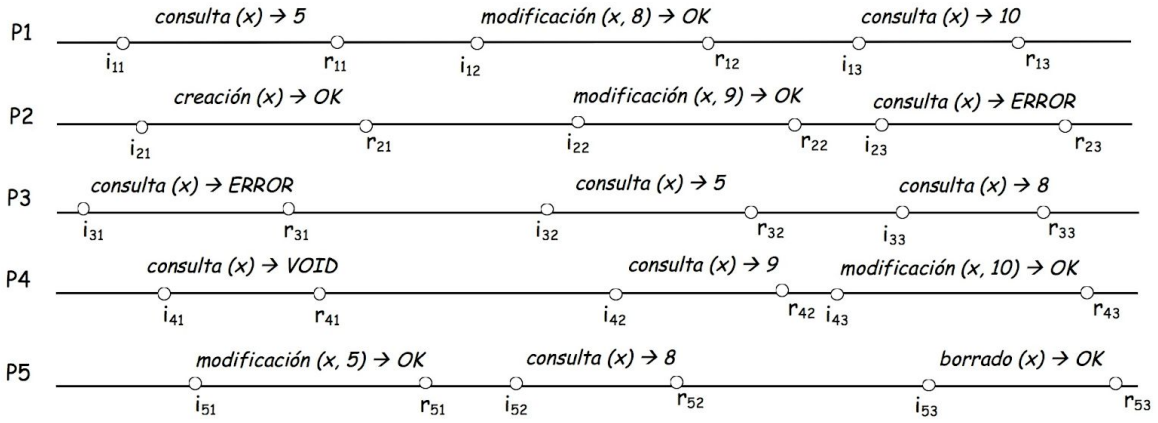
The semantics of transactions that we will follow in this example is that of the databases and guarantees that the individual operations are atomic and are ordered, in some way, over time. Note that a Dropbox file service has session semantics. In this case the transactional semantics are applied separately to the operations of reading the file and writing the file. Editing a file is not an atomic operation, but a session that includes read and write operations.

Now, answering the previous questions:

In the first case, it meets the consistency criteria. As it is used only from one device, and it's a file system in the cloud that does not allow file sharing, all its operations are individual, and according to the semantics, they guarantee that all the individual operations are atomic and ordered in some way over time.

In the second case, it also meets the consistency criteria. In this case we have objects shared, like Dropbox. This type of file services have session semantics. Even if editing a file is not an atomic operation, it is true that it is a session that includes read and write operations, operations that earlier had applied the transactional semantics separately.

3.- executions on the example



Invalid

a) **[O11, O31, O21, O41, O51, O32, O22, O42, O12, O52, O33, O43, O13, O53, O23]**
Here we can see that i11 receives 5 when doing consulta(X) despite the fact that x is not created.

b) **[O31, O21, O41, O51, O11, O32, O22, O42, O12, O52, O33, O13, O43, O53, O23]**
Here we can see that i33 receives 8 when doing consulta(X) and immediately afterwards i13 receives 10 when doing consulta(X) despite not any changes in between.

c) **[O31, O21, O41, O51, O11, O32, O12, O52, O22, O42, O33, O43, O13, O53, O23]**
Here we can see that i33 receives 8 when doing consulta(x) despite the fact that the value of X is 9 as per the change done by i22.

Valid execution no consistency criteria met

[O23,O31, O21, O41, O51, O11, O32, O33, O22, O42, O12, O52, O13, O43, O53]
In this execution sequential consistency is broken as i23 executes before the other 2 tasks in the process, and that would require sequence reordering. Linearizability can't happen as well as one of its prerequisites is sequential consistency.

Valid execution sequential consistency met

[O31, O21, O41, O51, O11, O32, O12, O52, O33, O22, O42, O13, O43, O53,O23]
In this execution sequential consistency is achieved as every invocation of any process is executed after the previous ones have finished. Linearizability can't happen in this example because i33 happens before i22 which would be impossible as i22 receives it's reply before i33 makes the invocation.

Valid execution both consistency criteria met

[O31, O21, O41, O51, O11 O32, O22, O42, O12, O52, O33, O43, O13, O53, O23]

As seen on the previous execution, we can see that both consistency criteria are met and it is indeed linearizable. We can confirm this by consulting the definition of linearizability, which states that a legal execution σ is linearizable if is sequentially consistent (which in this case it is) and if given any pair of operations Ok and Oh, if Rk occurs before Ih in τ, then in σ Ok is executed before Oh.