The Question for Exercise number 14.

Change transaction T 2 in Figure 17.2b to read:

read\_item(X);

X:= X+M;

if X > 90 then exit

else write\_item(X);

Discuss the final result of the different schedules in Figure 17.3 where M = 2

and N = 2, with respect to the following questions. Does adding the above condition change the final outcome? Does the outcome obey the implied consistency rule

(that the capacity of X is 90)?

***Answer:***

***The above condition does not change the final outcome unless the initial value of X > 88.***

***The outcome, however, does obey the implied consistency rule that X < 90, since the value of X is not updated if it becomes greater than 90.***

The Question for Exercise number 22.

Which of the following schedules is (conflict) serializable?

For each serializable schedule, determine the equivalent serial schedules.

(a) r1 (X); r3 (X); w1(X); r2(X); w3(X)

(b) r1 (X); r3 (X); w3(X); w1(X); r2(X)

(c) r3 (X); r2 (X); w3(X); r1(X); w1(X)

(d) r3 (X); r2 (X); r1(X); w3(X); w1(X)

***Answer:***

Let there be three transactions T1, T2, and T3. They are executed concurrently and produce

a schedule S.

1. This schedule is **not serializable** because T1 reads X (r1(X)) before T3 but T3 reads X(r3(X)) before T1 writes X (w1(X)), where X is a common data item.

The operation r2(X) of T2 does not affect the schedule at all so its position in the schedule is irrelevant. In a serial schedule T1, T2, and T3, the operation w1(X) comes after r3(X),

which does not happen in the question.

1. This schedule is ***not serializable*** because T1 reads X ( r1(X)) before T3 but T3 writes X(w3(X)) before T1 writes X (w1(X)). The operation r2(X) of T2 does not affect the schedule at all so its position in the schedule is irrelevant.

In a serial schedule T1, T3,and T2, r3(X) and w3(X) must come after w1(X), which does not happen in the question.

(c) This schedule is **serializable** because all conflicting operations of T3 happens before

all conflicting operation of T1. T2 has only one operation, which is a read on X (r2(X)), which does not conflict with any other operation. Thus this serializable schedule is equivalent to r2(X); r3(X); w3(X); r1(X); w1(X) serial schedule.

1. This is **not** a **serializable** schedule because T3 reads X (***r3***(X)) before T1 reads X (r1(X)) but r1(X) happens before T3 writes X (w3(X)). In a serial schedule T3, T2, and T1, r1(X) will happen after w3(X), which does not happen in the question.

The Question for Exercise number 23.

Consider the three transactions T1, T2, and T3, and the schedules S1 and S2 given below.

Draw the serializibility (precedence) graphs for S1 and S2 and state whether each schedule is serializable or not.

If a schedule is serializable, write down the equivalent serial schedule(s).

T1: r1(x); r1(z); w1(x)

T2: r2(z); r2(y); w2(z); w2(y)

T3: r3(x); r3(y); w3(y)

S1: r1(x); r2(z); r1(x); r3(x); r3(y); w1(x); w3(y); r2(y); w2(z); w2(y)

S2: r1(x); r2(z); r3(x); r1(z); r2(y); r3(y); w1(x); w2(z); w3(y); w2(y)

***Answer:***

Schedule S1:

It is a serializable schedule because

• T1 only reads X (r1(X)), which is not modified either by T2 or T3,

• T3 reads X (r3(X)) before T1 modifies it (w1(X)),T2 reads Y (r2(Y)) and writes it (w2(Y)) only after T3 has written to it (w3(Y)).

Thus, the serializability graph is

*T3 T1 T2*

**Schedule S2:**

It is **not** a **serializable** schedule because

• T2 reads Y (r2(Y)), which is then read and modified by T3 (w3(Y))

• T3 reads Y (r3(Y)), which then modified before T2 modifies Y (w2(Y)).

In the above order T3 interferes in the execution of T2, which makes the schedule

non serializable.

*T1 z T2*

*Y Y*

*T3*