Figure 19.1 Interleaved processing versus parallel processing of concurrent transactions.

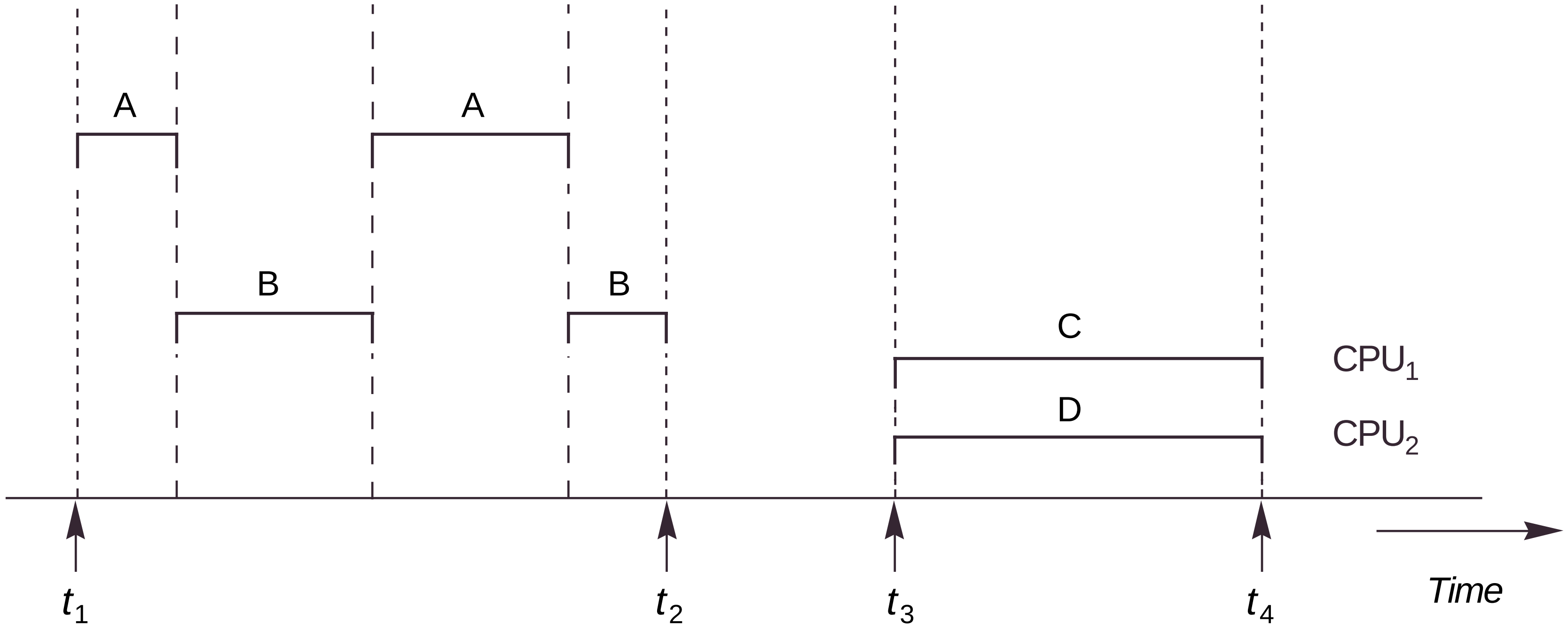


Figure 19.2 Two sample transactions. (a) Transaction T_1. (b) Transaction T_2.

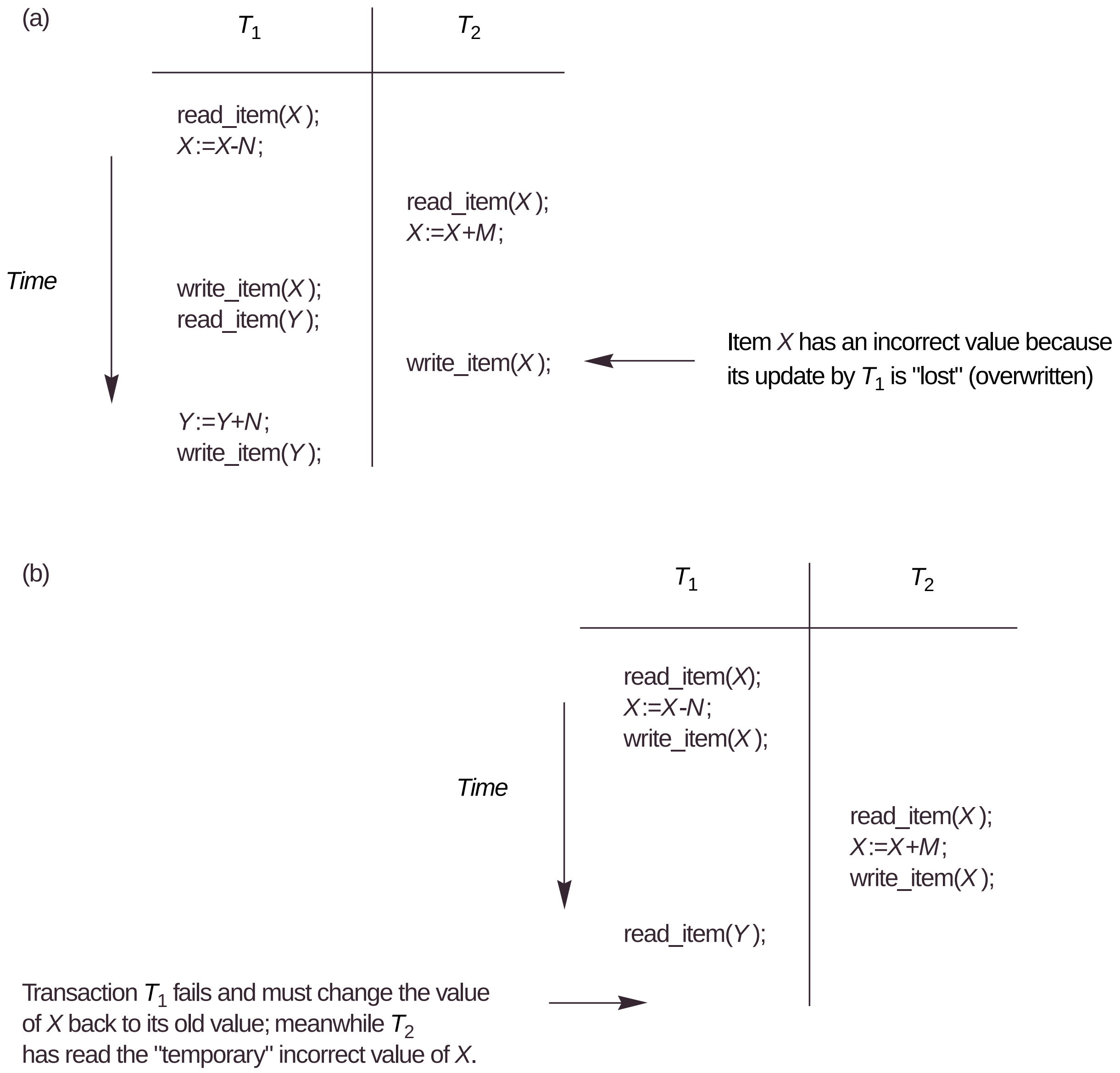
(a) T_1

```
read_item (X);
X:=X-N;
write_item (X);
read_item (Y);
Y:=Y+N;
write_item (Y);
```

(b) T_2

```
read_item (X);
X:=X+M;
write_item (X);
```

Figure 19.3 Some problems that occur when concurrent execution is uncontrolled. (a) The lost update problem. (b) The temporary update problem.



(a)

T_1	T_2
read_item(X); $X:=X-N$;	
	read_item(X); $X:=X+M$;

write_item(X);
read_item(Y);	

	write_item(X);
$Y:=Y+N$; write_item(Y);	

Time

Item X has an incorrect value because its update by T_1 is "lost" (overwritten)

(b)

T_1	T_2

read_item(X);
$X:=X-N$;
write_item(X);	

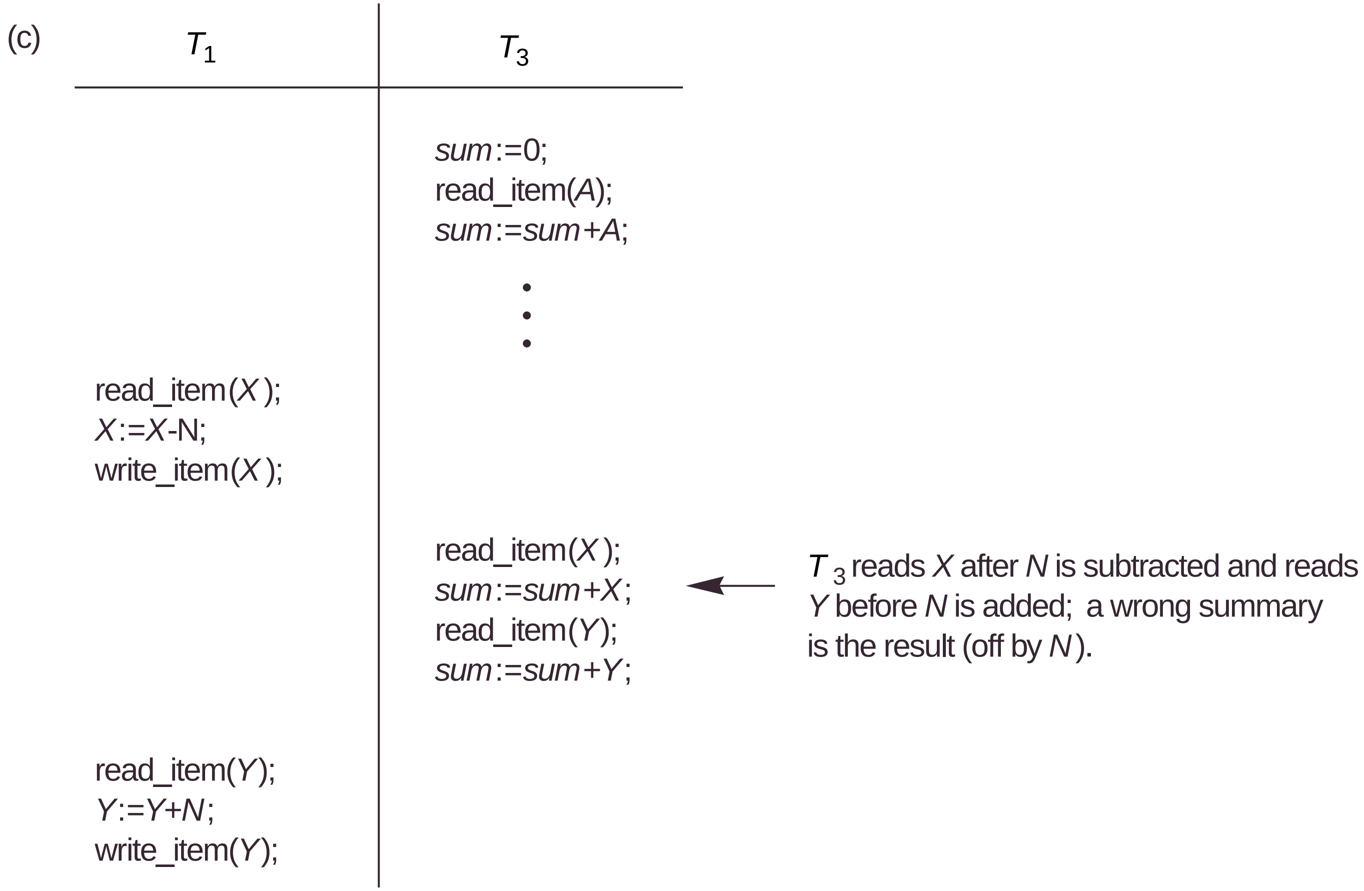
	read_item(X);
	$X:=X+M$;
	write_item(X);

read_item(Y);	

Time

Transaction T_1 fails and must change the value of X back to its old value; meanwhile T_2 has read the "temporary" incorrect value of X.

Figure 19.3 Some problems that occur when concurrent execution is uncontrolled. (c) The incorrect summary problem.



(c)	T_1	T_3

		sum:=0;
		read_item(*A*);
		sum:=*sum*+*A*;
		•
		•
		•

	read_item(*X*);
	X:=*X*-N;
	write_item(*X*);	

		read_item(*X*);
		sum:=*sum*+*X*;
		read_item(*Y*);
		sum:=*sum*+*Y*;

	read_item(*Y*);
	Y:=*Y*+*N*;
	write_item(*Y*);	

T_3 reads *X* after *N* is subtracted and reads *Y* before *N* is added; a wrong summary is the result (off by *N*).

Figure 19.4 State transition diagram illustrating the states for transaction execution.

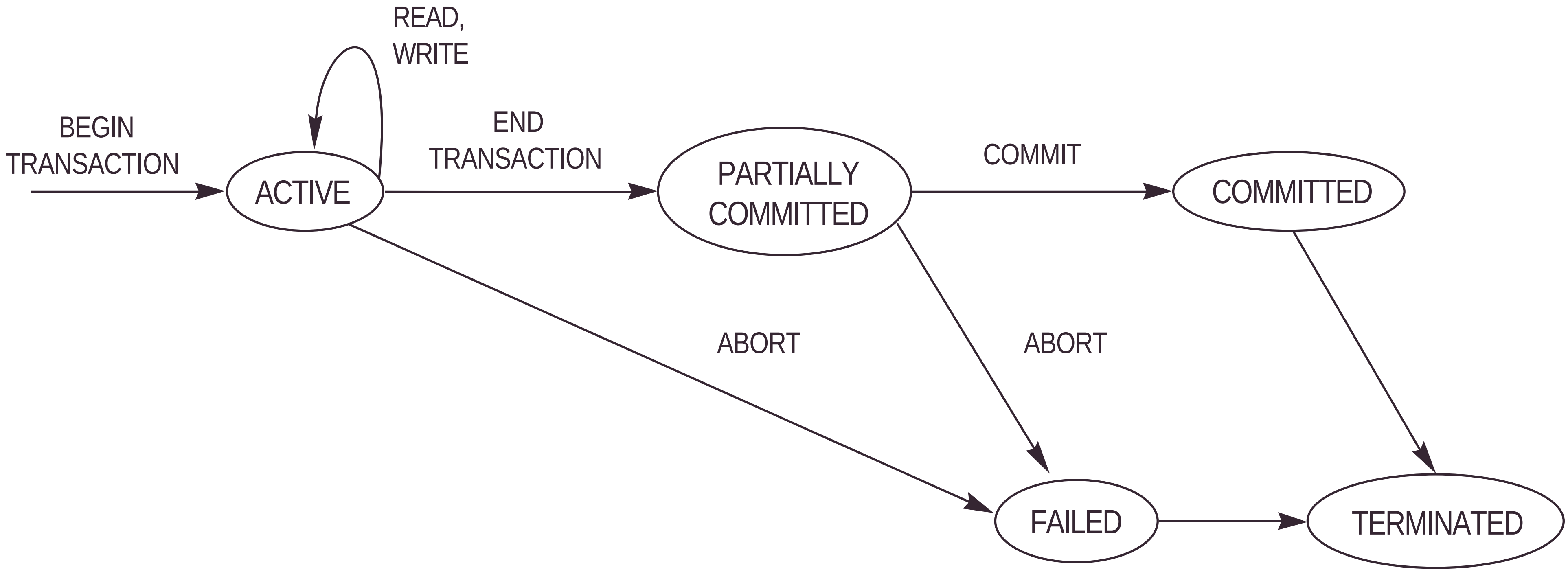
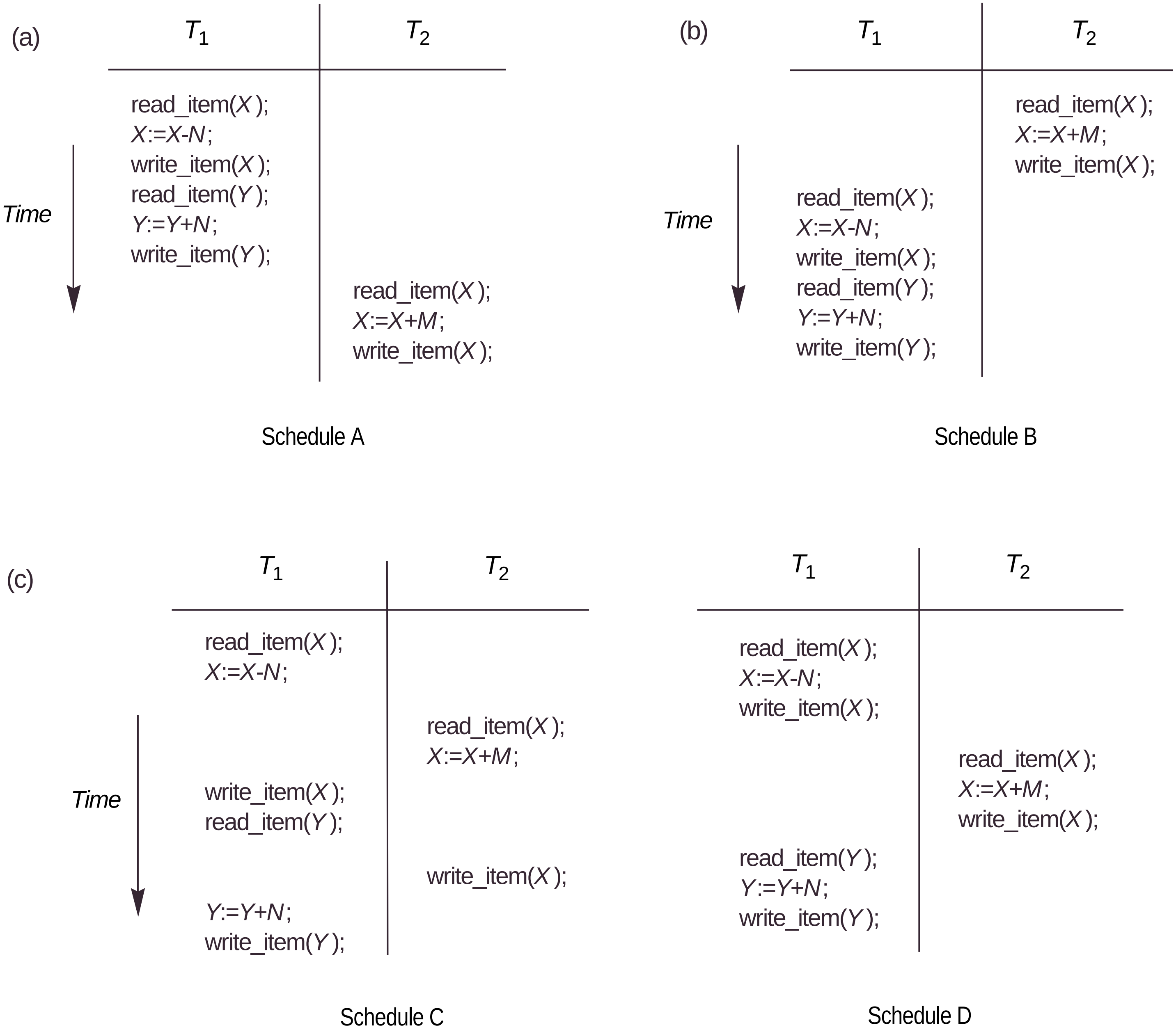


Figure 19.5 Examples of serial and nonserial schedules involving transactions T_1 and T_2. (a) Serial schedule A: T_1 followed by T_2. (b) Serial schedule B: T_2 followed by T_1. (c) Two nonserial schedules C and D with interleaving of operations.



(a)

T_1	T_2

read_item(X);
X:=X-N;
write_item(X);
read_item(Y);
Y:=Y+N;
write_item(Y);	

	read_item(X);
	X:=X+M;
	write_item(X);

Schedule A

(b)

T_1	T_2

	read_item(X);
	X:=X+M;
	write_item(X);

read_item(X);
X:=X-N;
write_item(X);
read_item(Y);
Y:=Y+N;
write_item(Y);	

Schedule B

(c)

T_1	T_2
read_item(X); X:=X-N;	
	read_item(X); X:=X+M;

write_item(X);
read_item(Y);	

	write_item(X);
Y:=Y+N; write_item(Y);	

Schedule C

T_1	T_2

read_item(X);
X:=X-N;
write_item(X);	

	read_item(X);
	X:=X+M;
	write_item(X);

read_item(Y);
Y:=Y+N;
write_item(Y);	

Schedule D

Figure 19.6 Two schedules that are result equivalent for the initial value of X = 100 but are not result equivalent in general.

S_1	S_2
read_item(X);	read_item(X);
X:=X+10;	X:=$X*$1.1;
write_item(X);	write_item(X);

Figure 19.7 Constructing the precedence graphs for schedules *A* to *D* from Figure 19.5 to test for conflict serializability. (a) Precedence graph for serial schedule *A*. (b) Precedence graph for serial schedule *B*. (c) Precedence graph for schedule *C* (not serializable). (d) Precedence graph for schedule *D* (serializable, equivalent to schedule *A*).

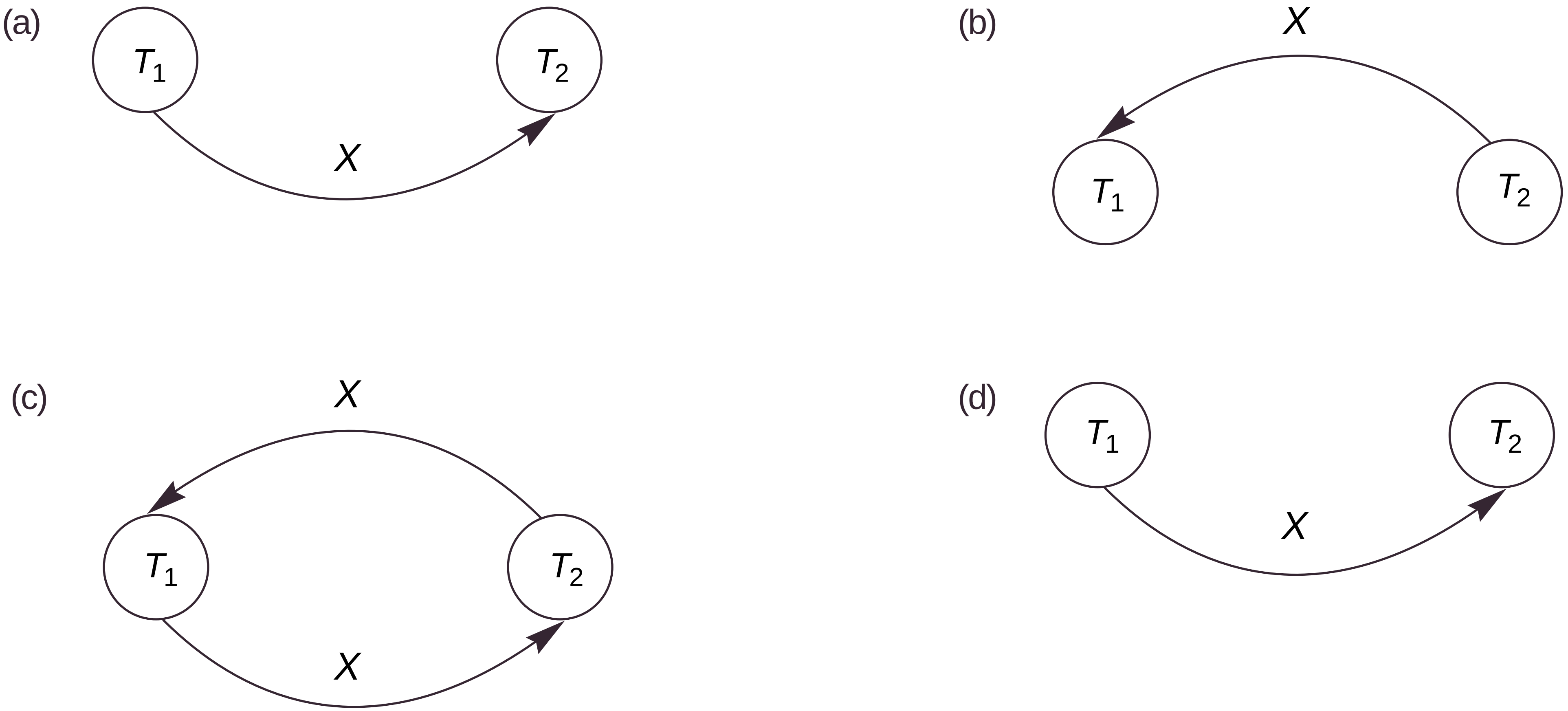


Figure 19.8 Another example of serializability testing. (a) The READ and WRITE operations of three transactions T_1, T_2, and T_3. (b) Schedule E. (c) Schedule F.

(a)

transaction T_1
read_item (X);
write_item (X);
read_item (Y);
write_item (Y);

transaction T_2
read_item (Z);
read_item (Y);
write_item (Y);
read_item (X);
write_item (X);

transaction T_3
read_item (Y);
read_item (Z);
write_item (Y);
write_item (Z);

(b)

Time →

transaction T_1	transaction T_2	transaction T_3

	read_item (Z);
	read_item (Y);
	write_item (Y);	

		read_item (Y);
		read_item (Z);

read_item (X);
write_item (X);		

		write_item (Y);
		write_item (Z);

	read_item (X);	

read_item (Y);
write_item (Y);		

	write_item (X);	

Schedule E

(c)

Time →

transaction T_1	transaction T_2	transaction T_3

		read_item (Y);
		read_item (Z);

read_item (X);
write_item (X);		

		write_item (Y);
		write_item (Z);

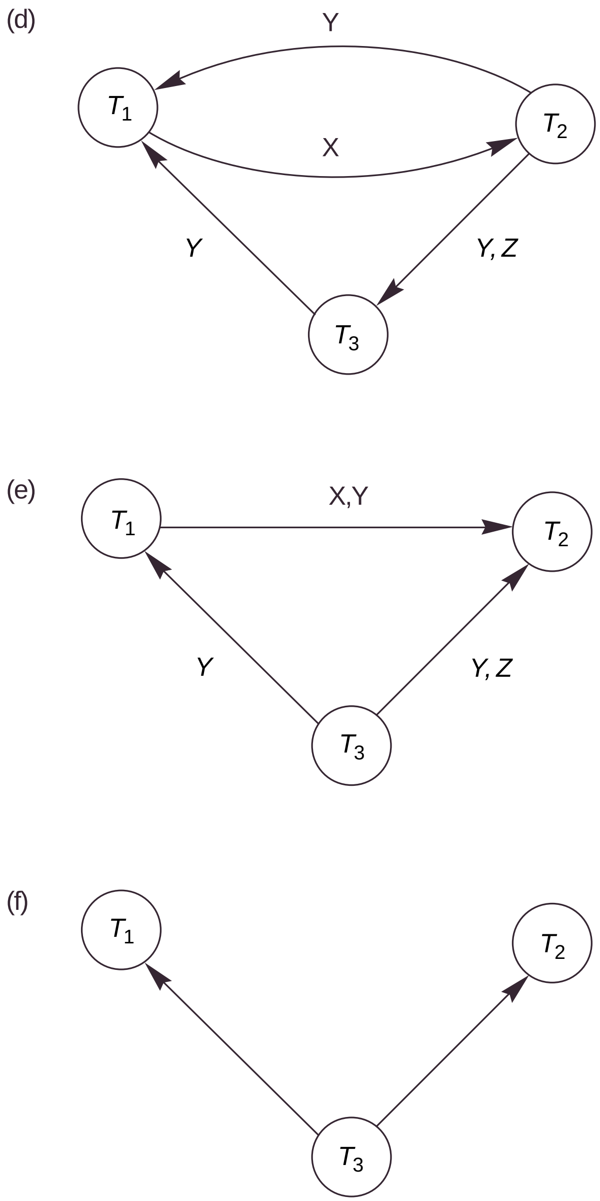
	read_item (Z);	

read_item (Y);
write_item (Y);		

	read_item (Y);
	write_item (Y);
	read_item (X);
	write_item (X);	

Schedule F

Figure 19.8 Another example of serializability testing. (d) Precedence graph for schedule *E*. (e) Precedence graph for schedule *F*. (f) Precedence graph with two equivalent serial schedules.



(d)

Equivalent serial schedules

None

Reason

cycle $X (T_1 \rightarrow T_2), Y (T_2 \rightarrow T_1)$

cycle $X (T_1 \rightarrow T_2), YZ (T_2 \rightarrow T_3), Y (T_3 \rightarrow T_1)$

(e)

Equivalent serial schedules

$T_3 \longrightarrow T_1 \longrightarrow T_2$

(f)

Equivalent serial schedules

$T_3 \longrightarrow T_1 \longrightarrow T_2$

$T_3 \longrightarrow T_2 \longrightarrow T_1$