nition if $H$ were a *complete* history. Otherwise there are problems. First, there is an artificial problem in that a partial history can never be equivalent to a serial one. This is because serial histories must be complete by definition, and two histories can be equivalent only if they contain the same set of operations. But even assuming for the moment that we did not insist on serial histories being complete, we would still have a problem—a problem of substance and not mere definition. Namely, an incomplete execution of a transaction does not necessarily preserve the consistency of the database. Thus, serializability would be an inappropriate correctness condition if it merely stated that a history be computationally equivalent to a serial execution of some possibly partial transactions, since such a history does not necessarily represent a consistency preserving execution.

We are therefore led to a slightly more complex definition of serializability. Although more complex, the definition is still natural and, most importantly, sound. A history $H$ is *serializable* (*SR*) if its committed projection, $C(H)$, is equivalent to a serial history $H_s$.

This takes care of the problems, because $C(H)$ is a *complete* history. Moreover, it is not an arbitrarily chosen complete history. If $H$ represents the execution so far, it is really only the *committed* transactions whose execution the DBS has unconditionally guaranteed. All other transactions may be aborted.

## 2.3 THE SERIALIZABILITY THEOREM

We can determine whether a history is serializable by analyzing a graph derived from the history called a serialization graph. Let $H$ be a history over $T = \{T_1, \ldots, T_n\}$. The *serialization graph* (*SG*) for $H$, denoted $SG(H)$, is a directed graph whose nodes are the transactions in $T$ that are committed in $H$ and whose edges are all $T_i \rightarrow T_j$ $(i \neq j)$ such that one of $T_i$'s operations precedes and conflicts with one of $T_j$'s operations in $H$. For example:

$$H_4 = \begin{array}{l} \qquad\qquad\qquad\qquad r_3[x] \rightarrow w_3[x] \rightarrow c_3 \\ r_1[x] \rightarrow w_1[x] \rightarrow w_1[y] \rightarrow c_1 \\ \qquad r_2[x] \rightarrow w_2[y] \rightarrow c_2 \end{array}$$

$$SG(H_4) = \overset{\frown}{T_2 \rightarrow T_1 \rightarrow T_3}$$

The edge $T_1 \rightarrow T_3$ is in $SG(H_4)$ because $w_1[x] < r_3[x]$, and the edge $T_2 \rightarrow T_3$ is in $SG(H_4)$ because $r_2[x] < w_3[x]$. Notice that a single edge in $SG(H_4)$ can be present because of more than one pair of conflicting operations. For instance, the edge $T_2 \rightarrow T_1$ is caused both by $r_2[x] < w_1[x]$ and $w_2[y] < w_1[y]$.

In general, the existence of edges $T_i \rightarrow T_j$ and $T_j \rightarrow T_k$ in an SG does not necessarily imply the existence of edge $T_i \rightarrow T_k$. For instance, with $w_3[z]$ replacing $w_3[x]$ in $T_3$, $SG(H_4)$ becomes

$T_2 \rightarrow T_1 \rightarrow T_3$

since there is no conflict between $T_2$ and $T_3$.[7]

Each edge $T_i \rightarrow T_j$ in SG($H$) means that at least one of $T_i$'s operations precedes and conflicts with one of $T_j$'s. This suggests that $T_i$ should precede $T_j$ in any serial history that is equivalent to $H$. If we can find a serial history, $H_s$, consistent with all edges in SG($H$), then $H_s \equiv H$ and so $H$ is SR. We can do this as long as SG($H$) is acyclic.

In our previous example, SG($H_5$) is acyclic. A serial history where transactions appear in an order consistent with the edges of SG($H_5$) is $T_2\ T_1\ T_3$. Indeed this is the only such serial history. You can easily verify that $H_s$ is equivalent to $T_2\ T_1\ T_3$ and is therefore SR. We formalize this intuitive argument in the following theorem—the fundamental theorem of serializability theory.

**Theorem 2.1:** (The Serializability Theorem) A history $H$ is serializable iff SG($H$) is acyclic.

*Proof:* (if) Suppose $H$ is a history over $T = \{T_1, T_2, \ldots, T_n\}$. Without loss of generality, assume $T_1, T_2, \ldots, T_m$ ($m \leq n$) are all of the transactions in $T$ that are committed in $H$. Thus $T_1, T_2, \ldots, T_m$ are the nodes of SG($H$). Since SG($H$) is acyclic it may be topologically sorted.[8] Let $i_1, \ldots, i_m$ be a permutation of $1, 2, \ldots, m$ such that $T_{i_1}, T_{i_2}, \ldots, T_{i_m}$ is a topological sort of SG($H$). Let $H_s$ be the serial history $T_{i_1} T_{i_2} \ldots T_{i_m}$. We claim that C($H$) $\equiv H_s$. To see this, let $p_i \in T_i$ and $q_j \in T_j$, where $T_i$, $T_j$ are committed in $H$. Suppose $p_i$, $q_j$ conflict and $p_i <_H q_j$. By definition of SG($H$), $T_i \rightarrow T_j$ is an edge in SG($H$). Therefore in any topological sort of SG($H$), $T_i$ must appear before $T_j$. Thus in $H_s$ all operations of $T_i$ appear before any operation of $T_j$, and in particular, $p_i <_{H_s} q_j$. We have proved that any two conflicting operations are ordered in C($H$) in the same way as $H_s$. Thus C($H$) $\equiv H_s$ and, because $H_s$ is serial by construction, $H$ is SR as was to be proved.

(only if) Suppose history $H$ is SR. Let $H_s$ be a serial history equivalent to C($H$). Consider an edge $T_i \rightarrow T_j$ in SG($H$). Thus there are two conflicting operations $p_i$, $q_j$ of $T_i$, $T_j$ (respectively), such that $p_i <_H q_j$. Because C($H$) $\equiv H_s$, $p_i <_{H_s} q_j$. Because $H_s$ is serial and $p_i$ in $T_i$ precedes $q_j$ in $T_j$, it follows that $T_i$ appears before $T_j$ in $H_s$. Thus, we've shown that if $T_i \rightarrow T_j$ is in SG($H$) then $T_i$ appears before $T_j$ in $H_s$. Now suppose there is a cycle in SG($H$), and without loss of generality let that cycle be $T_1 \rightarrow T_2 \rightarrow \ldots \rightarrow T_k \rightarrow T_1$. These edges imply that in $H_s$, $T_1$ appears before $T_2$ which appears before $T_3$ which appears ... before $T_k$ which appears before $T_1$. Thus, the existence of the cycle implies that each of $T_1, T_2, \ldots, T_k$ appears before

[7]We say that *two transactions conflict* if they contain conflicting operations.

[8]See Section A.3 of the Appendix for a definition of "topological sort of a directed acyclic graph."

itself in the serial history $H_s$, an absurdity. So no cycle can exist in SG($H$). That is, SG($H$) is an acyclic directed graph, as was to be proved. □

From the proof of the "if" part of this theorem we see that if a complete history H has an acyclic SG($H$), then $H$ is equivalent to *any* serial history that's a topological sort of SG($H$). Since the latter can have more than one topological sort, $H$ may be equivalent to more than one serial history. For instance,

$$H_6 = w_1[x]\ w_1[y]\ c_1\ r_2[x]\ r_3[y]\ w_2[x]\ c_2\ w_3[y]\ c_3$$

has the serialization graph

$$SG(H_6) = \quad T_1 \rightarrow T_3 \qquad T_2$$

which has two topological sorts, $T_1$, $T_2$, $T_3$ and $T_1$, $T_3$, $T_2$. Thus $H_6$ is equivalent to both $T_1\ T_2\ T_3$ and $T_1\ T_3\ T_2$.

## 2.4 RECOVERABLE HISTORIES

In Chapter 1, we saw that to ensure correctness in the presence of failures the scheduler must produce executions that are not only SR but also recoverable. We also discussed some additional requirements that may be desirable—preventing cascading aborts and the loss of before images—leading us to the idea of strict executions. Like serializability, these concepts can be conveniently formulated in the language of histories.

A transaction $T_i$ reads data item $x$ from $T_j$ if $T_j$ was the transaction that had last written into $x$ but had not aborted at the time $T_i$ read $x$. More precisely, we say that $T_i$ *reads x from* $T_j$ in history $H$ if

1. $w_j[x] < r_i[x]$;
2. $a_j \not< r_i[x]$[9] and
3. if there is some $w_k[x]$ such that $w_j[x] < w_k[x] < r_i[x]$, then $a_k < r_i[x]$.

We say that $T_i$ *reads from* $T_j$ in $H$ if $T_i$ reads some data item from $T_j$ in $H$. Notice that it is possible for a transaction to read a data item from itself (i.e., $w_i[x] < r_i[x]$).

A history $H$ is called *recoverable* ($RC$) if, whenever $T_i$ reads from $T_j$ ($i \neq j$) in $H$ and $c_i \in H$, $c_j < c_i$. Intuitively, a history is recoverable if each transaction commits after the commitment of all transactions (other than itself) from which it read.

A history $H$ *avoids cascading aborts* ($ACA$) if, whenever $T_i$ reads $x$ from $T_j$ ($i \neq j$), $c_j < r_i[x]$. That is, a transaction may read only those values that are written by committed transactions or by itself.

---

[9] $p \not< q$ denotes that operation $p$ does not precede $q$ in the partial order.