Efficient Black-box Checking of Snapshot Isolation in Databases

(Conference VLDB'2024)

Hengfeng Wei

hfwei@nju.edu.cn

August 7, 2023

Database Transactions

A database transaction is a *group* of operations

that should be executed atomically.

Isolation Levels

Transactions may be executed concurrently.

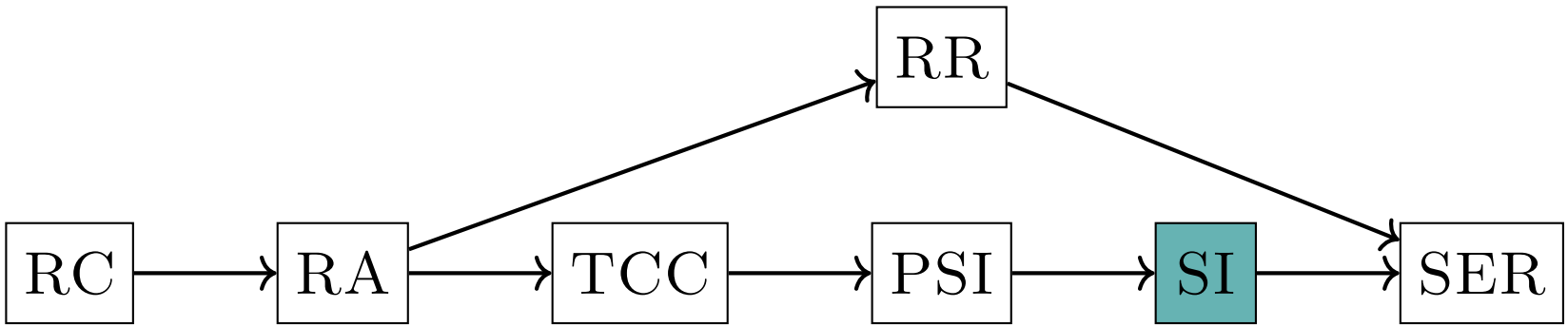
The isolation levels specify how they are isolated from each other.

Serializability (SER)

All transactions appear to execute serially, one after another.

too expensive, especially for distributed transactions

Snapshot Isolation (SI)



Snapshot Isolation (SI)

example

Snapshot Read: Each transaction reads data from a snapshot of committed data valid as of the (logical) time the transaction started.

Snapshot Write: Concurrent transactions cannot write to the same key. One of them must be aborted.

SI Prevents the "Lost Update" Anomaly

T_0

$\boxed{\mathsf{W}(acct, 0)}$

SI Prevents the "Lost Update" Anomaly

T_A

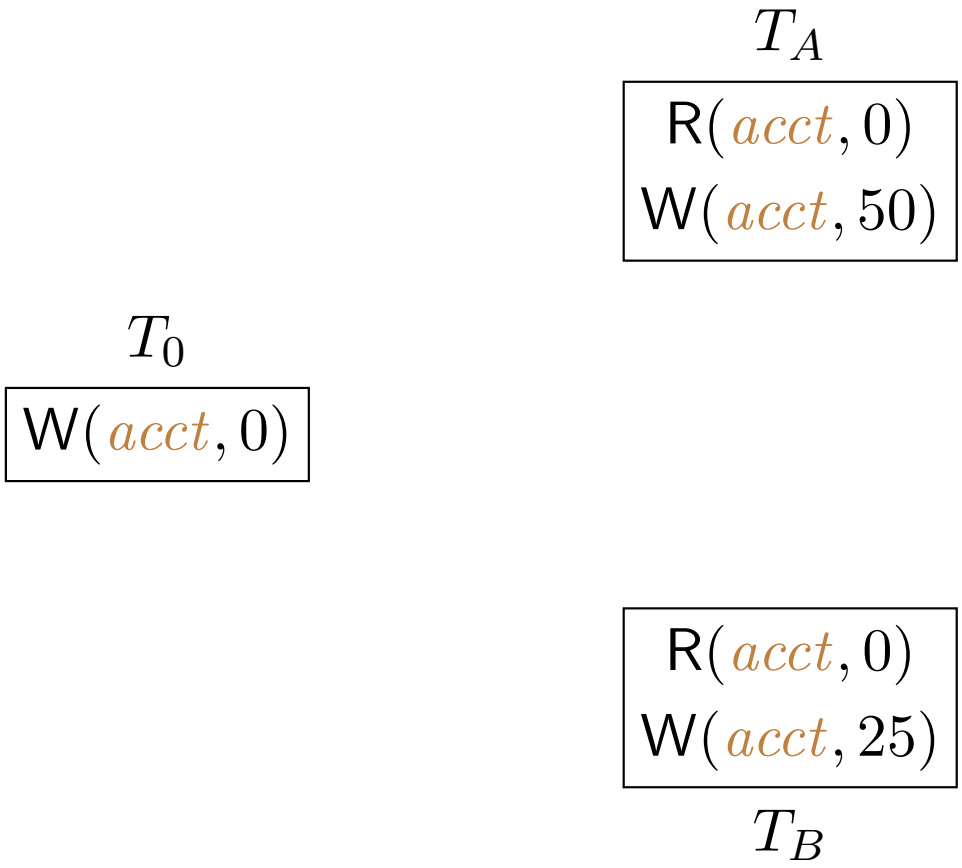
$\mathsf{R}(acct, 0)$

$\mathsf{W}(acct, 50)$

T_0

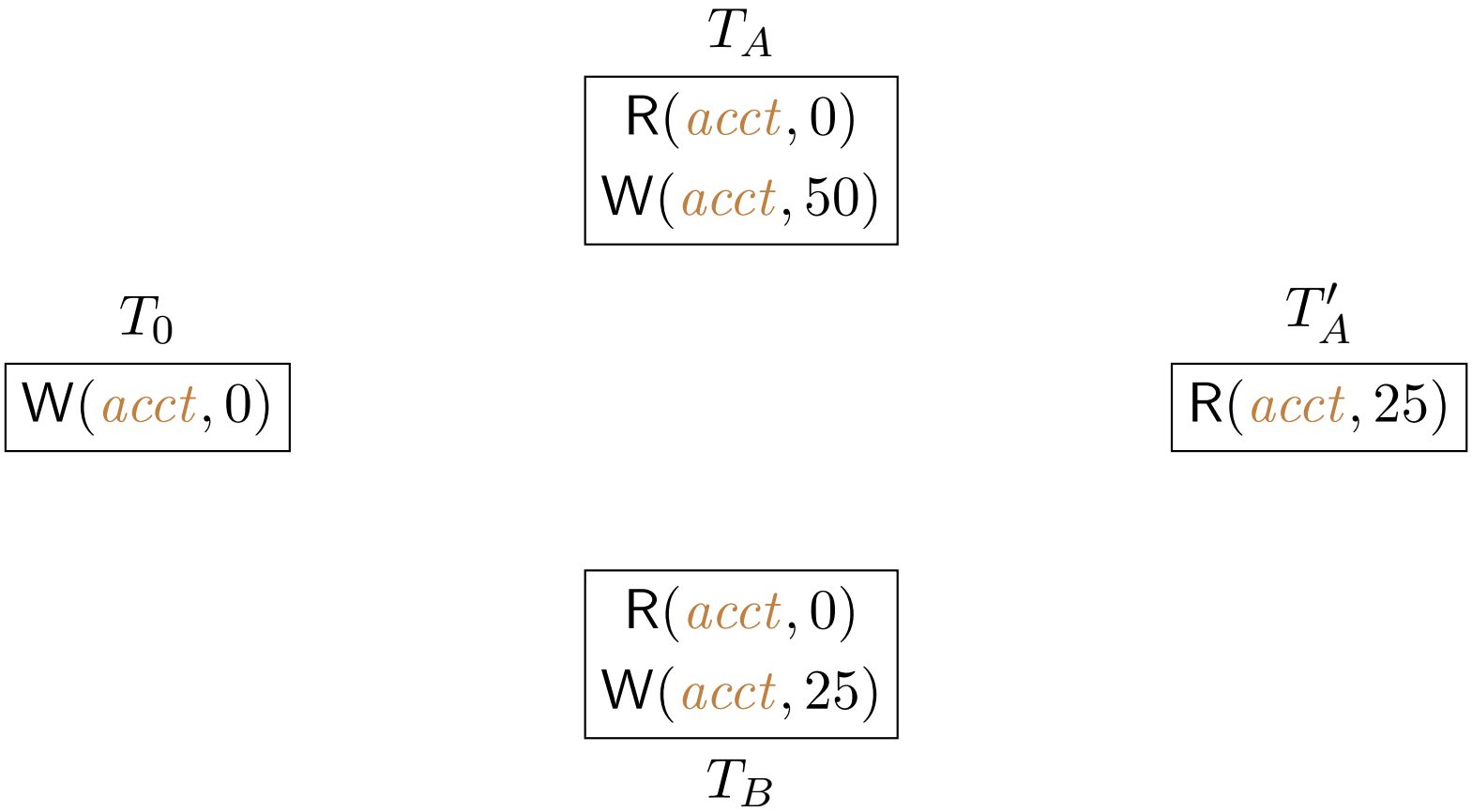
$\mathsf{W}(acct, 0)$

SI Prevents the "Lost Update" Anomaly



T_A and T_B are executed concurrently.

SI Prevents the "Lost Update" Anomaly



T_A and T_B are executed concurrently.

SI Prevents the "Causality Violation" Anomaly

$$T_A \boxed{\mathsf{W}(x, post)}$$

SI Prevents the "Causality Violation" Anomaly

$$T_A \boxed{\mathsf{W}(x, post)}$$

$$T_B \boxed{\begin{array}{c}\mathsf{R}(x, post)\\ \mathsf{W}(y, comment)\end{array}}$$

SI Prevents the "Causality Violation" Anomaly

$$T_A \boxed{\mathsf{W}(x, post)}$$

$$T_B \boxed{\begin{array}{c}\mathsf{R}(x, post)\\ \mathsf{W}(y, comment)\end{array}}$$

$$T_C \boxed{\begin{array}{c}\mathsf{R}(x, empty)\\ \mathsf{R}(y, comment)\end{array}}$$

SI Allows the "Write Skew" Anomaly

Databases that Claim to Support SI

database logos

Snapshot Isolation (SI)

Database systems may fail to provide SI as they claim.
+papers

The SI Checking Problem

Given a history H of a database system,

to decide whether H satisfies SI?

+fig

Motivation: Black-box SI Checker

Since the internals of database systems are often unavailable or are hard to understand,

a *black-box* SI checker is highly desirable.

Motivation: Black-box SI Checker

A black-box SI checker should be

Sound: return no false positives

Complete: miss no violations

Efficient: run in a reasonable time even for large workloads

Informative: report understandable counterexamples

Motivation: Black-box SI Checker

related-work

Contributions: PolySI

Sound & Complete: characterization of SI in terms of *generalized polygraphs*

Efficient: encoding into MonoSAT queries;
domain-specific pruning before solving

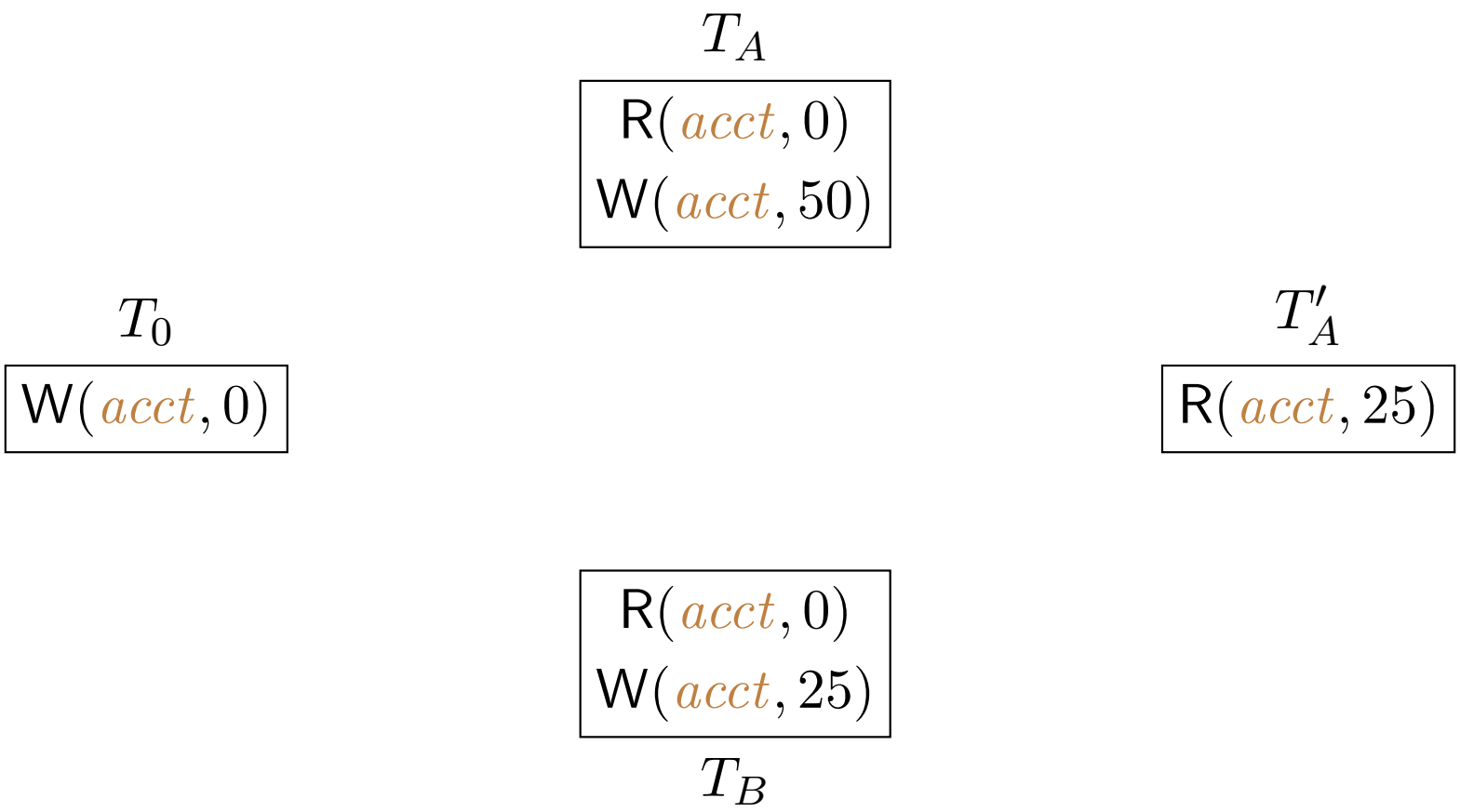
Informative: extract understandable counterexamples from MonoSAT unsatisifiable core

Contributions: PolySI

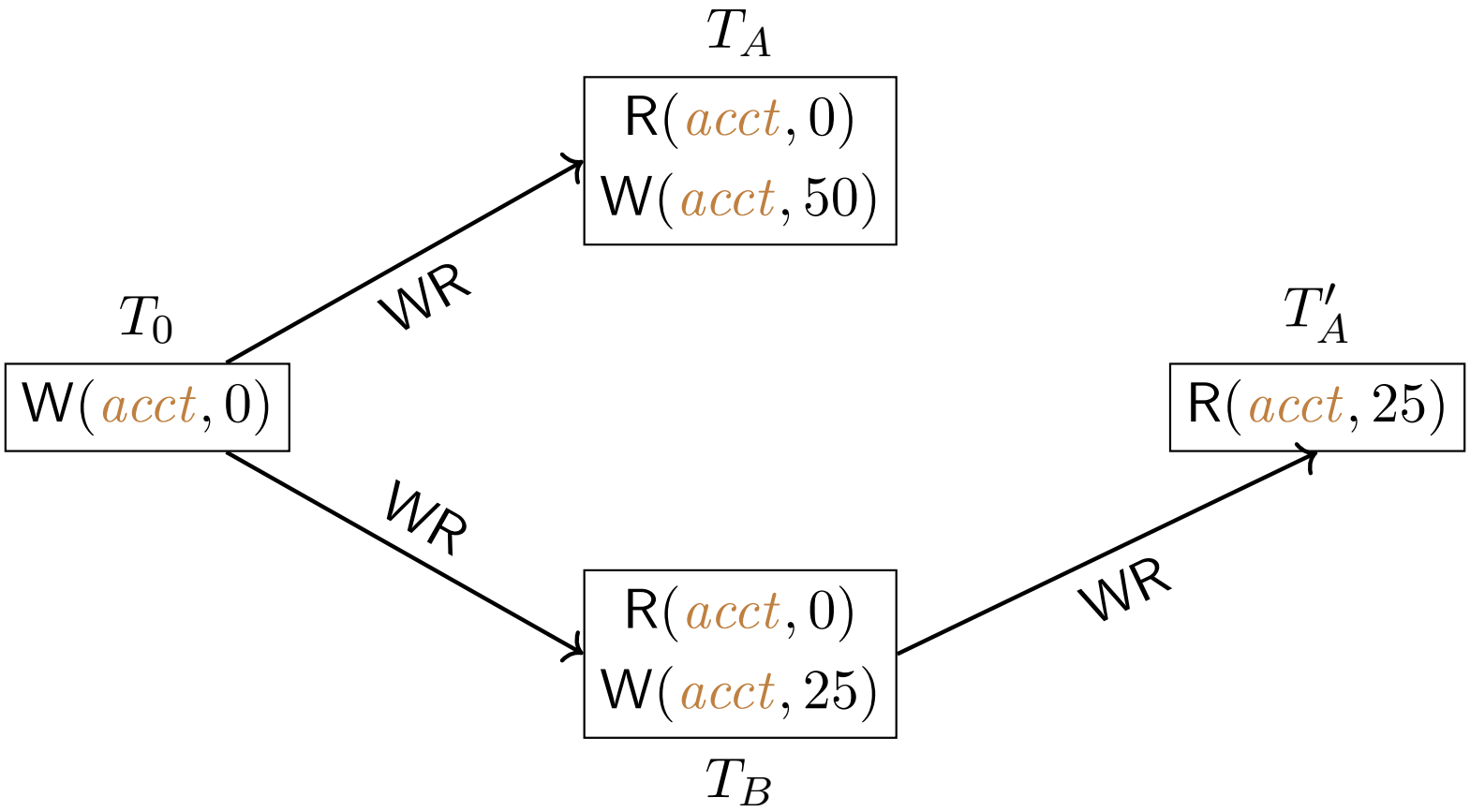
PolySI found SI violations in production database systems.

PolySI outperformed state-of-the-art black-box SI checkers and scales up to large workloads.

Characterization of Snapshot Isolation

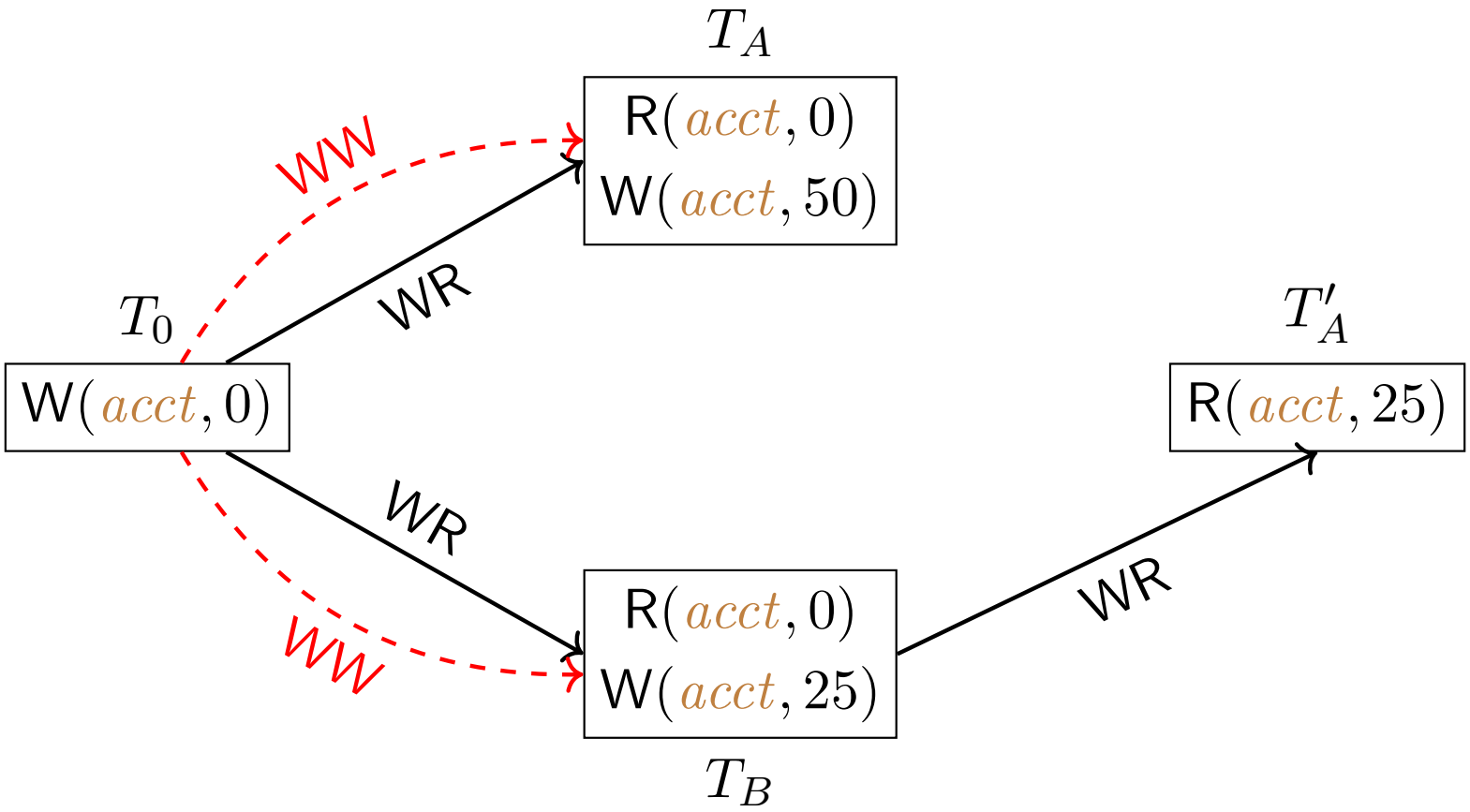


Characterization of Snapshot Isolation



WR: "write-read" dependency capturing the "read-from" relation

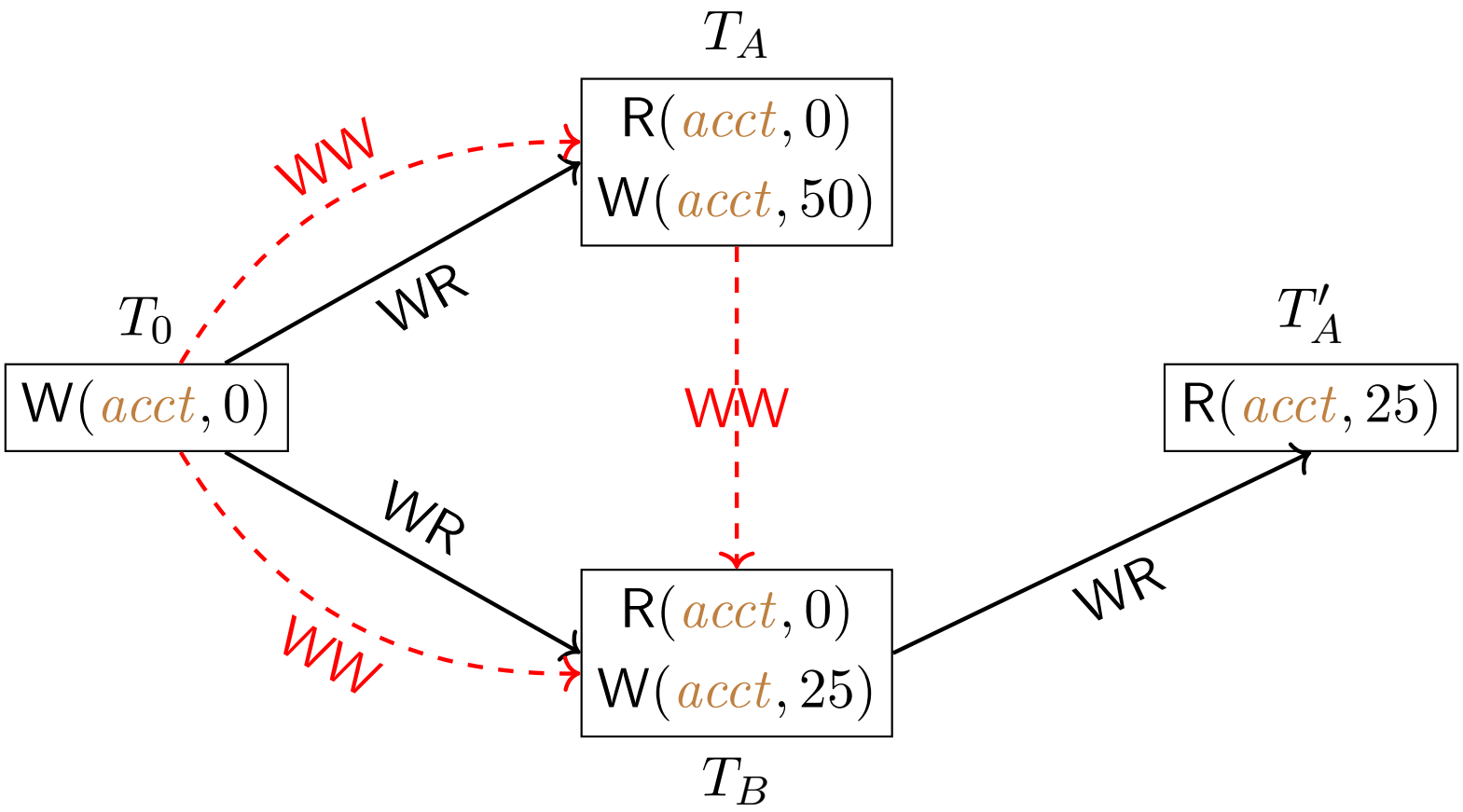
Characterization of Snapshot Isolation



WW: "write-write" dependency capturing the version order

Characterization of Snapshot Isolation

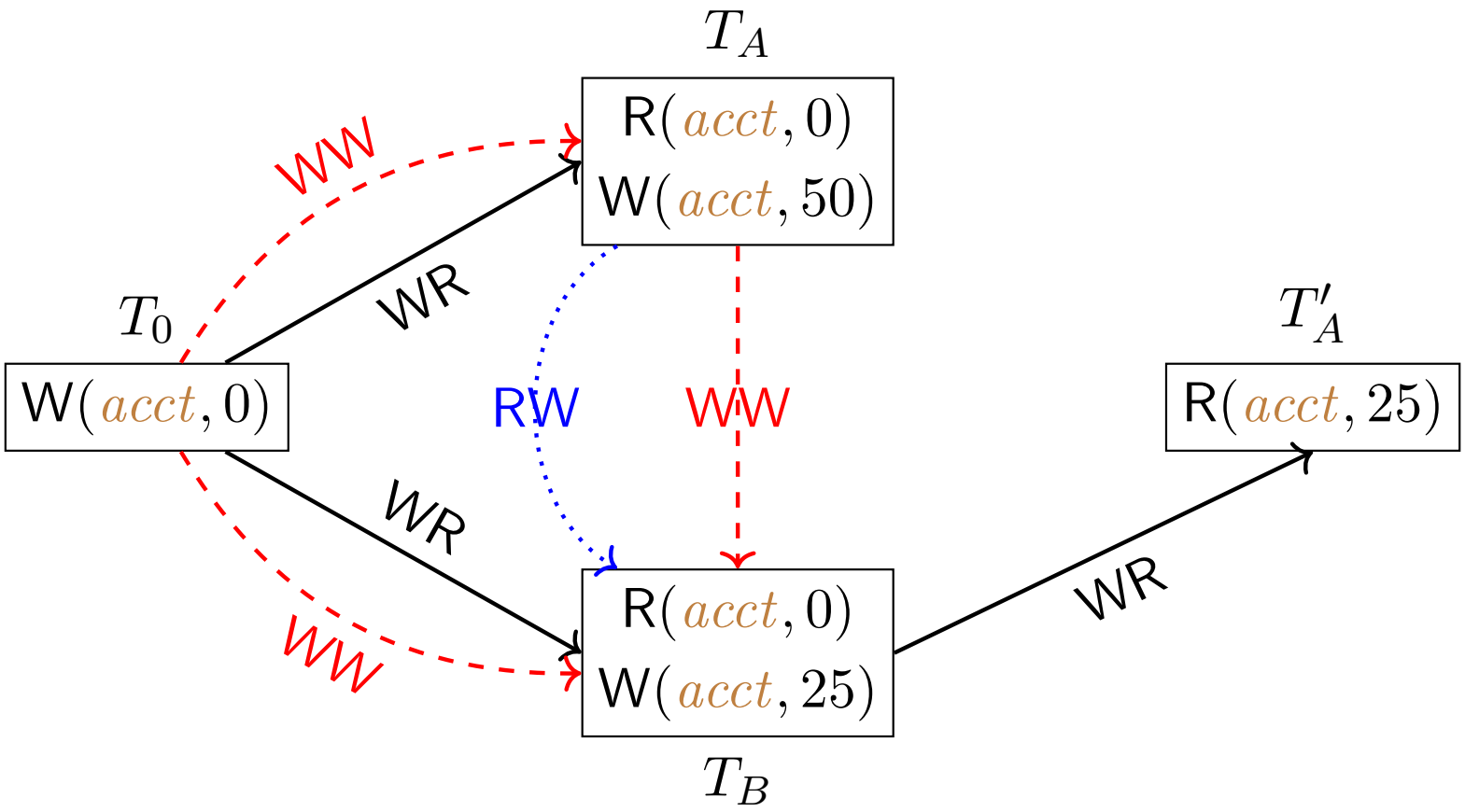
Suppose that $T_A \xrightarrow{\text{WW}} T_B$



WW: "write-write" dependency capturing the version order

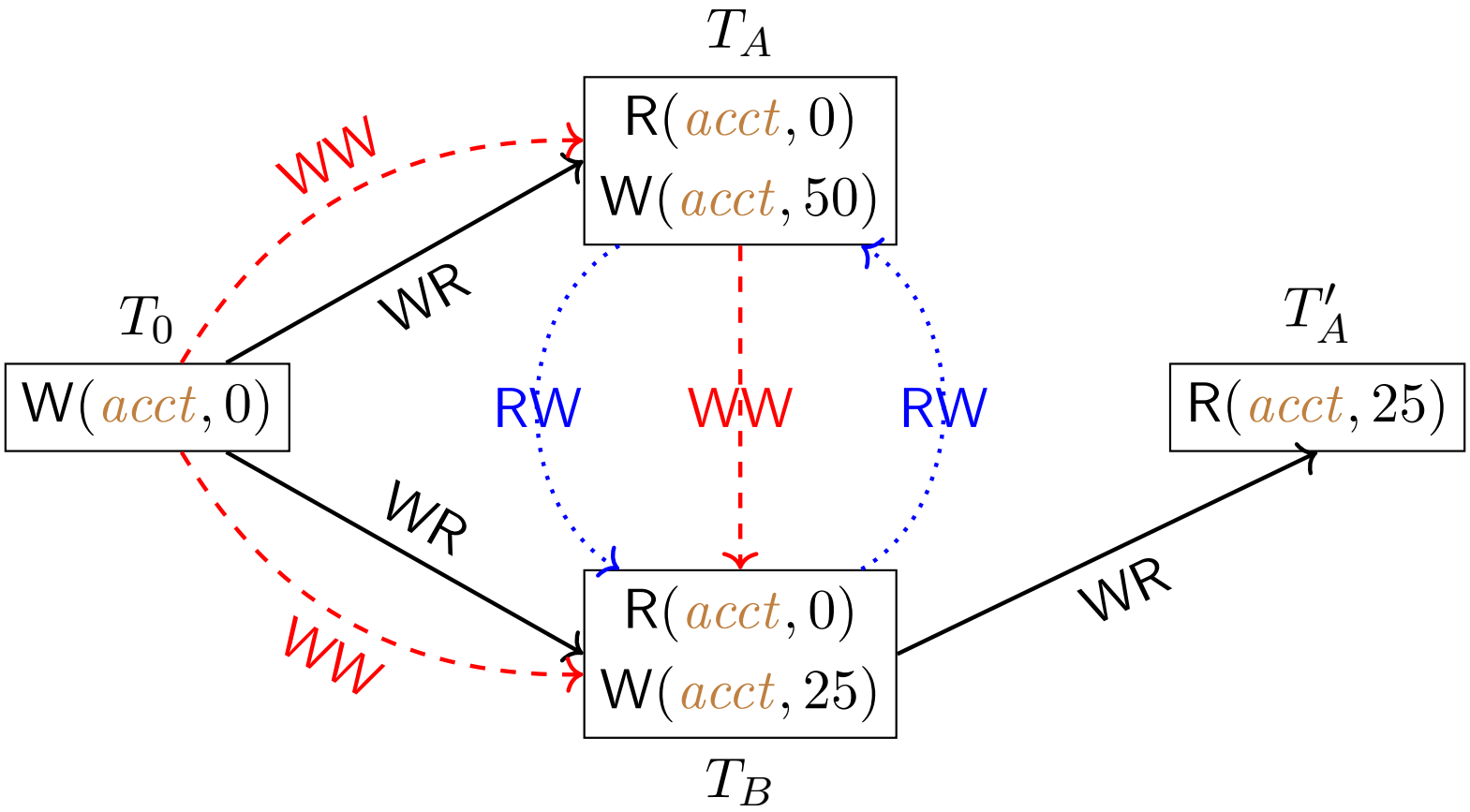
Characterization of Snapshot Isolation

Suppose that $T_A \xrightarrow{\text{WW}} T_B$



RW: "read-write" dependency capturing the overwritten relation

Characterization of Snapshot Isolation



undesiable cycle: $T_A \xrightarrow{\text{WW}} T_B \xrightarrow{\text{RW}} T_A$

Characterization of Snapshot Isolation

Hengfeng Wei (hfwei@nju.edu.cn)