Efficient Black-box Checking of Snapshot Isolation in Databases

(Conference VLDB'2024)

Hengfeng Wei

hfwei@nju.edu.cn

August 8, 2023

Database Transactions

A database transaction is a *group* of operations

that should be executed atomically.

Isolation Levels

Transactions may be executed concurrently.

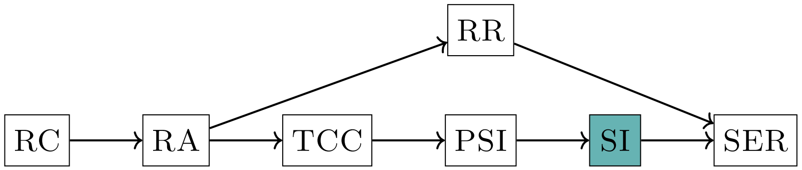
The isolation levels specify how they are isolated from each other.

Serializability (SER)

All transactions appear to execute serially, one after another.

too expensive, especially for distributed transactions

Snapshot Isolation (SI)



$$\begin{tikzcd}
 & & & RR \arrow[rr] & & \\
RC \arrow[r] & RA \arrow[r] \arrow[urr] & TCC \arrow[r] & PSI \arrow[r] & SI \arrow[r] & SER
\end{tikzcd}$$

Snapshot Isolation (SI)

example

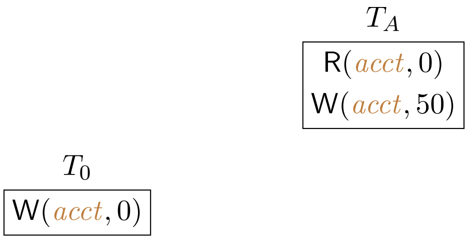
Snapshot Read:	Each transaction reads data from a snapshot of committed data valid as of the (logical) time the transaction started.
Snapshot Write:	Concurrent transactions cannot write to the same key. One of them must be aborted.

SI Prevents the “Lost Update” Anomaly

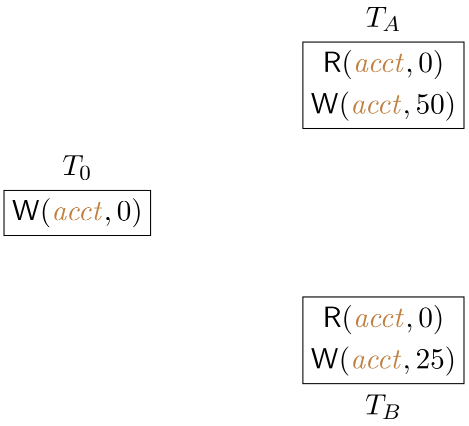
T_0

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

SI Prevents the “Lost Update” Anomaly

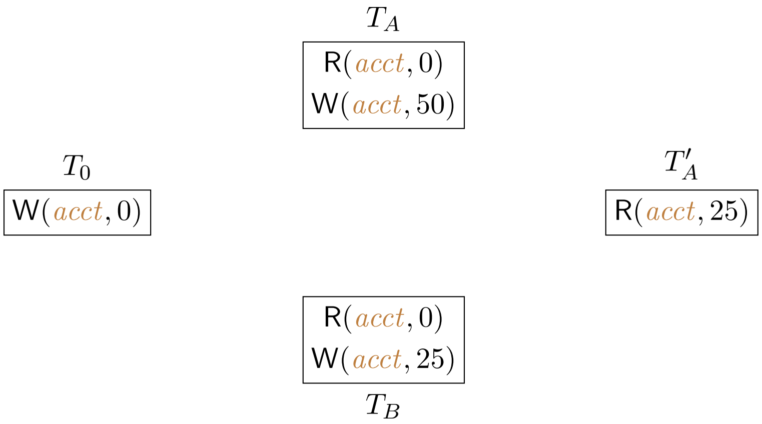


SI Prevents the “Lost Update” Anomaly



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

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

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

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

SI Allows the “Write Skew” Anomaly

Databases that Claim to Support SI

database logos

Snapshot Isolation (SI)

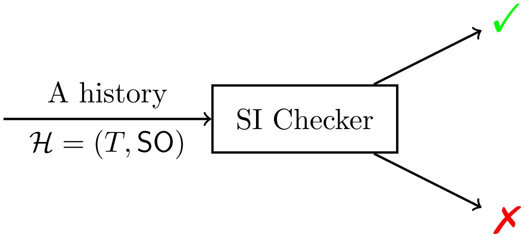
Database systems may fail to provide SI as they claim.

+papers

The SI Checking Problem

Definition (The SI Checking Problem)

The SI checking problem is the decision problem of determing whether a given history $\mathcal{H} = (T, \textsf{SO})$ satisfies SI?

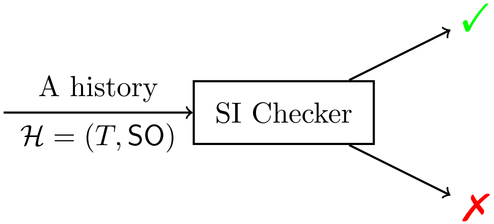


Motivation: Black-box SI Checker

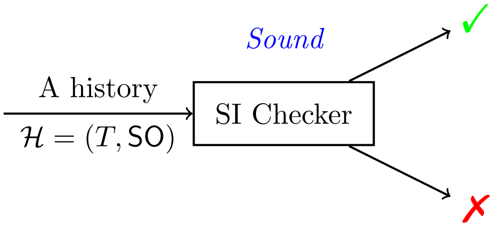
Since the internals of database systems are often unavailable or are hard to understand,

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

Motivation: Black-box SI Checker

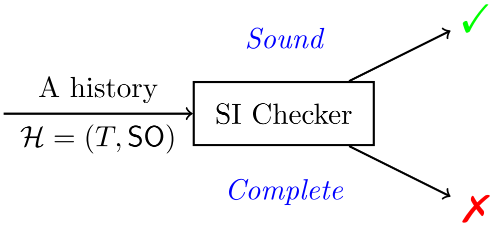


Motivation: Black-box SI Checker



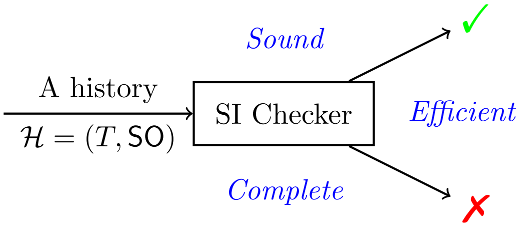
Sound: If the checker says ✗, then the history is not SI.

Motivation: Black-box SI Checker



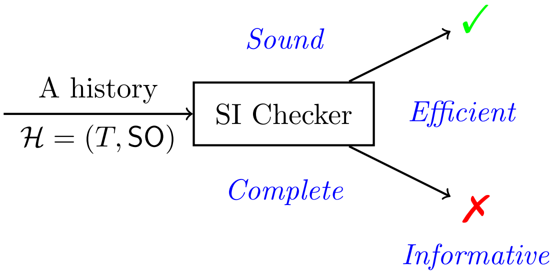
Complete: If the checker says ✓, then the history is SI.

Motivation: Black-box SI Checker



Efficient: The checker should scale up to large workloads.

Motivation: Black-box SI Checker

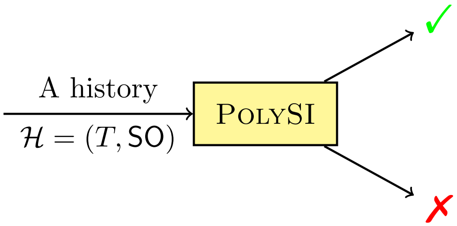


Informative: The checker should provide understandable counterexamples if it says ✗.

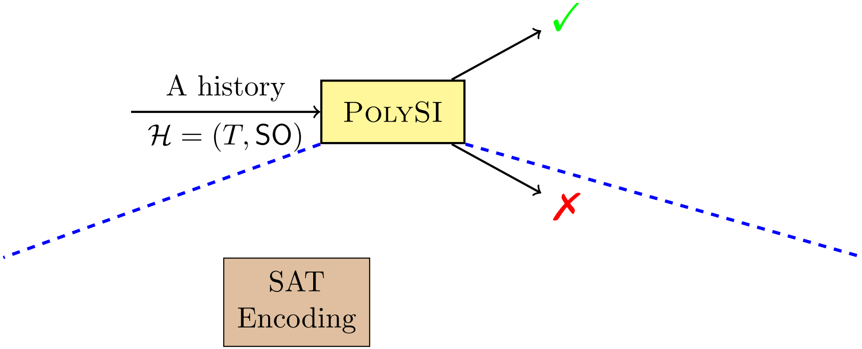
Motivation: Black-box SI Checker

related-work

Contributions: the PolySI Checker

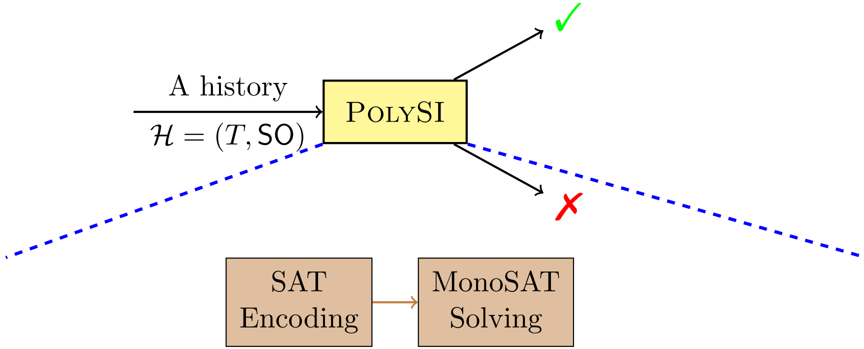


Contributions: the PolySI Checker



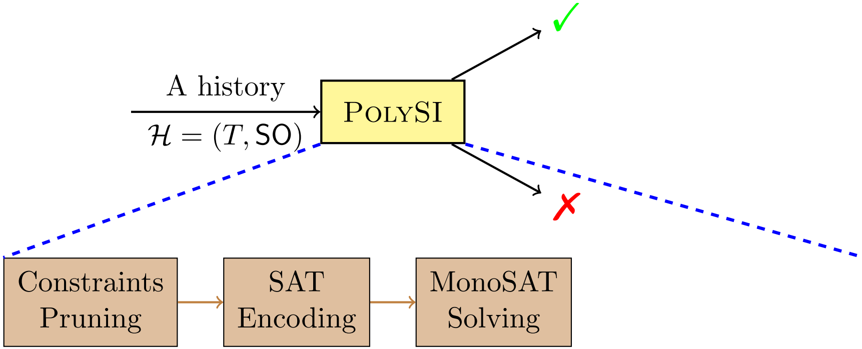
Sound & Complete: polygraph-based characterization of SI

Contributions: the PolySI Checker



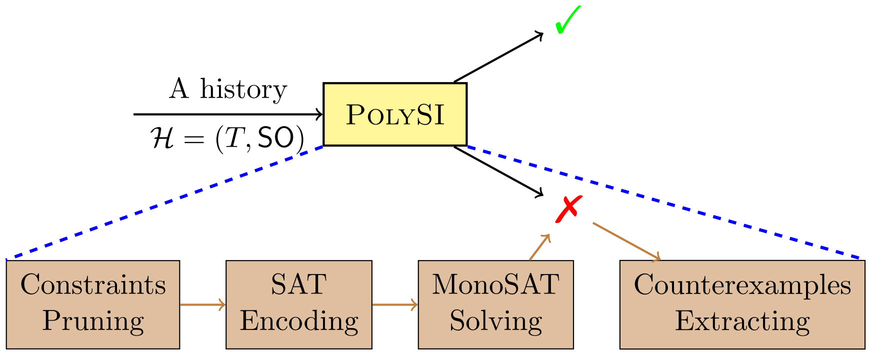
Efficient: utilizing MonoSAT solver optimized for graph problems

Contributions: the PolySI Checker



Efficient: domain-specific pruning before encoding

Contributions: the PolySI Checker



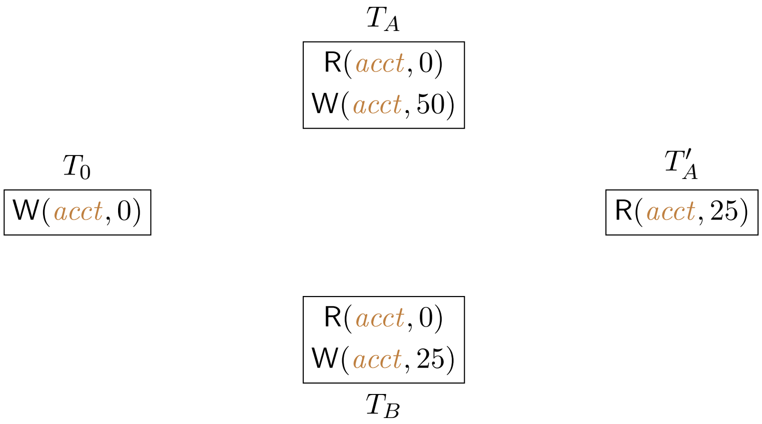
Informative: extract counterexamples from the unsatisifiable core

Contributions: PolySI

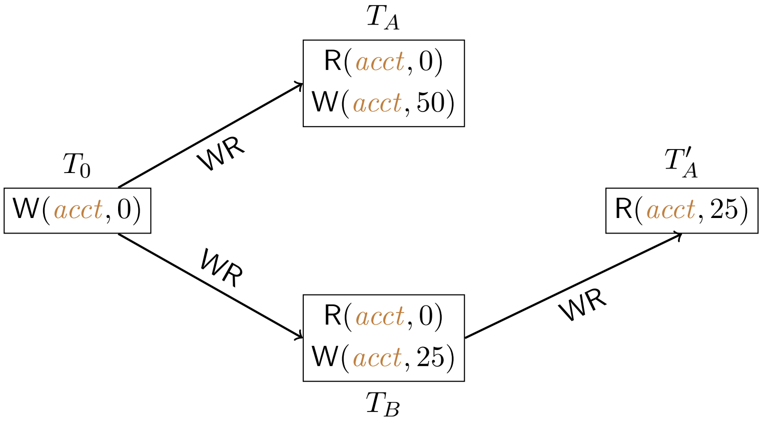
PolySI found SI violations in production database systems.

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

Dependency Graph-based Characterization of SI

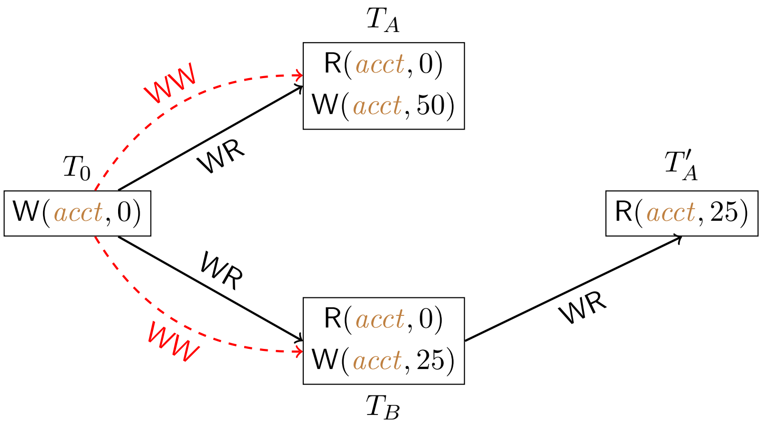


Dependency Graph-based Characterization of SI



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

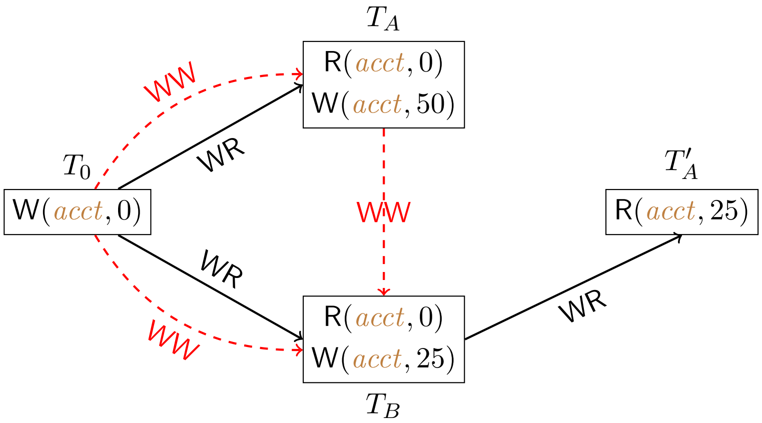
Dependency Graph-based Characterization of SI



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

Dependency Graph-based Characterization of SI

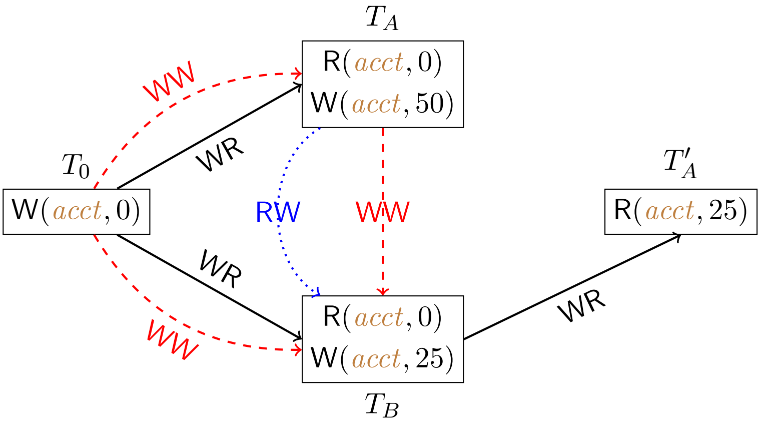
Suppose that $T_A \xrightarrow{\mathsf{WW}} T_B$



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

Dependency Graph-based Characterization of SI

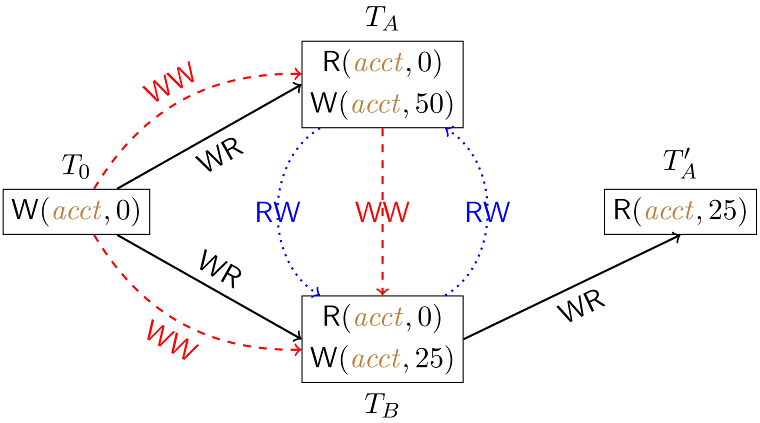
$$T_0 \xrightarrow{\mathsf{WR}} T_A \land T_0 \xrightarrow{\mathsf{WW}} T_B \implies T_A \xrightarrow{\mathsf{RW}} T_B$$



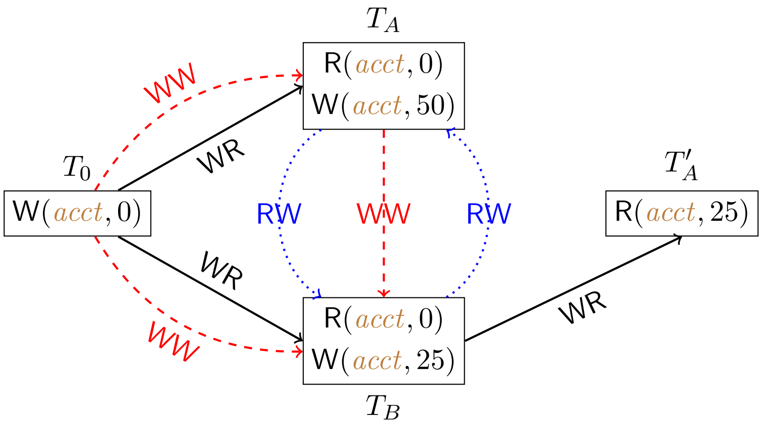
RW: “read-write” dependency capturing the overwritten relation

Dependency Graph-based Characterization of SI

$$T_0 \xrightarrow{\mathsf{WR}} T_B \wedge T_0 \xrightarrow{\mathsf{WW}} T_A \implies T_A \xrightarrow{\mathsf{RW}} T_A$$



Dependency Graph-based Characterization of SI



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

Dependency Graph-based Characterization of SI

Dependency Graph-based Characterization of SI

Dependency Graph-based Characterization of SI

SI is characterised by dependency graphs that contain only cycles with *at least two adjacent anti-dependency* edges.

Theorem (Theorem 4.1 of [Cerone and Gotsman, 2018])

For a history $\mathcal{H} = (T, \mathsf{SO})$,

$$\begin{aligned}\mathcal{H} \models \mathrm{SI} \iff \mathcal{H} \models \textsc{Int} \land \\ \exists \mathsf{WR}, \mathsf{WW}, \mathsf{RW}.\ \mathcal{G} = (\mathcal{H}, \mathsf{WR}, \mathsf{WW}, \mathsf{RW}) \land \\ (((\mathsf{SO}_{\mathcal{G}} \cup \mathsf{WR}_{\mathcal{G}} \cup \mathsf{WW}_{\mathcal{G}})\ ;\ \mathsf{RW}_{\mathcal{G}}?)\ \ \textit{is acyclic}).\end{aligned}$$

Hengfeng Wei (hfwei@nju.edu.cn)

Cerone, Andrea and Alexey Gotsman (Jan. 2018). “Analysing Snapshot Isolation”. In: *J. ACM* 65.2. ISSN: 0004-5411. DOI: 10.1145/3152396. URL: https://doi.org/10.1145/3152396.